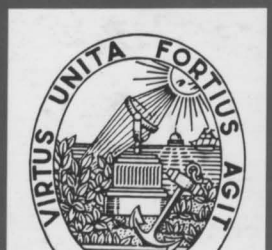
UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA

# Linguagem de Comunicação entre Aplicações de Gestão Autónomas

Bruno Filipe Lopes Garcia Marques

Porto, Dezembro de 2000



# Linguagem de Comunicação entre Aplicações de Gestão Autónomas

Bruno Filipe Lopes Garcia Marques

Porto, Dezembro de 2000

# Linguagem de Comunicação entre Aplicações de Gestão Autónomas

Bruno Filipe Lopes Garcia Marques
(Licenciado em Engenharia Electrotécnica
pela Escola Superior de Tecnologia, integrada no
Instituto Superior Politécnico de Viseu)

Dissertação submetida para satisfação parcial dos
requisitos do grau de Mestre em
Engenharia Electrotécnica e de Computadores
(área de especialização de Informática Industrial),

Realizada sob a orientação do
Professor Doutor Raúl Filipe Teixeira de Oliveira,
do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

*À minha namorada Cláudia,*
*pela paciência e por ter acreditado em mim;*

*Aos meus pais,*
*e à minha tia Rosa Maria Felgar,*
*por tudo aquilo que por mim fazem fazem.*

# Abstract

The software agent paradigm has evolved from a specialized stream of distributed artificial intelligence. However, thought the first work on agents was as long ago as the late 1970s, the software agent paradigm has taken a considerable amount of time to come to fruition in the communications domain, especially in the telecommunications world. At the present time, interest in the application of the software agent paradigm to communications systems has grown enormously, mainly because it is perceived as a powerful means to satisfy the commercially important need for software systems to inter-operate and to manage large heterogeneous networks. Therefore, communications infrastructures are now seen as a natural application domain for Software Agents.

The approach followed in this research work lead to the development of an new *multi-agent architecture*, suitable to an context of ATM circuits service providing between private networks over wyde area networks. Here the agents are responsible for asking information on services and respective costs, to negotiate and to establish a qualified service and at the request of an agent on behalf of a final user (private net). With the agents, we expect to reduce the human factor, nowadays found, by dynamicaly establishing communication circuits and to make an better improvement of net resources.

Beside the development of this architecture and the definition of possible communication sceneries, it was necessary the specification of a content language that, used by the agents, got in a transparent way to integrate the distributed resolution of a problem (the one of the dynamic establishment of a communication circuit specified by a end-user) in an atmosphere of the heterogeneous telecommunications networks. Thus, we succeeded an multi-agent hybrid and reactive architecture.

All the tied up concepts to this new technology (terminologies, communication and content languages concepts) in what it refers to a context of telecommunications networks, are explained and detailed in the thesis.

# Résumé

Le paradigme des agents de software s'est développé en partant du domaine de l'intelligence artificielle. Pourtant quoique le premier travail sur les agents ait été réalisé il y a quelques années (je parle de la décade de soixante-dix), le paradigme des agents intelligents a pris longtemps à devenir réalité dans le monde des télécommunications. À présent, on assiste à un large développement de l'intérêt dans l'utilisation d'applications autonomes de software (agents) aux systémes de télécommunications, à cause surtout de la conscience que ce paradigme peut offrir de meilleurs moyens pour l'interligation et la gestion de grandes voies de télécommunications. Ainsi, les structures des communications sont maintenant envisagées comme un domaine naturel de l'applicabilité des agents intelligents.

L'approche suivie dans ce travail de maîtrise a conduit au développement d'une nouvelle architecture multiagent adaptée à tout un context de livraison de services ATM, de liaison entre les réseaux privés à travers une ou plusieurs voies opératrices de zone élargie. Dans ce cas, les agents prennent la responsabilité de demander des informations sur des services et sur leur coût, de négocier et d'établir un service qualifié et sur demande d'un agent au nom d'un usager final (voie privée). Avec les agents on cherche à réduire le facteur humain du présent, en établissant dynamiquement des circuits de communication et en mieux profitant les ressources présents dans l'équipement des réseaux.

Outre le développement de cette architecture et de la définition des possibles encadrements de communication, on a eu besoin de la spécification d'un langage de contenu qui, étant utilisé par les agents, pourrait intégrer si clairement que possible la résolution distribuée d'un problème (celui de l'établissement dynamique d'un circuit de communication spécifié par l'usager final), dans un entourage de réseaux hétérogènes. Ainsi, on est arrivé à la définition d'une architecture d'agents hybrides et réactifs, puisqu'ils agissent dans leur environment, en intervenant dans les équipements de voie d'accord à la connaissance qu'ils en possèdent.

Tous les concepts liés à cette nouvelle téchnologie (terminologies, concepts de langages de communication et de langages de contenu), en ce qui concerne le contexte de voies de télécommunications, sont expliqués en détail dans la présente thèse.

# Resumo

O paradigma dos agentes de software desenvolveu-se a partir da área da inteligência artificial. No entanto, apesar do primeiro trabalho sobre os agentes ter sido efectuado há já bastantes anos (falo da década de 70), o paradigma dos agentes inteligentes levou bastante tempo a tornar-se realidade no mundo das telecomunicações. Presentemente, tem-se assistido a um grande crescimento no interesse da utilização de aplicações autónomas de software (agentes) nos sistemas de telecomunicações devido, principalmente, à consciência de que este paradigma pode oferecer melhores meios para a interligação e gestão de grandes redes com características acentuadamente heterogéneas como é o caso das redes de telecomunicações. Assim sendo, as estruturas das comunicações são agora vistas como um domínio natural da aplicabilidade dos agentes inteligentes.

A aproximação seguida neste trabalho de mestrado conduziu ao desenvolvimento de uma nova arquitectura multi-agente, apropriada a todo um contexto de fornecimento de serviços ATM de ligação entre redes privadas através de uma ou mais redes operadoras de área alargada. Neste, os agentes são responsáveis por pedir informações sobre serviços e respectivos custos, negociar e estabelecer um serviço qualificado e a pedido de um agente em nome de um utilizador final (rede privada). Com os agentes, pretende-se reduzir o factor humano actualmente encontrado, estabelecendo-se dinâmicamente circuitos de comunicação e fazendo um melhor aproveitamento dos recursos existentes no equipamento das redes.

Para além do desenvolvimento desta arquitectura e definição dos possíveis cenários de comunicação, foi necessária a especificação de uma linguagem de conteúdo que, utilizada pelos agentes, conseguisse de uma forma transparente integrar a resolução distribuída de um problema (o do estabelecimento dinâmico de um circuito de comunicação especificado pelo utilizador final) num ambiente de redes heterogéneas. Assim, conseguiu-se definir uma arquitectura de agentes híbridos e reactivos, já que eles agem no seu meio ambiente, actuando nos equipamentos de rede de acordo com o conhecimento que dele possuem.

Todos os conceitos ligados a esta nova tecnologia (terminologias, conceitos de linguagens de comunicação e de linguagens de conteúdo), no que se refere a um contexto de redes de telecomunicações, são explicados em detalhe na presente tese.

# Agradecimentos

Agradeço ao Prof. Doutor Raúl Filipe Teixeira Oliveira a confiança que depositou em mim, propondo-me este trabalho, a disponibilidade que sempre manifestou na sua orientação e os estreitos laços de amizade que se criaram.

Agradeço também à minha colega Rita Marques, pela forma de troca de conhecimentos e experiências dentro da nossa equipe de investigação.

Aos outros amigos e colegas que me incentivaram, e que não necessito nomear, a minha gratidão.

A todos os que partilham com a comunidade tempo e conhecimentos, participando no desenvolvimento do SunOS, do Solaris, do Linux, do Emacs, do Java, do TEX e do LATEX, um muito obrigado pelo quanto facilitaram a minha actividade.

Por último, o meu agradecimento ao Departamento de Engenharia Electrotécnica da Escola Superior de Tecnologia, integrada no Instituto Superior Politécnico de Viseu, pelos meios materiais disponibilizados na elaboração deste trabalho.

# Conteúdo

# Lista de Tabelas

# Lista de Figuras

# Nomenclatura

## Abreviaturas

| | |
|---|---|
| ABR | *Available Bit rate* |
| ACL | *Agent Communication Language* |
| AI | *Artificial Intelligence* |
| API | *Application Program Interface* |
| ASP | *ATM Service Provider* |
| ASPA | *ATM Service Provider Agent* |
| ATM | *Asynchronous Transfer Mode* |
| BDI | *Beliefs, Desires and Intensions* |
| BNF | *Backus-Naur Form* |
| CAC | *Connection Admission Protocol* |
| CBR | *Constant Bit Rate* |
| CORBA | *Common Object Request Broker Architecture* |
| DAI | *Distributed Artificial Intelligence* |
| DARPA | *Defense Advanced Research Projects Agency* |
| DPE | *Distributed Processing Environment* |
| DPS | *Distributed Problem Solving* |
| FA | *Facilitator Agent* |
| FIPA | *Foundation for Intelligent Physical Agents* |
| IA | *Intelligent Agents* |

| | |
|---|---|
| IP | *Internet Protocol* |
| ITU | *International Telecommunication Union* |
| JKQML | *Java API for KQML* |
| JVM | *Java Virtual Machine* |
| KIF | *Knowledge Interchange Format* |
| KQML | *Knowledge Query and Manipulation Language* |
| KSF | *Knowledge Sharing Effort* |
| LAN | *Local Area Network* |
| MAS | *Multi-Agent System* |
| MBS | *Maximum Burst Size* |
| MCR | *Maximum Cell Rate* |
| PCR | *Peak Cell Rate* |
| PVC | *Permanent Virtual Channel* |
| QdS | *Qualidade de Serviço* |
| QoS | *Quality of Service* |
| SCR | *Sustainable Cell Rate* |
| SLA | *Service Level Agreement* |
| RMI | *Remote Method Invocation* |
| RSVP | *Resource Reservation Protocol* |
| SPVC | *Soft Private Virtual Circuit* |
| SVC | *Switched Virtual Circuit* |
| UA | *User Agent* |
| UBR | *Unspecified Bit Rate* |
| VBR | *Variable Bit Rate* |
| VCI/VPI | *Virtual Channel Identifier / Virtual Path Identifier* |
| VPN | *Virtual Private Network* |
| WAN | *Wide Area Network* |

# Capítulo 1

# Introdução

A crescente liberalização do mercado das Telecomunicações tem levado ao aparecimento de novas empresas operadoras (ou à reestruturação de existentes), capazes de oferecerem serviços diferenciados, de acordo com as necessidades individuais dos respectivos clientes.

A adopção da tecnologia ATM (Asynchronous Transfer Mode) neste mercado fez com que estas empresas ficassem a poder oferecer serviços de ligação de elevada qualidade entre redes privadas. Na realidade, quando se estabelecem ligações entre redes privadas e redes operadoras com serviços ATM, que adiante designaremos por ASPs (ATM Service Providers), maior faculdade de escolha possuirá um utilizador final. Actualmente, algumas das redes operadoras começam a oferecer largas configurações de acesso regular, nomeadamente circuitos ATM entre redes privadas locais e/ou de longa distância. No entanto, ainda persistem algumas dificuldades que têm de ser ultrapassadas. Por um lado, existe um conjunto de paradigmas quando um cliente pede por um serviço de ligação extremo-a-extremo com uma qualidade de serviço - QdS, (também por ele especificada), podendo este atravessar redes ATM distintas. Por outro lado, existem diferentes protocolos e formas de encaminhamento de dados não suportados por ligações virtuais comutadas (*Switched Virtual Connections* - SVCs). Adicionalmente, devido ao facto da interoperabilidade de diferentes domínios ATM se encontrar ainda em fase de desenvolvimento, os ASPs têm que desenvolver meios próprios para controlar e gerir os acessos dos seus clientes, muito possivelmente compelindo uma política de admissão de ligações. Todos estes aspectos prefazem um conjunto de características incompatíveis, se vistas do prisma da análise de redes heterogéneas ATM.

Usualmente, a prestação de um serviço de ligação estabelece-se à custa de um factor humano, isto é, os circuitos são abertos manualmente, o que torna o processo lento. Com o presente trabalho, propõe-se a utilização de uma arquitectura multi-agente com o objectivo de tentar eliminar, não só a demora no estabelecimento de circuitos, como também ultrapassar os problemas referidos. Recorrendo a agentes poder-se-á abrir circuitos a pedido, e de forma imediata, pois a comunicação entre uma rede privada e uma rede de operador passará a ser automática. Outra vantagem da utilização deste tipo de solução reside no facto de se permitir manter uma qualidade de serviço superior à que se obtém com a intervenção do referido factor humano.

No seguimento do exposto, o objectivo principal do trabalho efectuado é o desenvolvimento de uma arquitectura multi-agente, inserida no contexto do pedido de estabelecimento de circuitos de comunicação entre redes privadas, através de uma ou mais redes públicas de área alargada. A arquitectura proposta é composta por agentes que agem em nome de um utilizador final e por agentes que operam em nome das redes operadoras, fornecendo informações sobre as características dos serviços de ligação que podem prestar, assim como custos a eles associados, e tentam negociar contratos de prestação de serviços de ligação.

Para completar a nossa arquitectura, é necessário que os seus agentes possuam um meio de comunicação eficiente, isto é, possuir um mecanismo que lhes possibilite a troca de informações e de conhecimento, correspondendo o mesmo à existência de uma linguagem comum de comunicação. Sendo uma linguagem de comunicação o mecanismo de transporte de mensagens entre agentes, torna-se necessária a compreensão do seu conteúdo. Aqui, entram em acção as chamadas linguagens de conteúdo que fornecem os aspectos sintáticos da representação da troca de informações e de conhecimento passados através de mensagens. Neste sentido, torna-se necessário especificar uma linguagem de conteúdo que, sendo utilizada pela linguagem de comunicação entre agentes optada (KQML), permita aos referidos agentes o mútuo entendimento e conhecimento das tarefas respectivamente atribuídas na arquitectura.

## 1.1 Agentes Inteligentes

A utilização da tecnologia dos agentes no problema da gestão de redes confere uma nova dimensão à gestão e ao controlo do tráfego em redes ATM, adicionando uma coordenação entre pontos de comutação, considerada uma grande falha em sistemas semelhantes [1]. Com ela, espera-se adquirir as vantagens de: redução da intervenção humana no estabelecimento de circuitos devido à eliminação do processo de configuração passo-a-passo; expansão da inteligência até às fronteiras das redes, reduzindo a dependência da integração de equipamento de comutação entre diferentes operadoras; atribuição de funções lógicas a agentes particulares, aliviando a necessidade da intervenção humana; codificação dos processos decisionais efectuada por uma forma standard pelos agentes; utilização mais eficiente dos recursos já que os agentes os optimizam; as redes tornam-se mais flexíveis e escaláveis devido à natureza adaptativa e distribuída dos agentes; confiabilidade nas redes, uma vez que os agentes possibilitam o controlo distribuído onde as decisões para o estabelecimento de circuitos não são apenas tomadas tendo em conta a qualidade do serviço, mas também outros factores como o custo; opção de seleccionar o operador e em consequência reduzir o custo de utilização de circuitos. Por outro lado, o facto de se poder abrir circuitos a pedido durante intervalos de tempos limitados e variáveis no tempo é outra vantagem.

## 1.2 Os Agentes na Gestão de Redes

Os agentes podem ser vistos segundo uma tecnologia tridimensional [2]: exibem propriedades características, comunicam por intermédio de um conjunto de linguagens e são desenvolvidos seguindo determinadas arquitecturas. Estas três dimensões tornam os agentes especialmente apropriados na substituição do Homem por aplicações de *software* baseados na tecnologia dos agentes. É muito comum, quando nos referimos à gestão e controlo de redes, encontrar operadores humanos a interagirem com utilizadores finais.

As quatro propriedades mais importantes dos agentes, autonomia, actividade, reactividade e sociabilidade, são características fundamentais para aplicações que funcionam em nome de um utilizador final ou de uma operadora fornecedora de serviços ATM, por forma a negociar contratos de tráfego, ou encontrar e filtrar informações de gestão num ambiente distribuído e complexo.

Para que os agentes possam interagir eficientemente é imperativa a existência de três componentes fundamentais: uma linguagem comum, uma compreensão comum do conhecimento trocado e a habilidade para trocar tudo o que é definido no nosso contexto de actuação [3]. No entanto, por forma a poderem comunicar, os agentes utilizam diferentes tipos de linguagens de comunicação que lhes permitem trocar informações correspondentes com as suas propriedades e com as tarefas que enfrentam. Os agentes utilizam estas linguagens de comunicação para desenvolverem actos de comunicação do tipo declarativo, interrogativo e permissivo. A semântica da informação passada através dos actos de comunicação é expressa por linguagens de conteúdo que podem ser estendidas a ontologias particulares, associadas ao conteúdo.

Uma arquitectura multi-agente deve basear-se em uma de três aproximações: reactiva, deliberativa [4] ou híbrida [5]; e adaptar-se ao domínio do problema, especialmente no que diz respeito à parte reactiva, a responsável pelo ponto de ligação dos agentes ao meio ambiente real.

### 1.2.1 Linguagens de Comunicação e Linguagens de Conteúdo

A utilização de uma linguagem de comunicação, como o KQML [6], requer uma linguagem de conteúdo para representar o conhecimento transportado por actos de comunicação (adiante referidos como *performativas*). Esta linguagem de conteúdo pode ser, por exemplo, o KIF [7], ou outra linguagem que satisfaça os requisitos da habilidade de modelizar o conhecimento.

A linguagem de comunicação **KQML** pressupõe implicitamente uma arquitectura multi-agente baseada em agentes regulares e agentes facilitadores. Os agentes regulares prestam serviços do domínio do problema e os agentes facilitadores fornecem as infraestruturas aos agentes regulares, para facilmente construirem um sistema multi-agente. Por exemplo, um agente facilitador aceita o registo dos agentes regulares e a informação sobre as habilidades e interesses dos mesmos. Por estes motivos, os agentes facilitadores superiorizam-se aos agentes regulares, sendo facilmente integráveis em sistemas multi-agente.

## 1.3 Metodologia a Adoptar

Para se poderem atingir os objectivos atrás referidos, é necessário, primeiro, estudar a forma como os agentes comunicam e interpretam o conteúdo das mensagens por eles trocadas, e segundo especificar uma linguagem de conteúdo que se adapte aos propósitos da arquitectura aqui proposta.

Como linguagem de comunicação entre agentes (*Agent Communication Language* - ACL) adoptar-se-á o KQML, uma linguagem de mensagens tipadas, usualmente entendidas como performativas, codificadas como cadeias alfanuméricas ASCII seguindo uma sintaxe semelhante ao LISP, e transportadas através de ligações TCP/IP com o objectivo de trocar conhecimento e informação entre aplicações de software. Esta linguagem, sendo o meio de transporte dos actos de comunicação entre os agentes, requer uma linguagem de conteúdo que satisfaça os requisitos dos elementos da arquitectura no que diz respeito à troca de conhecimento (informações, intenções, etc.), isto é, é necessário especificar uma linguagem de conteúdo por forma a permitir que os agentes interpretem convenientemente as mensagens trocadas.

## 1.4 Organização da Tese

A presente tese divide-se em seis capítulos, incluindo este e o último onde se apresentam as principais conclusões tiradas do trabalho desenvolvido e se formulam algumas propostas para um trabalho futuro.

O capítulo 2 descreve o estado actual dos agentes inteligentes na área das telecomunicações, apresentando algumas definições dos mesmos, a noção de arquitectura multi-agente salientando a importância da coordenação entre os agentes, e os desafios que eles enfrentam nesta área.

O capítulo 3 descreve alguns dos sistemas de comunicação entre os agentes, isto é, as linguagens de comunicação como o KQML e o Standard FIPA ACL. Para além da sua apresentação, é feita ainda uma comparação entre elas, e tenta-se dar alguns conselhos quanto à opção de uma ou de outra. Como as linguagens de comunicação apenas fazem o transporte das mensagens, é necessária a utilização de uma linguagem de conteúdo que implemente o conteúdo dos actos de comunicação. Aqui, apresenta-se uma breve visão de uma das primeiras linguagens de conteúdo a aparecer e que permitiu o desenvolvimento de outras: refiro-me à linguagem KIF.

O capítulo 4 apresenta a proposta de uma arquitectura multi-agente para permitir a automatização dos pedidos de estabelecimento de circuitos de comunicação entre redes privadas através de redes operadoras públicas, tornando esse processo dinâmico e fazendo um melhor aproveitamento dos recursos das redes. Aqui, os aspectos funcionais da rede encontram-se bem identificados e, por fim, são definidos todos os cenários de comunicação entre os agentes, onde se mostra como é utilizada a linguagem de comunicação KQML para transporte das mensagens trocadas entre os agentes.

Por fim, o capítulo 5 faz a apresentação da linguagem de conteúdo, especificada de

raíz para esta arquitectura, justificando o seu "baptismo" com o nome de MngmtCL e a utilização de uma ontologia com o nome CAC. Sendo o objectivo principal da arquitectura negociar e estabelecer contratos de prestação de serviço, a linguagem de conteúdo é constituída por um conjunto de expressões responsáveis por permitir que os agentes possam trocar conhecimento e actuar de acordo com os papéis que lhes foram atribuídos na definição da arquitectura. Assim, encontramos expressões consideradas como de base que se constituem à custa de outras, e se identificam com os objectivos de cada acto de comunicação, sendo por isso independentes das tecnologias físicas utilizadas na implementação da arquitectura. Para além destas, foram definidas outras expressões que, estando muito dependentes das tecnologias e protocolos físicos possíveis de ser utilizada(o)s, prefazem os chamados níveis tecnológicos da linguagem de conteúdo.

# Capítulo 2

# Agentes Inteligentes e os Sistemas de Telecomunicações

## 2.1 Introdução

Um dos grandes desafios que se pôem à automatização de tarefas de gestão em redes de telecomunicações é a supressão do factor humano através da introdução de entidades de software que colhem, processam e organizam informação, e possuem uma habilidade inata para actuarem nos seus equipamentos de controlo, de comutação e de encaminhamento - este é o modelo de gestão emergente de estudos recentemente desenvolvidos. Para realizar esta visão, as plataformas de telecomunicações enfrentam um novo paradigma: o paradigma dos agentes de software. Alguns investigadores desta área defendem que, sem a adopção deste paradigma, as vantagens que as novas tecnologias de área alargada oferecem (rápido desdobramento de serviços, flexibilidade lógica de configuração, e escalabilidade na gestão) não se materializarão.

Neste contexto, o presente capítulo faz a introdução a esse novo paradigma (a tecnologia dos agentes) e justifica porque razão a sua adopção trará benefícios tanto a uma rede operadora, como a um utilizador final, de tal forma que os sistemas de comunicação podem ser geridos mais eficazmente. Com ele, pode assistir-se a um aumento de produtividade e à oferta de novos serviços de valor acrescentado.

## 2.2 Algumas Definições

Tem sido largamente reconhecido que o futuro das infraestruturas das telecomunicações se tem tornado um campo bastante fértil para a utilização de aplicações baseadas em aplicações de software [8]. Decina e Trecordi [9] afirmaram que *"Os serviços das futuras redes de telecomunicações serão "povoados" por aplicações de software (agentes) capazes de assistirem um utilizador final na utilização da cada vez mais crescente informação disponível"*.

O termo *agente* tem vindo a ser adoptado genericamente para se descrever o conceito de

uma entidade desenvolvida em software, isto é, uma aplicação, que automatize determinadas tarefas consideradas como cansativas para um agente humano. Ao longo da presente tese, esse termo será utilizado para se referir a um componente de software que certos investigadores utilizam para decompor problemas complexos, denominado por *Agente Inteligente Cooperativo*, muitas vezes referido apenas como *Agente Inteligente* (*Intelligent Agent* - IA). O aparecimento destes agentes foi apoiado no conceito básico da utilização de processos de software separados, implementados por exemplo por evocação de procedimentos, processos leves - *"threads"* para tratarem de tarefas de controlo automatizado. Por este motivo, os agentes tornaram-se desde os anos noventa numa das maiores tecnologias do software. Um dos maiores domínios potenciais de aplicação para os agentes é o da indústria das telecomunicações, já que possibilitam a gestão automatizada e o controlo das mesmas.

Facilmente se percebe que os sistemas destas áreas são sistemas distribuídos e heterogéneos. Aqui não podemos falar simplesmente em agentes, mas sim em sistemas multi-agente (*Multi-Agent Systems - MAS*), definidos como um conjunto de agentes que interagem uns com os outros e com o meio que os rodeia, com o propósito de, coordenadamente, resolver um problema particular. Tal pode ser realizado através da utilização de agentes individuais que procuram uma optimização comportamental localizada (muitas vezes referidos como agentes auto-interessados), ou através de um sistema global cooperativo optimizado.

É notório que a aproximação efectuada pelos agentes tem influenciado significativamente a área das redes de telecomunicações, no sentido de se não pensar na informação como globalmente centralizada, mas sim em termos de sistemas abertos onde a distribuição geográfica e o controlo dessa informação aumentam a capacidade de processamento e a necessidade de fornecer serviços cada vez mais específicos.

Como foi referido, um agente pode ser descrito como uma aplicação de software (muitas vezes implementada como processos UNIX ou como múltiplas *threads*), capazes de manipular autonomamente (sem a actuação directa de um humano) a selecção de acções quando ocorre determinado evento. Assim, os agentes possuem diferentes habilidades e qualidades, próprias para lidarem com o seu mundo (i.e. o domínio de aplicação). Tendo sido utilizado desde há bastante tempo, na área da computação distribuída, o termo *agente* tem sido aplicado na referência a entidades específicas - *Client/Server* - na resolução de problemas nos sistemas distribuídos de processamento.

Entre os atributos das actividades dos agentes, encontramos os seguintes: *sociabilidade, autonomia, reactividade, adaptabilidade, pro-actividade, aprendizagem e grau de granularidade*, descritos como o seguinte:

- **sociabilidade** - um agente é capaz de utilizar a comunicação como base para a troca e partilha de informação com os restantes agentes do seu meio. Desta forma, os agentes operam à volta de um objectivo global;

- **autonomia** - os agentes devem operar sem a intervenção de elementos externos (outros agentes ou humanos). Eles devem possuir algum tipo de controlo sobre as suas acções e sobre os seus estados internos;

- **reactividade** - os agentes compreendem o seu ambiente e respondem atempadamente às alterações que possam ocorrer;
- **adaptabilidade** - os agentes são caracterizados pela sua flexibilidade, adaptação, e facilidade para definir os seus próprios objectivos, baseando-se nos seus interesses. Uma das suas maiores características é a sua habilidade para adquirir e processar informação sobre determinada situação;
- **pro-actividade** - os agentes devem exibir publicamente o seu comportamento, isto é, mostrar que as acções tomadas produzem alterações benéficas ao seu ambiente. Esta capacidade requer muitas vezes a previsão de situações que possam vir a ocorrer, em vez de responder simplesmente a essas alterações;
- **aprendizagem** - a comunidade de agentes pode, por si mesmo, possuir a habilidade de aprender, ou cada agente possuir um algoritmo próprio de aprendizagem. Esta aprendizagem permite muitas vezes a um agente alterar as suas futuras sequências de acção e de comportamento por forma a que possíveis erros não ocorram. Aprender é muitas vezes um factor que fornece a um agente a habilidade de demonstrar um comportamento adaptativo.
- **grau de granularidade** - os agentes podem ter vários graus de complexidade. Os mais simples são caracterizados pela falta de inteligência quando nos referimos ao seu comportamento. Estes agentes são denominados de *reactivos*. Os agentes mais complexos são denominados de *cognitivos* ou *inteligentes* e são caracterizados pela sua habilidade de conhecer o ambiente, agir sobre eles mesmos e no próprio ambiente. O comportamento neles observado é fruto da sua percepção, conhecimento e interacção.

Um dos maiores objectivos adjacentes à investigação desenvolvida na área da inteligência artificial (*Artificial Intelligence* - AI) é o do desenvolvimento de software com características semelhantes às capacidades inteligentes dos seres humanos, tais como razão, comunicação através de uma linguagem natural, e aprendizagem. Além do mais, o crescente avanço das tecnologias na implementação das redes de computadores tem sido o grande dinamizador destes estudos, onde não se fala simplesmente em inteligência artificial, mas em inteligência artificial distribuída (*Distributed Artificial Intelligence* - DAI). Aqui, a grande preocupação é a do desenvolvimento de agentes onde exista interacção na resolução de problemas.

A um nível mais elevado, a aproximação aos sistemas multi-agente é bastante intuitiva e simples: os investigadores baseiam-se na sua própria experiência para a resolução de problemas do mundo real, o que lhes dá a vantagem de, quando desenvolvem aplicações autónomas para a implementação de sistemas multi-agente, utilizarem uma metodologia natural e fiável na separação desses problemas em vários módulos. Por isso, o conceito de sistema multi-agente permite adquirir uma conceptualização requerida no controlo descentralizado, como é o caso, por exemplo, da atribuição de papéis aos agentes. A aproximação usada no desenvolvimento de agentes oferece ainda uma analogia apropriada à decomposição de problemas e à delegação de tarefas, já que permite a abstracção de determinados

problemas complexos que estão gradualmente a emergir do domínio das telecomunicações [10].

Segundo a literatura existente sobre a aproximação à resolução de problemas nos sistemas distribuídos com características acentuadamente heterogéneas, (como é o caso da realidade das redes de telecomunicações), podem utilizar-se os sistemas multi-agente para, resumidamente:

- resolver problemas de uma dimensão muito elevada para um simples agente centralizado devido a, por exemplo, limitações de recursos ou impossibilidade de recuperação de falhas ou de acontecimentos inesperados;
- permitir a redução de custos de processamento - é menos custoso, em termos de hardware, a utilização de processadores menos dispendiosos do que a utilização de um único processador com equivalente capacidade de processamento;
- permitir a interligação e interoperação de sistemas distintos, como é o caso de sistemas de apoio à decisão, dos sistemas de redes utilizando diferentes protocolos de comunicação, etc.;
- melhorar a escalabilidade, uma vez que a estrutura organizacional dos agentes pode ser alterada dinamicamente para exprimir o ambiente onde se encontram inseridos. Por exemplo, à medida que vai crescendo uma rede de comunicações, a organização dos agentes pode ser reestruturada pelos mesmos através da alteração dos seus papéis, crenças e acções;
- fornecer soluções inerentes aos problemas distribuídos;
- fornecer soluções inferidas das fontes distribuídas de informação;
- fornecer soluções onde o conhecimento é distribuído.

Adicionalmente, o paradigma dos sistemas multi-agente é inerentemente escalável devido à sua modularidade, uma vez que, para se controlar sistemas complexos é necessário:

- existência de coordenação entre os múltiplos objectivos, às vezes conflituosos, como é o caso de se ser responsável por objectivos com um nível superior de prioridade e continuar, em simultâneo, a prestar serviços a objectivos de prioridade inferior;
- gerir dados adquiridos, muitas vezes incompletos e inconsistentes;
- adoptar uma estratégia de controlo quando o ambiente (ou parte dele) sofre mutações drásticas, ou a estrutura de controlo deixa de funcionar normalmente.

Estes requisitos correspondem a muitos problemas encontrados na gestão de redes, como é o exemplo do comportamento assíncrono presente na multiplexagem estatística de células ou da tecnologia de comutação de pacotes de dados, implicando a ocorrência

de um conjunto complexo de interacções. É extremamente difícil codificar cada possível cenário onde um grande número de incertezas pode potencialmente ocorrer em simultâneo. Assim, requer-se uma técnica robusta para codificar todos esses factores que se interagem com o propósito da adaptação a alterações de estados nestes sistemas. A utilização de sistemas baseados em aplicações de agentes possui vantagens, face a sistemas de controlo centralizado, quando se pretende, por um lado, gerir muita informação e, por outro, atingir variados objectivos. Como consequência, as arquitecturas de gestão e controlo de redes de comunicações que utilizam a tecnologia dos agentes são entendidas como um desafio e um passo importante a ser tomado no futuro dos sistemas deste tipo.

## 2.3 Arquitecturas de Agentes

Sem dúvida que é bem conhecido, e aceite por toda a gente, que a construção de sistemas complexos é mais difícil do que a construção de sistemas mais pequenos. Na prática, os sistemas complexos modulares de software introduzem os seus próprios problemas, já que estamos perante relações entre esses módulos. Uma forma de se conseguir uma análise estruturada quando se constroem sistemas baseados em subsistemas que interagem entre si, é a de desenvolver uma *arquitectura* bem definida.

Uma arquitectura de agentes poderá ser analisada segundo dois graus de granularidade: primeiro, a arquitectura que define a relação entre os componentes internos constituintes de um agente, muitas vezes referida como a *arquitectura interna* de um agente; segundo, a arquitectura de um sistema multi-agente que define as relações e as interacções entre cada agente individual.

Wooldridge e Jennings [11] classificam os agentes como *reactivos, deliberativos ou híbridos.* Numa arquitectura reactiva de agentes, os agentes não possuem uma representação interna do ambiente. Todavia, são muitos os agentes classificados como reactivos apesar de não possuirem uma representação limitada do mundo. Esta representação é muitas vezes limitada a um conjunto de regras condicionantes de acções onde o agente efectua uma acção associada a um evento determinado por uma regra. Genericamente, um agente reactivo pode actuar efectuando alterações no mundo mais rapidamente do que os agentes baseados nas arquitecturas que a seguir se descrevem.

Numa arquitectura deliberativa, os agentes possuem geralmente uma representação interna mais rica do ambiente onde se encontram inseridos. Esta representação interna é constantemente alterada por forma a decidir qual das acções planeadas irão ocorrer em conformidade com os papéis definidos para os agentes. O conhecimento extra que é intrínseco a esta arquitectura pode levar a um comportamento mais flexível por parte de um agente, quando comparado com o comportamento de um agente numa arquitectura puramente reactiva. Contudo, os agentes deliberativos tendem a corrigir com mais sucesso as semelhanças entre estes dois tipos de sistemas. Um dos modelos mais bem conhecidos que pode ser visto como uma instância de uma arquitectura deliberativa é o modelo BDI (*Belief, Desire and Intension*) [12]. Finalmente, uma arquitectura híbrida é um misto das arquitecturas anteriores. Por este motivo, integra mecanismos mais rápidos de resposta

com comportamentos proactivos.

Jennings et al [13] afirmam que a importância da adopção apropriada de uma arquitectura interna dos agentes está intimamente relacionada com o nível de funcionamento do sistema: "*O facto de não existir apenas um objectivo para todo um sistema MAS requer, por um lado uma arquitectura de agentes que reflicta a dualidade de papéis de se ser um membro individual e comunitário e, por outro, um raciocínio explícito sobre o processo de coordenação*".

Uma arquitectura ao nível multi-agente deve ser vista como uma estrutura superior que os agentes necessitam para agir e muitas vezes constitui o mecanismo de coordenação, necessário para se atingir o objectivo global da arquitectura de agentes.

A figura 2.1 mostra como um agente interage com o meio envolvente, indicando a informação recebida e enviada, assim como as acções por ele efectuadas.

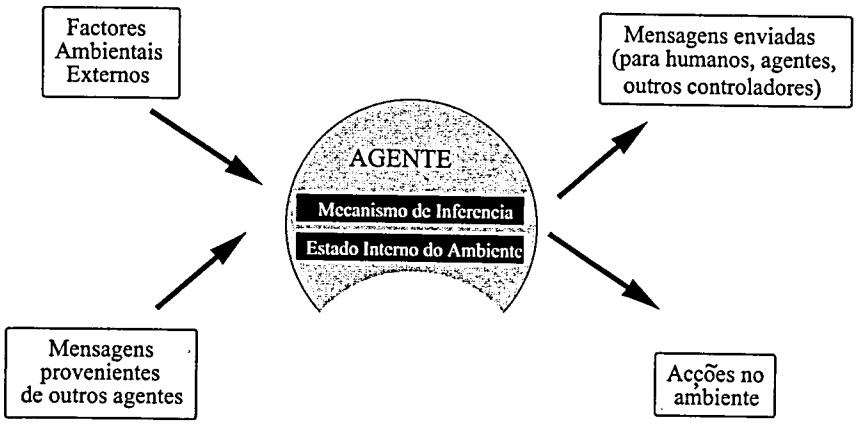



Figura 2.1: Interacção de um agente com o seu meio externo

## 2.4 Sistemas Abertos Heterogéneos Multi-Agente

O conceito chave base dos sistemas multi-agente (MAS) é a existência de uma sociedade de agentes que trocam informação. A maior parte das vezes, a resolução de problemas que pode ser conseguida através de um sistema MAS é determinada pela quantidade de cooperação e colaboração que os agentes, como entidades individuais, podem demonstrar na actividade nesse sistema, e pela própria dimensão da informação partilhada. Deve, no entanto, ser feita uma distinção entre os tipos de agentes que constituem um sistema MAS pelo facto destes possuirem características distintas devido ao seu tipo de comportamento. Assim, neste tipo de sistemas encontramos *agentes cooperativos* e *agentes auto-interessados.*

O comportamento auto-interessado é o mais apropriado quando os agentes são desenvolvidos para competir em ambientes abertos. A coordenação é muitas vezes essencial em sistemas de controlo críticos, como o caso do estado de funcionamento equilibrado de um sistema que deve ser bem conhecido (por exemplo, uma falha no controlo de uma rede). As diferenças mais relevantes nestes distintos tipos de agentes para, por exemplo, a gestão de redes de comunicações, podem ser vistas da seguinte forma:

- **Agentes cooperativos** - trabalham em conjunto por forma a resolver um problema comum, sendo um modelo apropriado para a gestão de redes, quando visto segundo a perspectiva da existência de um único fornecedor de serviços de rede (normalmente de grande dimensão).

- **Agentes auto-interessados** - a introdução de fornecedores independentes de serviços tem implicado que, sistemas de negociação automática com agentes auto-interessados (aqueles que tentam maximizar os seus próprios recursos sem se preocuparem com os recursos globais do sistema, servindo apenas os outros para compensarem em termos de, por exemplo, custos), se tenham tornado muito importantes na gestão das actuais redes de comunicações. A utilização crescente das infraestruturas standard de comunicação (onde se inclui o CORBA, o JAVA RMI, o KQML, e o FIPA) tem tornado essas interacções possíveis, tendo-se assistido ao crescimento, por esse motivo, da actividade de investigação sobre as propriedades das interacções auto-interessadas em tais sociedades de agentes.

Entre a literatura deste grande tema - "*Os Agentes e os Sistemas de Redes de Telecomunicações*" - encontramos trabalhos alusivos aos problemas de planeamento distribuído, escalonamento, reserva e atribuição de recursos, e problemas de controlo que os agentes cooperativos enfrentam. Entre as principais razões para distribuição de tarefas nestes sistemas encontramos:

- o desejo de reduzir os custos de comunicação associada à resolução de um problema;
- o desejo de melhorar o desempenho através do paralelismo de processos;
- o desejo de se obter uma reacção descentralizada; e
- o desejo de melhorar a robustez dos sistemas através da não dependência de um simples ponto de processamento.

Infelizmente, em termos práticos, o caso da distribuição da resolução de um problema entre agentes cooperativos nem sempre tem sido entendido como consistente, existindo uma necessidade de ter que se analisar os argumentos que surgem no contexto da gestão de redes de comunicações. Vejamos primeiro um exemplo simples onde a distribuição é efectiva: uma forma simplificada do controlo de admissão de acesso a uma rede fornecedora de serviços. Se o problema da política de admissão é formalizado em termos de, por exemplo, minimização de utilização e estar ainda dependente dos recursos até então utilizados, é

facilmente demonstrável que a política de atribuição de recursos a caminhos mais curtos, quando concretizável, é equivalente a aplicar um gradiente baseado em aproximações sucessivas para convergir numa melhor utilização. O mesmo se aplica a diferentes métricas utilizadas. Isto pode ser utilizado para se fornecerem políticas de atribuição de recursos a agentes reactivos localizados nos pontos de entrada das redes. Quando é efectuado um novo pedido de acesso, simplesmente é atribuído o caminho mais curto que possui uma largura de banda adequada. Os agentes distribuídos, agindo autonomamente como o descrito, interagem por forma a fornecerem um controlo reactivo que normalmente converge para uma utilização optimizada dos recursos da rede. Porém, esta solução global optimizada nem sempre é garantida, já que nem sempre se encontram soluções locais óptimas. Ao nível reactivo, quando se efectua o controlo de admissão a uma rede, existe um argumento muito forte para a distribuição, pois estamos de facto a obter as vantagens anteriormente citadas (o problema de estabelecer uma ligação com largura de banda adequada torna o problema mais complexo, mas o princípio é o mesmo). À medida que se vão utilizando mais níveis estratégicos de controlo ou de planeamento, mais necessidade se tem de utilizar a coordenação entre os agentes.

Como exemplo oposto ao anterior, vejamos o caso da gestão de uma rede onde se verifica que a distribuição de agentes cooperativos pouco contribui para a resolução de um problema. Suponha-se que é pretendido optimizar o fluxo de dados de uma rede segundo determinados modelos de procura nos pontos de entrada da mesma. Aqui o problema identificado é o da reserva de recursos que pode ser formalizado em termos de uma resolução central por parte de um agente. No entanto, o problema pode também ser formalizado em termos da distribuição de tarefas de processamento, correspondentes às acções necessárias para melhorar o fluxo de tráfego na rede a agentes autónomos cooperativos, e conseguir que os agentes possuam um comportamento convergente para a reserva e atribuição de recursos de rede globalmente optimizados. Este exemplo leva-nos a uma solução mais robusta. Se um ponto da rede, por qualquer motivo, deixa de funcionar, o processamento continua a ser efectivo. Contudo, todos os agentes distribuídos pela rede devem ter a consciência desse facto por forma a que se possam tomar medidas correctivas. Assim, torna-se necessária maior comunicação, já que eles devem estar actualizados sobre os estados da rede. Em alternativa a este excesso de comunicação, podem replicar-se todos os cálculos necessários em mais pontos nos extremos da rede, tornando-se esta solução bastante mais atractiva. A robustez da rede é também assegurada e, contrariamente ao exemplo anterior onde nem sempre se encontra uma solução optimizada, consegue-se resolver o problema em questão.

O trabalho mais recentemente desenvolvido na área dos sistemas MAS tem-se centrado quase exclusivamente na interacção dos agentes desenvolvidos para operarem em ambientes cooperativos. Por este motivo, podemos contar com os agentes para o benefício global de um sistema, tendo sempre em atenção que esse sistema não é homogéneo e podendo neles ser encontrados agentes desenvolvidos por outras pessoas. Assim, a aproximação utilizada nestes sistemas para a resolução de um dado problema no chamado *paradigma dos sistemas abertos* introduz outros aspectos, salientando-se a necessidade de se lidarem com questões de segurança, questões de se pagar recursos de processamento de outras pessoas, etc. Consequentemente, para que num sistema multi-agente os agentes possuam

um comportamento coerente e coordenado, é necessária a existência de um mecanismo que lhes permita dialogar, trocando assim as mensagens correspondentes às acções a serem efectuadas.

## 2.5 Coordenação em Sistemas Multi-Agente

A resolução de um problema efectuada pelos agentes em sistemas MAS toma a forma de uma resolução distribuída do problema (DPS - "*Distributed Problem Solving*") e envolve coordenação, negociação e comunicação. Por forma a serem resolvidos os problemas nestes sistemas, os agentes devem trocar (normalmente através de uma comunicação explícita) a informação sobre as suas actividades. Além do mais, devem ainda coordenar essas actividades e negociar entre si quando aparecem determinados conflitos por forma a resolvê-los. Estes conflitos podem variar desde simples limitações de contenção de recursos a questões mais complexas onde os agentes estão em desacordo devido a discrepâncias entre os seus domínios de conhecimento. Assim, é necessária uma coordenação para determinar a estrutura organizacional entre um grupo de agentes. Existem bastantes mecanismos que permitem esta coordenação, incluindo-se a estruturação dos sistemas de agentes em hierarquias. Desta forma, a coordenação torna-se o ponto central dos sistemas MAS: sem ela, desaparecem todos os benefícios existentes na interacção, e um grupo de agentes rapidamente degenera numa colecção de entidades com um comportamento caótico. A coordenação pode ser descrita essencialmente como um processo que garante a uma comunidade de agentes agir de uma forma coerente e harmoniosa. Como motivos para a existência de coordenação entre os agentes [14] encontramos:

- Primeiro, para prevenir o caos: numa forma geral, nenhum agente possui uma visão global de toda a comunidade a que pertence, pois é simplesmente impraticável que tal aconteça numa comunidade razoavelmente complexa. Consequentemente, os agentes apenas possuem uma visão local dos objectivos e conhecimento que podem interferir com eles.

- Segundo, para ir ao encontro de constrangimentos globais: os agentes que gerem uma rede devem responder atempadamente a certas falhas. Assim, é necessário um comportamento coordenado para que tal se consiga concretizar.

- Os agentes nos sistemas MAS possuem diversas capacidades e conhecimentos: muito analogamente aos seres humanos, onde especialistas coordenam as suas capacidades, também os agentes devem coordenar as suas actividades.

- As acções dos agentes são frequentemente interdependentes e, por isso, um agente pode ter a necessidade de esperar que outro agente complete determinadas tarefas antes de executar as suas. Obviamente que estas tarefas mutuamente dependentes precisam de ser coordenadas.

## 2.6 Gestão Centralizada ou Gestão Distribuída

No domínio das telecomunicações, a decisão de se implementar uma arquitectura de gestão de redes centralizada, hierarquizada, distribuída ou híbrida é um facto importante para a implementação efectiva, ou mesmo apropriada, de agentes de software. O rápido crescimento das redes, associado à sua complexidade, flexibilidade e diversidade, tem levantado a grande questão sobre saber até que ponto um sistema centralizado é escalável. Infelizmente, a passagem de informação de monitorização, controlo e de confirmação desde os equipamentos de comutação até às estações de gestão impõem um grande atraso no mecanismo de comunicação das camadas físicas básicas. Um sistema centralizado implica uma maior vulnerabilidade a colapsos quando um falha ocorre. Não existe uma política localizada de controlo, efectuando-se o controlo através de um sistema de gestão central. No entanto, uma gestão centralizada fornece mecanismos efectivos. Por exemplo, um número pequeno de entidades redundantes requer uma activação controlada. Tal deve-se ao facto de uma aproximação centralizada oferecer benefícios em termos do fornecimento de informação global do estado da rede e, logo, reduzir a probabilidade de ocorrência de inconsistências e conflitos. Obviamente que os benefícios oferecidos por este tipo de gestão devem ser pesados e ponderados face à necessidade de se escalabilizar o sistema. Uma desvantagem óbvia da aproximação centralizada dos sistemas é a de não permitir que os dados de controlo sejam processados onde e quando tal é necessário. Por forma a diversificar aplicações e serviços de rede, devem ser considerados alguns pontos, quer haja uma necessidade de informação global propagada até aos variados pontos da rede, tais como a necessidade de disponibilizar interfaces comuns de gestão, quer a necessidade de reduzir gastos na distribuição geográfica da capacidade de processamento.

Contudo, o controlo das redes de telecomunicações é inerentemente distribuído e os sistemas de gestão necessitam de uma capacidade para tratarem excepções ocorridas em tempo útil de funcionamento das aplicações e ser robustos sob essas circunstâncias. À medida que aumenta a complexidade dos serviços (particularmente nas redes de suporte a aplicações multi-média), maior é a quantidade de informação necessária de ser processada, e algumas falhas parciais no sistema requerem uma rápida rectificação. Por este motivo, a gestão de uma rede tem sido cada vez mais difícil de ser concretizada por mecanismos centralizados. Após a determinação destes factos, pode-se pensar no processo de gestão distribuído, através de, por exemplo:

- Distribuição de tarefas apropriadas e disponibilizadas pelos recursos computacionais;
- Melhoramento da robustez em termos de recuperação de erros e de tratamento de informação incerta;
- Melhoramento do tempo de resposta: o controlo localizado pode permitir respostas mais rápidas a eventos locais;
- Utilização de técnicas do domínio da inteligência artificial quando existe falta de conhecimento sobre como desenvolver um algoritmo de processamento óptimo, isto

é, quando não existe um procedimento bem definido ou um algoritmo para tratar certos eventos;

- Possibilitar a extensão do fornecimento de novos serviços, como por exemplo, a introdução de novos serviços complexos como é o caso da difusão de video à escolha para determinados grupos de utilizadores da rede ("*multicast video-on-demand*").

## 2.7 Gestão de Redes Baseada em Agentes

Normalmente falamos em gestão de redes quando nos referimos a um mecanismo que aje globalmente em muitas entidades e tende a fazer alterações na configuração de um sistema, (por exemplo readjudicando recursos), levando-o de um estado estável para outro. As tarefas de gestão ajem normalmente sob uma escala temporal relativamente grande. O controlo de um sistema de telecomunicações pode, resumidamente, entender-se como acções muito pequenas necessárias para manter o mesmo em condições estáveis. No entanto, este conceito tende a tornar-se um pouco confuso com o aparecimento da nova tecnologia dos agentes, onde o conhecimento localizado dos próprios agentes permite a adptação lógica, distribuída e coordenada, do controlo do equipamento de rede, como por exemplo, uma alteração muito pequena do funcionamento de um equipamento invocada de um ponto geograficamente distante.

A gestão de serviços extremo-a-extremo e a atribuição de recursos em ambientes heterogéneos, disponibilizados por múltiplas operadoras de rede, é agora uma missão crítica no contexto do actual mercado aberto de serviços de rede. Para se poder aplicar esta nova tecnologia (a dos agentes) a esta realidade, as equipas de investigação da área têm se preocupado com a integração de conceitos de controlo e de gestão nestes sistemas, aplicando ainda conceitos económicos e conceitos aplicados aos jogos de inteligência artificial na adjudicação de todos os recursos dos mesmos.

Apesar da adjudicação de recursos ser um factor muito importante a ter em atenção quando se aplicam agentes aos sistemas de telecomunicações, existem outros, entre os quais se salienta:

- Que tipo de comunicação deve um agente ter com outro? (devemos minimizar a comunicação entre eles para que uma acção decidida seja realizada mais rapidamente);
- Onde devem estar localizados os agentes? Um agente por ponto de rede? Um agente em cada ligação física? Um a representar cada ligação?
- Devem os agentes ser estáticos ou móveis?
- Que visão e conhecimento de uma rede deve possuir um agente? É benéfico que um agente tenha o conhecimento de eventos indirectamente com ele relacionados e que ocorrem localmente a outro agente?
- Que grau de dependência deve existir entre os agentes nos sistemas? Se um agente "morre" continuará o sistema robusto?

## 2.8 Os Agentes na Gestão de Redes de Área Alargada

Até agora, os assuntos introduzidos por este capítulo apenas se centraram em alguns conceitos alusivos aos agentes e em algumas particularidades dos sistemas de telecomunicações. No entanto, deve existir uma infraestrutura sobre a qual estes agentes assentam. A figura 2.2 mostra o exemplo de uma estrutura de suporte a ambiente de agentes independente da plataforma utilizada, implementável, por exemplo, através de máquinas virtuais Java (JVM - "*Java Virtual Machine*") localizadas em todos os pontos de hardware dos sistemas de rede de telecomunicações. A criação de semelhantes ambientes de processamento distribuído (DPE - "*Distributed Processing Environment*") disponibiliza uma comunicação física transparente aos agentes através de, por exemplo, Java RMI ou CORBA.

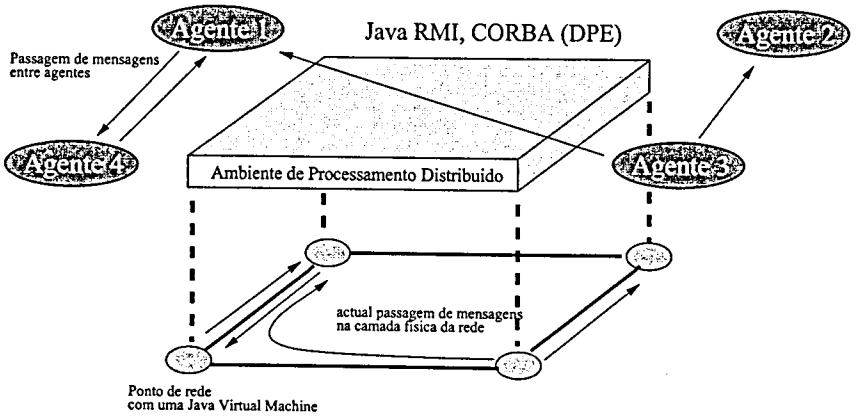



Figura 2.2: Ambiente de acção de um agente e sua relação adjacente à infraestrutura de comunicações

É também importante relembrar que a escolha apropriada de um ambiente de agentes é difícil. A opção do tipo de plataforma a utilizar deve ter em atenção que esta deve integrar um ambiente de suporte a agentes com os actuais sistemas de telecomunicações, quer em termos da sua arquitectura, quer em termos das interfaces entre eles. Assim, com o aparecimento das novas tecnologias que aproximam a indústria das telecomunicações aos novos desenvolvimentos segundo uma aproximação orientada aos agentes, cria-se uma potencial área de desenvolvimento de novos e avançados serviços de comunicação.

É evidente que o ambiente das comunicações em redes de área alargada é provavelmente uma das áreas de maior interesse na aplicabilidade da tecnologia dos agentes. Tal facto deve-se à necessidade de se fornecer capacidades de comutação mais rápida, e ao seu potencial de disponibilizar serviços mais diversificados, factores difíceis de materializar utilizando os actuais paradigmas de gestão de redes. Os serviços das redes de área alargada levam naturalmente ao aumento do próprio controlo da rede e, por conseguinte, induz uma gestão mais complexa.

O problema da atribuição de recursos é um problema de difícil resolução porque os estados do funcionamento de uma rede deste género são dinâmicos. Embora se tenham

conduzido muitos estudos, por exemplo, nas estratégias optimizadas para o controlo de admissão de ligações (CAC - "*Connection Admission Control*"), nas topologias de encaminhamento, ou na optimização da capacidade de configuração dos sistemas, existe ainda disponível pouca literatura [8] sobre os problemas associados com a reconfiguração dinâmica das topologias de rede, a qual requer uma coordenação entre os seus diferentes aspectos funcionais segundo uma perspectiva global.

Entre um dos maiores grupos de investigação nesta área encontramos a British Telecom [15] que tem conduzido um estudo exaustivo sobre a gestão do desempenho da tecnologia ATM utilizando uma aproximação orientada a agentes: os agentes são localizados nos "buffers" de comutação e fornecem uma informação de resposta ao agente da fonte de tal forma que, ligações acrescidas à rede não interferem na congestão do tráfego existente.

Outro grande grupo em actividade nesta área é a Fundação FIPA ("*Foundation for Intelligent Physical Agents*") que definiu um cenário de estudo onde é dada importância à combinação de diferentes serviços disponibilizados por múltiplas operadoras, por forma a se obter um único serviço simples extremo-a-extremo para um utilizador final [16].

## 2.9 Desafios Para os Agentes nos Sistemas de Telecomunicações

Uma das razões que levam a que a utilização da tecnologia dos agentes ainda não esteja a ser utilizada nos actuais sistemas de telecomunicações, é a falta de infraestruturas convenientes ao suporte do controlo autónomo baseado em agentes. É claro que esta situação está a ser alterada com a cada vez maior disponibilidade de plataformas e a crescente popularidade da internet. No entanto, existem outras preocupações que impedem a utilização desta tecnologia: a resolução distribuída de problemas, as técnicas de inteligência artificial para permitirem negociação entre os agentes, a atribuição e reserva de recursos distribuídos, e os próprios agentes ainda se encontram numa relativa fase de desenvolvimento. Devido a este estágio de desenvolvimento, todos aqueles que desenvolvem aplicações utilizando esta tecnologia têm tido uma preocupação maior quando se trata de obter autorização legal para a integração de sistemas, algo que é muito complexo e relacionado com os sistemas de telecomunicações actualmente existentes. Por este motivo, pensa-se que os agentes em software serão adoptados em áreas corporativas onde os custos do desenvolvimento de novas infraestruturas podem ser facilmente aceites. Após a adopção de soluções baseadas nesta tecnologia de agentes, espera-se adquirir uma vantagem competitiva que suplante os efeitos iniciais de tais custos.

Por fim, quando se opta por utilizar a tecnologia dos agentes em redes de telecomunicações, devem ser considerados, na resolução distribuída de um problema, outros factores onde se inclui a possibilidade de se tratarem "*deadlocks*", "*livelocks*", sincronismos, e atrasos na propagação da informação. Em adição a estes, devemos considerar ainda como se deve lidar com a "morte" de um agente; com o grau de granularidade que deve um agente possuir; quais os mecanismos de coordenação devem ser escolhidos para se cum-

prirem os requisitos de gestão de uma rede; qual a linguagem de comunicação os agentes deverão utilizar; e quais as arquitecturas internas dos agentes e do próprio sistema devem ser adoptadas.

## 2.10 Conclusão

No presente capítulo foi feita uma introdução à tecnologia dos agentes e conceitos com ela relacionados, nomeadamente a definição e a classificação da terminologia "agente". Tentou-se justificar o motivo da utilização de aplicações multi-agente que levaram à apresentação da noção geral de arquitecturas de agentes, e à necessidade da existência de mecanismos de coordenação. Tendo em vista os sistemas de telecomunicações, fez-se uma introdução aos sistemas abertos heterogéneos multi-agente (MAS) onde se distinguiram *agentes cooperativos* e *agentes auto-interessados*. Foram ainda levantadas algumas questões quanto à gestão de uma rede onde se tentou mostrar que tipo de gestão deve ser adoptado (centralizado ou distribuído). Como o problema central aqui levantado é o da utilização de agentes em sistemas de telecomunicações, fez-se-lhe uma introdução e identificaram-se alguns factores a considerar. Finalmente, foram apresentados alguns desafios que todos aqueles que trabalham nesta área devem enfrentar quando pretendem integrar a tecnologia dos agentes nas redes de telecomunicações.

Como conclusão, a adopção da tecnologia dos agentes na gestão e controlo de redes de telecomunicações está muito dependente da habilidade de se desenvolver uma "*sociedade de agentes*" que possa ser suficientemente flexível para responder aos requisitos futuros das referidas redes. O estado seguinte do desenvolvimento de agentes em sistemas de telecomunicações deverá precisar de mais estudos e está dependente das tecnologias de suporte à comunicação como é o caso, por exemplo, das linguagens de comunicação e das linguagens de conteúdo.

# Capítulo 3

# Sistemas de Comunicação Entre Agentes

## 3.1 Introdução

Nos últimos dez anos assistimos a uma transformação do termo **agente**, o qual "transitou" do mundo da inteligência artificial, para o nosso dia-a-dia. A acompanhar essa transição, encontramos promessas de proezas fantásticas e notáveis sobre o que os agentes poderão fazer. O termo **agente** - ao qual se designam as aplicações de software - refere-se a um paradigma aplicável ao desenvolvimento de software que acentua autonomia, adaptatibilidade e cooperação, muitas vezes atraído ao mundo dos sistemas distribuídos e heterogéneos.

Esta transformação afectou as ferramentas e metodologias utilizadas no desenvolvimento do software para agentes, tendo sido o tema persistente na evolução conceptual dos próprios agentes a sua habilidade para interagir, isto é, comunicar uns com os outros e, por conseguinte, serem capazes de enfrentar problemas colectivos que um simples agente é incapaz individualmente.

Para a maior parte da comunidade de agentes, o papel de assegurar que são capazes de comunicar levou ao aparecimento das linguagens de comunicação para agentes (ACLs). O KQML (*Knowledge Query and Manipulation Language*) foi concebido no início dos anos 90, e definiu gradualmente o conceito de uma linguagem de comunicação para agentes. Na sua fase inicial, surgiu do trabalho realizado pelo grupo de investigação KSE (Knowledge Sharing Effort), um consórcio dirigido pela maior parte dos investigadores da área da inteligência artificial (AI). Com ele, procurou chegar-se à interoperabilidade entre bases de conhecimento.

Recentemente, o KQML resumiu-se a uma colecção de actos de comunicação, como tipos de mensagens, expressos por cadeias alfanuméricas ASCII, com uma sintaxe semelhante ao LISP, transportadas por ligações TCP/IP com o objectivo de trocar conhecimento e informações entre sistemas de software, os quais são vistos como bases virtuais de conhecimento.

Neste capítulo são introduzidos alguns conceitos relacionados com as linguagens de co-

municação para agentes e é feito uma avaliação entre as duas linguagens mais utilizadas (KQML e FIPA ACL). O KQML é o veículo utilizado para serem introduzidas as noções fundamentais de uma ACL. A semântica desta linguagem tem sido a questão mais importante nas discussões sobre linguagens de comunicação, sendo, por isso, apresentada uma breve visão sobre o trabalho mais relevante efectuado nesse contexto. Como linguagem de comunicação alternativa, é apresentada o FIPA ACL, que foi desenvolvido pela Fundação para Agentes Inteligentes Físicos (*Foundation for Intelligent Physical Agents*), o primeiro esforço organizado centrado no desenvolvimento de standards na vasta área dos agentes.

É ainda feita uma comparação entre o KQML e o FIPA ACL, onde se dá uma visão sobre o assunto dominante da semântica associada a uma linguagem de comunicação. Por fim, é apresentado o KIF (Knowledge Interchange Format), uma linguagem de contéudo possível de ser utilizada por estas linguagens de comunicação.

## 3.2 Linguagens de Comunicação para Agentes - ACLs

### 3.2.1 Conceitos Básicos

As linguagens de comunicação possibilitam aos agentes a troca de informações e de conhecimento [17]. No entanto, antes do seu aparecimento, usavam-se outros métodos para mostrar uma semelhança com a troca de informação e de conhecimento entre aplicações: desde a evocação remota de procedimentos (RPC e RMI) até ao CORBA, os objectivos eram os mesmos. O que distingue as linguagens de comunicação desses esforços passados são os objectos do discurso e a complexidade da sua semântica. As linguagens de comunicação para agentes encontram-se a um nível superior, quando comparados com esses métodos, devido a duas grandes razões: primeiro porque tratam proposições, regras e acções em vez de simples objectos sem semântica associada; segundo, numa linguagem de comunicação uma mensagem descreve um estado desejado através de uma linguagem declarativa em vez de um método ou função.

Contudo, estas linguagens não cobrem toda a gama de aplicações onde poderá ocorrer troca de informações. Os agentes podem, e devem, trocar objectos mais complexos tais como objectivos e expectativas, ou mesmo partilhar experiências e estratégias de actuação a longo prazo. A um nível mais técnico, quando se utiliza uma linguagem de comunicação, os agentes trocam mensagens através da rede utilizando um protocolo de baixo nível como, por exemplo, o SMTP, o TCP/IP ou o HTTP.

A linguagem de comunicação define, por si só, o tipo de mensagens (e seu significado) que os agentes podem trocar entre si. No entanto, eles não podem ocupar-se apenas de mensagens simples. Eles possuem tarefas orientadas a conversações, isto é, sequências de mensagens partilhadas que seguem, tais como negociações e actuações.

De uma forma geral, os diferentes tipos de mensagens das linguagens de comunicação são entendidos como sendo actos de comunicação que, por seu turno, são usualmente utilizados para descrever e definir crenças, desejos e intenções (BDI - *Beliefs, Desires and Intensions*) dos próprios agentes. Este tipo de descrição ao nível intencional pode tornar-se numa forma

útil de se ver um sistema, possuindo um aspecto concreto de processameno computacional. Neste caso, pode desenvolver-se uma larga gama de agentes BDI, os quais podem possuir representações implícitas e explícitas dessas correspondentes modalidades (crenças, desejos e intenções). Essas representações são construídas através de uma descrição conceptual do conhecimento de um agente, e dos seus objectivos e compromissos, também conhecida como teoria BDI [18].

### 3.2.2 Origem das Linguagens de Comunicação

O grupo de investigação KSE [19], criado em 1990 pela agência DARPA ("*Defense Advanced Research Projects Agency*"), incluiu a participação de investigadores académicos e industriais para o desenvolvimento de tecnologias, metodologias e ferramentas de software para partilha e reutilização de conhecimento. O conceito central deste grupo era o de que a partilha de conhecimento requeria comunicação, a qual, por seu turno, requeria uma linguagem comum, tendo sido por isso o seu esforço centrado na definição de uma linguagem de comunicação comum.

No modelo KSE, os sistemas de software são bases virtuais de conhecimento que trocam proposições utilizando uma linguagem que expressa variadas e complexas atitudes. Estas atitudes não são mais do que relações tripartidas entre um agente, uma proposição com transporte de conteúdo (por exemplo, "*Está a chover*"), e um conjunto finito de atitutes proposicionais que um agente possa ter no que lhes diz respeito (por exemplo, acreditar, ter mêdo, esperar, reclamar, admirar, etc.). Como exemplo de uma atitude proposicional, podemos considerar a seguinte: **<a, medo, chover ($t_{agora}$)>**.

### 3.2.3 KQML

Entre as linguagens de comunicação para agentes encontramos o KQML [20] [6]. O KQML é uma linguagem de comunicação de alto nível, orientada a mensagens e, ao mesmo tempo, um protocolo para a troca de informação independentemente da sintaxe de conteúdo, e de uma ontologia aplicável, sendo por isso necessário garantir que a semântica do conteúdo de uma mensagem seja preservada entre aplicações. Por outras palavras, um mesmo conceito, objecto ou entidade devem possuir um significado uniforme, mesmo que aplicações diferentes utilizem nomes distintos para os referenciar. Desta forma, pode dizer-se que uma ontologia é uma conceptualização particular de um conjunto de objectos e uma relação entre eles. Assim, uma ontologia consiste em termos, sua definição e axiomas relacionados [21], organizados segundo uma taxonomia. Por isso, o KQML é independente do mecanismo de transporte (TCP/IP, SMTP, HTTP, etc.) e da ontologia assumida pelo conteúdo.

#### 3.2.3.1 Organização

Conceptualmente, podem identificar-se três níveis ou camadas numa mensagem KQML: conteúdo, comunicação e transporte (figura 3.1). O nível do conteúdo é reponsável por levar


Figura 3.1: O KQML visto segundo três níveis

o conteúdo da mensagem, representado numa linguagem própria de determinada aplicação. Aqui, o KQML pode ter uma qualquer linguagem de representação, onde se podem incluir as linguagens expressas segundo cadeias de caractares alfanuméricos (strings ASCII), e as linguagens que utilizam notação binária. Cada implementação KQML ignora a parte do conteúdo da mensagem, exceptuando-se aquela que determina o seu fim.

O nível de comunicação codifica um conjunto de características da mensagem que descrevem os parâmetros de comunicação a baixo nível, tais como a identificação do agente emissor e agente recptor, e o identificador único associado à comunicação.

O nível de transporte codifica a mensagem que uma aplicação (agente) pretende enviar a outra. É a este nível que se determina o tipo de interacções que um agente pode ter. A sua função primária é a de identificar o protocolo de rede que irá entregar a mensagem, e fornecer actos de comunicação ou performativas [1] que o emissor anexa ao conteúdo. Tais actos de comunicação determinam quando uma mensagem é assertiva, interrogativa, um comando, ou um conjunto de outras performativas conhecidas. Por tudo o que se referenciou a este nível, diz-se que ele é o núcleo do KQML, daí se encontar situado na base dos outros níveis de uma mensagem KQML (figura 3.1).

Uma vez que o conteúdo é opaco ao KQML, o nível de transporte inclui características opcionais que descrevem a linguagem de conteúdo e a ontologia utilizada. Estas características permitem que implementações em KQML analisem, encaminhem e entreguem mensagens cujo conteúdo se encontra inacessível.

#### 3.2.3.2 Sintaxe e Performativas

A sintaxe do KQML baseia-se em expressões muito semelhantes ao LISP, isto é, listas de "strings" balizadas por parêntisis - "()". O elemento inicial da lista é o identificador da performativa. Os elementos seguintes são os seus argumentos traduzidos em pares de palavras chave ou de valores. Devido ao facto da linguagem ser relativamente simples, a sintaxe actual não é muito significativa e pode ser alterada, se necessário, no futuro. Na tabela 3.4 encontramos um sumário das palavras reservadas KQML e a associação do respectivo significado.

[1] Entende-se por performativas todos os tipos primitivos de mensagens adoptadas pelo KQML

| Palavra reservada | Significado |
|---|---|
| performative | *acto de comunicação desejado* |
| :sender | *o agente emissor da performativa* |
| :receiver | *o agente receptor da performativa* |
| :from | *a origem da performativa no conteúdo, quando se utiliza uma performativa forward* |
| :to | *o destino final da performativa no conteúdo, quando se utiliza uma performativa forward* |
| :in-reply-to | *identificação esperada numa resposta a uma mensagem anterior* |
| :reply-with | *identificação esperada para uma resposta à mensagem actual* |
| :language | *o nome que representa a linguagem do conteúdo* |
| :ontology | *o nome da ontologia assumida para o conteúdo* |
| :content | *a informação sobre a qual a performativa expressa uma atitude* |

Tabela 3.1: Sumário das palavras reservadas KQML e seu significado

A sintaxe, relevando as raízes iniciais da sua implementação (que foi feita em LISP), tornou-se bastante flexível. Vejamos o exemplo da tabela 3.2, onde uma mensagem de um agente "**A**" representa uma interrogação sobre um preço das acções da Sun Microsystems. Nesta mensagem a performativa é **ask-one**, o conteúdo "**:content (PRICE SUN ?price)**", e a ontologia assumida é identificada pela expressão "**:ontology NYSE-TICKS**".

```
(ask-one
     :sender      A
     :receiver    stock-server
     :reply-with  sun-stock
     :language    LPROLOG
     :ontology    NYSE-TICKS
     :content     (PRICE SUN ?price))
```

Tabela 3.2: Exemplo de uma mensagem KQML: pedido de informação sobre o preço de acções da SUN Microsystems, efectuado por um agente A

O receptor da performativa é uma aplicação ou agente servidor, identificado como **stock-server** e a interrogação foi escrita numa linguagem - **LPROLOG**. O valor da

palavra chave **:content** identifica o nível de conteúdo de uma mensagem KQML, e o nome da performativa e o conjunto de características com as palavras reservadas **:language** e **:ontology** formam o nível de transporte.

Como resposta a essa performativa, o agente **stock-server** poderá enviar uma mensagem ao agente **A**, respondendo-lhe à interrogativa. Como resposta, atente-se à tabela 3.3 que pretende exemplificar uma tal possível performativa.

```
(tell
     :sender       stock-server
     :receiver     A
     :in-reply-to  sun-stock
     :language     LPROLOG
     :ontology     NYSE-TICKS
     :content      (PRICE SUN 16))
```

Tabela 3.3: Exemplo de uma mensagem KQML: resposta ao pedido de informação sobre o preço de acções da SUN Microsystems

Apesar do KQML possuir um conjunto predefinido de performativas (ver tabela 3.4), elas podem não ser todas necessárias. Dependendo do âmbito da utilização dos agentes, pode optar-se por utilizar algumas das performativas, desde que os agentes acordem com a sua interpretação e protocolo. Desta forma é possível fazer uma extensão às performativas KQML. Contudo, uma implementação que opte por utilizar uma das performativas reservadas, deve ser efectuada através de formas standard.

Um dos critérios utilizados no desenvolvimento do KQML foi o de produzir uma linguagem que pudesse suportar uma grande variedade de arquitecturas para agentes. Assim, introduziu-se um pequeno número de performativas que os agentes podem utilizar para descrever as suas crenças, desejos, intenções e, ainda, uma classe especial de agentes, chamados de **agentes facilitadores**.

Um agente facilitador é um agente que implementa variados serviços úteis de comunicação, tais como manter o registo de nomes, seguimento de mensagens aos respectivos serviços, encaminhamento de mensagens baseando-se no seu conteúdo, comparação de informação entre agentes emissores e agentes receptores. Para além destes, um agente **facilitador** fornece também serviços mediáticos e serviços de tradução de endereços.

Entre o conjunto de performativas KQML, encontramos 36 (segundo a nova especificação [6]) que se podem dividir em três categorias: performativas de **discurso**, performativas de **intervenção e de mecanismo** e performativas de **facilitação e de rede**. Os dois primeiros tipos de performativas são utilizados directamente por agentes regulares para comunicação directa e o terceiro tipo aplica-se à comunicação entre os agentes facilitadores e os agentes regulares.

- **Performativas de discurso**
  Nesta categoria encontram-se as performativas que podem ser consideradas o mais próximo possível dos actos de comunicação. Por este motivo, são utilizadas no contexto da troca de informação e de conhecimento do tipo discursivo entre dois agentes.

- **Performativas de intervenção e de mecanismo**
  Estas são as performativas utilizadas no decurso normal de uma conversação. Por exemplo, vamos supôr que um agente **A** envia uma mensagem, iniciando assim uma conversação, e um agente **B** reage conforme tenha, ou não, uma resposta a ela. As performativas desta categoria podem, por um lado, terminar prematuramente uma conversação (*error*, *sorry*) e, por outro lado, sobrepôr-se ao protocolo utilizado (*standby*, *ready*, *next*, *rest* e *discard*).

- **Performativas de facilitação e de rede**
  As performativas desta categoria não são consideradas, no sentido directo, como actos de comunicação. Elas são funções primárias que permitem aos agentes procurarem outros que possam responder às suas interrogações. Apesar dos agentes regulares poderem optar por processar estas performativas, não é muito aconselhável que tal aconteça, uma vez que as performativas de facilitação contam com as performativas de advertimento, e apenas os agentes facilitadores têm o poder de as utilizar em toda uma comunidade de agentes.

  Tirando as performativas *register*, *unregister* e *transport-address*, utilizadas para resolver problemas de nomes, todas as outras são muito importantes para implementar comunicações complexas, designadas de actos sociais entre grupos de agentes e não podem ser associadas a um qualquer acto particular de comunicação. Tal facto não se torna uma desvantagem, já que são decisivas para simplificar a interacção entre um grande grupo de agentes onde todos são conhecidos. Neste sentido, estas performativas podem ser consideradas como a base para o desenvolvimento de comunidades de agentes.

Para sumariar o exposto nesta secção, apresenta-se a tabela 3.4 onde se encontram representadas todas as performativas KQML segundo a última especificacão (KQML97), separadas por tipo de categoria.

| Categoria | Performativa | Significado |
|---|---|---|
| Discursiva | ask-if | o agente emissor pretende saber se o contéudo da mensagem existe na base virtual de conhecimento do receptor |
| | ask-all | o agente emissor pretende conhecer todas as instâncias do contéudo da mensagem existentes na base virtual de conhecimento do receptor |

| | |
|---|---|
| ask-one | o agente emissor pretende conhecer uma instância do contéudo da mensagem existente na base virtual de conhecimento do receptor |
| stream-all | resposta múltipla semelhante a ask-all |
| eos | finalização do pedido de resposta múltipla |
| tell | indicação ao agente receptor que o conteúdo da mensagem existe na base virtual de conhecimento do agente emissor |
| untell | o conteúdo da mensgem não existe na base virtual de conhecimento de um agente |
| deny | a negação do conteúdo existe na base virtual de conhecimento de um agente |
| insert | o agente emissor pede para o agente receptor inserir na sua base de conhecimento o conteúdo da mensagem |
| uninsert | o agente emissor pede para o agente receptor tirar da sua base de conhecimento o conteúdo da mensagem |
| delete-one | o agente emissor pede para o agente receptor apagar da sua base de conhecimento uma proposição idêntica ao conteúdo da mensagem |
| delete-all | o agente emissor pede para o agente receptor apagar da sua base de conhecimento todas as proposições idênticas ao conteúdo da mensagem |

| | | |
|---|---|---|
| | undelete | o agente emissor pretende desfazer um acto anterior de apagar um conteúdo de mensagem de uma base virtual de conhecimento efectuado pelo agente receptor |
| | achieve | o agente emissor pretende que o agente receptor concretize uma acção correspondente com a veracidade do conteúdo da mensagem |
| | unachieve | acto de correspondência inversa ao do achieve |
| | advertise | a agente emissor pretende que o agente receptor saiba que ele consegue processar o conteúdo da mensagem |
| | undavertise | cancelamento de um acto achieve anterior |
| | subscribe | o agente emissor pretende receber do agente receptor actualizações do conteúdo da mensagem |
| Intervenção | error | o agente emissor considera que o conteúdo da mensagem anterior, recebida do agente receptor desta, se encontra mal indicado |
| e | sorry | o agente emissor entende o conteúdo da mensagem anterior mas não o consegue processar |
| mecanismo | standby | o agente emissor indica ao receptor que se encontra num estado de espera e pronto para processar o conteúdo da mensagem |
| | ready | o agente emissor indica ao receptor que se encontra pronto para responder a uma mensagem prévia |

| | | |
|---|---|---|
| | next | o agente emissor pretende receber a resposta seguinte a uma mensagem anteriormente enviada ao receptor |
| | rest | o agente emissor pretende receber as respostas restantes a uma mensagem enviada anteriormente ao receptor |
| | discard | o agente emissor não quer mais as respostas restantes a uma mensagem enviada anteriormente ao receptor |
| Facilitação | register | o agente emissor anuncia ao agente receptor a sua presença, indicando-lhe o seu nome simbólico |
| e de rede | unregister | o agente emissor pretende efectuar o correspondente acto inverso de register |
| | transport-address | o agente emissor associa o seu nome simbólico a um novo protocolo de transporte de mensagens |
| | forward | o agente emissor pretende fazer seguir uma mensagem recebida até ao agente receptor |
| | broadcast | o agente emissor pretende enviar o conteúdo de uma mensagem a todos os agentes que conhece |
| | broker-one | o agente emissor pretende descobrir uma resposta a uma performativa, que pode ser dada por um outro agente |
| | broker-all | o agente emissor pretende descobrir todas as respostas a uma performativa, que podem ser dadas por outros agentes |

| | | |
|---|---|---|
| | recommend-one | o agente emissor pretende conhecer um agente que possa responder a uma performativa |
| | recommend-all | o agente emissor pretende conhecer todos os agentes que possam responder a uma performativa |
| | recruit-one | o agente emissor pretende que lhe seja indicado um agente capaz de responder a uma performativa |
| | recruit-all | o agente emissor pretende que lhe seja indicado todos os agentes capazes de responder a uma performativa |

Tabela 3.4: Performativas reservadas KQML organizadas segundo a sua categoria

## 3.2.4 Fundação para os Agentes Inteligentes Físicos - FIPA

### 3.2.4.1 Introdução

A fundação para os agentes físicos inteligentes é uma associação sem fins lucrativos cujo propósito é promover o sucesso emergente das aplicações baseadas nos agentes, serviços e equipamentos. O objectivo principal do grupo FIPA é disponibilizar e criar especificações que maximizem a interoperabilidade entre os sistemas baseados em agentes. Em sequência destes objectivos, o FIPA tornou-se numa organização de standards na área do software para agentes.

A organização incluiu o termo físico para cobrir também os agentes baseados em hardware, os pertencentes à família dos robots. Este grupo mantem-se activo através de uma colaboração internacional de um conjunto de organizações membro (Universidades e Empresas), das quais se podem salientar: Alcatel, British Telecom, France Telecom, Deutsche Telecom, Hitachi, NEC, Nortel, Siemens.

A especificação mais actualizada é a FIPA97, disponível no site do grupo [22]. O grupo atribui tarefas especiais aos seus membros, responsáveis por produzir, manter e actualizar as especificações dessas tarefas.

No que diz respeito à presente Tese, referir-se-á a uma especificação concreta - FIPA-ACL, especificação essa referente a uma linguagem de comunicação para agentes. Para além desta, existem outras especificações de serviços, tais como a de facilitação, a de registo, a de plataformas para agentes e a de integração de agentes em aplicações de software, que,

no seu todo, constituem a espinha dorsal das especificações FIPA.

#### 3.2.4.2 FIPA ACL

A linguagem de comunicação para agentes FIPA-ACL, tal como o KQML, é baseada na teoria dos actos de comunicação. Segundo esta teoria, as mensagens são acções ou actos de comunicação, uma vez que pretendem efectuar alguma acção em virtude do que foi enviado. A especificação FIPA-ACL consiste num conjunto de mensagens tipadas e na descrição dos factos práticos que delas se devem extrair, isto é, os efeitos das atitudes mentais dos agentes emissores e receptores das mesmas.

A especificação descreve cada acto de comunicação através de uma forma narrativa e de uma semântica formal baseada em lógica modal. Para além destas, fornece ainda a descrição normativa de um conjunto de protocolos de alto nível de interacção onde se incluem requisições de acções, contratação de redes, e uma série de acções de leiloamento.

Basicamente, esta especificação descreve o modelo de referência para uma plataforma de agentes (figura 3.2) onde se identificam os papéis de alguns agentes chave necessários para a gestão da plataforma, e especifica a linguagem de conteúdo e ontologia utilizadas pelos agentes.

Foram identificados três papéis obrigatórios para uma plataforma de agentes, atribuídos a três tipos de agentes: o agente AMS, o agente ACC e o agente DF. O agente de gestão do sistema - AMS (*Agent Management System*) - é responsável por efectuar um controlo e supervisão aos acessos e à actividade dos agentes numa arquitectura, sendo, por isso, responsável pela autenticação dos agentes residentes e pelo controlo de registos. O agente do canal de comunicações - ACC (*Agent Communication Channel*) - é responsável por fornecer o caminho para o contacto básico entre os agentes dentro e fora da arquitectura, sendo por esse motivo o método por defeito de comunicação que fornece um serviço normal e fiável de entrega de mensagens. Finalmente, o agente de facilidades de directório - DF *Directory Facilitator* - fornece o serviço de "páginas amarelas" à arquitectura.

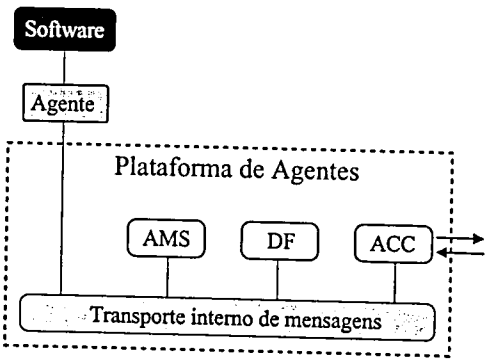



Figura 3.2: Modelo de referência para plataformas de agentes FIPA

Quanto à tecnologia actual para a implementação de uma arquitectura baseada nesta especificação, não foi feita qualquer restrição podendo utilizar-se para a troca de mensa-

gens plataformas baseadas em, por exemplo, e-mail, CORBA e aplicações suportadas por processos leves Java (Java multi-thread).

Para além destas especificações, é também definida uma linguagem de comunicação para agentes, FIPA-ACL, baseada na troca de mensagens que os agentes utilizam para comunicar. A especificação FIPA-ACL define uma linguagem standard de mensagens, especificando a codificação, a semântica e os factos a elas ligados, não sendo, no entanto, definido especificamente o mecanismo para o transporte interno das mensagens, já que os agentes podem situar-se em plataformas diferentes e utilizarem diferentes tecnologias de rede. Neste sentido, o grupo FIPA especificou que as mensagens transportadas entre plataformas devem ser codificadas num forma textual. Assim, assume-se que um agente deva possuir algum meio de transmissão dessa forma textual.

A linguagem de comunicação para agentes FIPA-ACL é muito similar ao KQML, sendo a sua sintaxe idêntica, exceptuando-se a nomenclatura de algumas performativas primitivas. Desta forma, é mantida a aproximação utilizada pelo KQML no que se refere à separação da linguagem exterior da linguagem interior: a linguagem exterior define o significado da intenção da mensagem; a linguagem interior ou de conteúdo indica a expressão aplicável ao que o interlocutor acredita, deseja ou tenciona.

O KQML tem vindo a ser muito criticado na utilização do termo *performativa* para se referir a actos primitivos de comunicação. Com o FIPA-ACL, os actos primitivos de comunicação são designados simplesmente por *actos de comunicação*. Apesar da diferença na nomeação desses actos, as performativas KQML e os actos de comunicação FIPA são do mesmo género.

| **Palavra reservada** | **Significado** |
|---|---|
| Communicative act type | *identifica o acto primitivo da comunicação utilizado pelos agentes* |
| :sender | *agente emissor do acto de comunicação* |
| :receiver | *agente receptor do acto de comunicação* |
| :content | *o próprio conteúdo da mensagem pretendida ser trocada* |
| :reply-with | *identificação esperada para uma resposta à mensagem actual* |
| :in-reply-to | *identificação esperada para uma resposta à mensagem anterior* |
| :envelope | *informação adicional útil enviada pelo serviço de transporte, tal como a hora de envio e recepção da mensagem* |
| :language | *identifica a linguagem de conteúdo utilizada* |
| :ontology | *indica em que termos (ontologias) o conteúdo da mensagem possui algum significado* |
| :reply-by | *indica a que horas ou data o agente emissor deseja receber uma resposta* |
| :protocol | *introduz um indentificador que mostre o protocolo utilizado pelo agente emissor* |

| | |
|---|---|
| :conversation-id | *expressão utilizada para identificar uma sequência de actos de comunicação em trânsito que, no seu todo, constituem uma conversação* |

Tabela 3.5: Parâmetros pré-definidos das mensagens FIPA

Atente-se à tabela 3.6, a seguir apresentada, onde se encontra patente um cenário de comunicação onde um agente **i** pede a uma agente **j** para lhe indicar o actual primeiro ministro da Inglaterra.

```
(request
     :sender    i
     :receiver  j
     :content   (inform-ref
                   :sender    j
                   :receiver  i
                   :content   (iota ?x (UKPrimeMinister ?x))
                   :ontology  world-politics
                   :language  sl
                )
     :reply-with query0
     :language   sl)


(inform
     :sender    j
     :receiver  i
     :content   (= (iota ?x (UKPrimeMinister ?x))
                   ''Tony Blair'')
     :language   sl
     :ontology  world-politics
     :in-reply-to query0)
```

Tabela 3.6: Exemplo de acto de comunicação FIPA

Como se pode verificar, os actos de comunicação que constituem este cenário são muito

semelhantes aos do KQML, exceptuando-se o nome das performativas, onde, em vez de se utilizar uma performativa *ask-if*, se utiliza um acto de comunicação *request* e, em vez de *tell* é utilizado *inform*. De resto, são de igual modo especificadas a linguagem de conteúdo e ontologia utilizadas. Informação detalhada sobre a semântica e a sintaxe adoptada pelo FIPA-ACL pode ser encontrada em [23]

Tal como foi anteriormente referido, de forma análoga ao KQML, o FIPA-ACL possui um conjunto de palavras reservadas para designar os actos primitivos de comunicação que podem ser utilizadas na interoperabilidade de agentes, fornecendo-lhes assim a capacidade de expressarem os seus desejos, crenças e intenções. Na tabela 3.7 encontram-se todos os actos de comunicação especificados pelo FIPA e respectivo significado.

| **Acto de comunicação** | **Significado** |
|---|---|
| accept-proposal | *acto de aceitar uma proposta submetida anteriormente* |
| agree | *acto de concordar numa possível forma de actuar* |
| cancel | *acto de cancelar alguma acção pedida anteriormente* |
| cfp | *acto de pedir propostas para se efectuar determinada acção* |
| confirm | *o emissor informa o receptor que determinada proposição é verdadeira* |
| disconfirm | *o emissor informa o receptor que determinada proposição é falsa* |
| failure | *acto de informar outro agente que foi tentado efectuar uma acção, porém sem êxito* |
| inform | *o emissor informa o receptor que uma dada proposição é verdadeira* |
| inform-if | *acção não atómica para o agente da acção, correspondente a informar se o conteúdo é ou não uma proposicção verdadeira* |
| inform-ref | *acção não atómica para o emissor no sentido de informar o receptor que objecto corresponde a um descritor (por exemplo, um nome)* |
| not-understood | *o emissor do acto informa o receptor que não entendeu a acção pretendida* |
| propose | *acto de submeter uma proposta para concretizar determinada acção* |
| query-if | *acto de perguntar a outro agente se uma proposição é ou não verdadeira* |
| query-ref | *acto de perguntar a outro agente por um objecto referente a uma expressão* |
| refuse | *acto de recusar efectuar uma acção e explicar porquê* |
| reject-proposal | *acto de rejeitar uma proposta durante uma negociação* |
| request | *o emissor pede ao receptor para efectuar uma acção* |

| | |
|---|---|
| request-when | *o emissor pede ao receptor para efectuar uma acção quando uma dada proposição for verdadeira* |
| request-whenever | *o emissor pede ao receptor para efectuar uma acção, assim e sempre que uma dada proposição for verdadeira* |
| subscrive | *Acto de requerer uma actualização constante de um objecto identificado* |

Tabela 3.7: Parâmetros pré-definidos de mensagens FIPA97

### 3.2.5 Comparação Entre as Linguagens de Comunicação KQML e FIPA-ACL

As linguagens de comunicação para agentes KQML e FIPA-ACL são muito idênticas no que diz respeito aos seus conceitos básicos e aos princípios que observam. Estas duas linguagens diferem primariamente no detalhe da sua estrutura semântica. Num sentido, esta diferença é substancial: devido às diferentes semânticas utilizadas, torna-se impossível arranjar formas de mapeamento exacto, ou de transformações entre as performativas KQML e os actos de comunicação correspondentes FIPA, ou vice-versa. Por outro lado, as diferenças inevitáveis podem ter uma importância significativa para muitos programadores se os agentes não forem agentes BDI.

Ambas as linguagens assumem um não-compromisso básico com uma determinada linguagem de conteúdo. No entanto, no caso do FIPA-ACL, um agente deve possuir algum conhecimento dessa linguagem de conteúdo. Como foi referido, as duas linguagens possuem uma sintaxe similar, isto é, as mensagens KQML e as mensagens FIPA são sintaticamente semelhantes, excepto no que se refere às designações das primitivas de comunicação. Este foi um importante atributo conferido à linguagem FIPA-ACL. O grupo FIPA alterou a sintaxe original da linguagem (semelhante ao Prolog) para que, na transição de sistemas baseados no KQML para sistemas FIPA, o processo fosse facilitado. O procedimento de tornar um sistema de comunicação disponível envolve a disponibilização de código que analise gramatical ou logicamente as mensagens recebidas, as prepare para transporte e as encaminhe através da rede, utilizando um protocolo de comunicação de baixo nível. Esta estrutura deve ser a mesma, quer se opte por uma ou por outra linguagem de comunicação.

No entanto, estes conceitos não se aplicam à semântica das duas linguagens. Seguindo a semântica do KQML descrita em [24], pode constatar-se que as duas linguagens diferem sintaticamente ao nível do que constitui a sua descrição semântica: pré-condições, pós-condicções e condições finais para o KQML; e a praticabilidade das pré-condições e o efeito racional para o caso do FIPA-ACL.

Para além destas diferenças, encontramos outra ao nível da escolha e da definição das modalidades que elas utilizam, isto é, a linguagem utilizada para descrever os estados dos

agentes. Apesar de se poder aproximar as primitivas KQML numa estrutura FIPA, e vice-versa, um tradução completa e precisa não é geralmente possível.

Outra diferença entre estas duas linguagens de comunicação encontra-se nas suas primitivas de facilidades e de tratamento de registo dos agentes. Estas primitivas cobrem uma gama importante de questões pragmáticas, tais como o registo, a actualização de informação de registos e a localização de agentes que possam processar certos pedidos. No KQML, estas tarefas são associadas a primitivas que a linguagem trata como objectos de primeira ordem. O FIPA, supostamente desenvolvido para ser uma linguagem de comunicação pura, não considera estas tarefas como actos de comunicação. Em vez disso, trata-os como pedidos para acção e define uma série de acções reservadas que envolvem tarefas de registo e de ciclo de vida dos agentes. Com esta aproximação, as acções reservadas não possuem uma definição formal de especificações e da semântica da linguagem, sendo esta definida em termos descritivos de uma linguagem natural. Além do mais, o FIPA-ACL, não disponibiliza actualmente primitivas de facilidades como é o caso do KQML. Muitos utilizadores das linguagens de comunicação para agentes têm expressado o desejo de o FIPA-ACL incluir primitivas deste género, por se terem habituado às do KQML -*broker, recommend,* e *recruit.* Tais desejos servem para chamar à atenção de que, para se ser prático, uma linguagem de comunicação requer um misto de aspectos teóricos e de aspectos pragmáticos.

O aparecimento do FIPA-ACL pode tornar-se numa dor de cabeça para aqueles que têm que decidir por si qual das duas linguagens de comunicação irão utilizar na implementação de uma arquitectura de agentes. Segundo [25] qualquer sistema que utilize uma destas linguagens deve fornecer as seguintes ferramentas:

1. um conjunto de APIs que facilitem a composição, o envio e a recepção de mensagens;

2. uma infraestrutura de serviços que assistam os agentes na nomeação, registo e serviços básicos de facilidades (encontrar outros agentes);

3. código para cada tipo de mensagens reservadas (performativas ou actos de comunicação) que efectue as acções determinadas pela semântica de uma aplicação em particular, dependendo esse código da linguagem utilizada pela aplicação, do seu domínio e do pormenor do sistema de agentes.

Idealmente, um programador deve apenas fornecer o item 3. Os itens 1 e 2 devem ser componentes reutilizáveis que um programador integra no código de uma aplicação. Actualmente, o programador não integra o apresentado no item 2: é aconselhável a sua existência como um serviço contínuo disponível para qualquer novo agente.

Teoricamente, a similaridade dos pressupostos básicos e das sintaxes existentes entre as linguagens de comunicação significa que apenas o item 3 deve sofrer uma alteração dependendo da escolha da linguagem. Mesmo assim, apesar do desânimo na definição semântica destas linguagens, pode prevalescer uma compreensão intuitiva das primitivas sobre uma

definição concisa da semântica da linguagem. Assim, exceptuando-se os agentes que seguem a teoria BDI ("*belief, desire, intension*"), a decisão na escolha de uma linguagem deve ser baseada em questões pragmáticas.

## 3.3 Linguagens de Conteúdo

As linguagens de conteúdo fornecem os aspectos sintácticos da representação da troca de informação e de conhecimento entre os agentes. Assim, uma linguagem de contéudo é especificada para ser um meio de transporte que exprimia conhecimento sobre algo. Neste sentido, elas devem ser o mais abertas possível para permitir a representação do conhecimento de um conjunto de ontologias. Com isto pretende-se dizer que uma linguagem deve aceitar novos vocábulos (palavras, termos e frases), requeridas por novas ontologias. Por forma a que um agente possa comunicar numa dada área de aplicação (ontologia), ele deve utilizar consistentemente palavras para se referir a objectos, funções e a relações que tenham significado nessa área. Uma forma para se desenvolver esta consistência é a de se criar um dicionário aberto de palavras apropriadas a essas áreas de aplicação comuns [26].

Sob o ponto de vista da teoria dos actos de comunicação, uma linguagem de conteúdo serve para expressar o objectivo de uma elocução, sendo por isso utilizado para escolher um acto de comunicação apropriado, isto é, uma performativa KQML.

### 3.3.1 KIF

A linguagem KIF (*Knowledge Interchange Format*) é uma linguagem orientada à computação para a troca de conhecimento entre aplicações díspares, isto é, agentes desenvolvidos por diferentes programadores, utilizando distintas linguagens de programação. Foi desenvolvida especialmente como uma linguagem de conteúdo para ser utilizada por linguagens de comunicação para agentes, como é o caso do KQML. Quando um agente necessita comunicar com outro, codifica a sua estrutura interna de dados em termos do KIF [7].

A especificação do KIF é muito pormenorizada existindo, no entanto, pormenores essenciais e pormenores arbitrários, sendo obrigatórias as seguintes características para a definição do KIF:

- A linguagem deve possuir uma semântica declarativa, isto é, o significado das expressões na representação de um conhecimento deve ser entendido sem ter de se recorrer a um interpretador que manipule tais expressões;
- A linguagem deve ser logicamente compreendida, o que permite uma expressão referente a uma qualquer proposição de primeira ordem gramatical;
- A linguagem deve fornecer uma representação do conhecimento, permitindo que todas as decisões que possam ser tomadas sobre essa representação sejam explícitas e permitam a introdução de novas decisões, sem que a linguagem sofra alterações.

#### 3.3.1.1 Vocabulário

O vocabulário básico da linguagem inclui variáveis, constantes, operadores, objectos, funções e relações, distinguindo-se as variáveis individuais utilizando o símbolo "?" no início do respectivo caracter. Para se distinguir uma sequência de variáveis, recorre-se à utilização no início dela do símbolo "@". Para além destes, existe ainda um conjunto de operadores, sendo as restantes palavras constantes [27].

#### 3.3.1.2 Visão Geral da Linguagem

A linguagem de conteúdo KIF inclui tanto uma especificação da sintaxe da linguagem como a sua semântica. Em primeiro lugar, o KIF permite expressar simples dados [3]. Por exemplo, a proposição da tabela 3.8, atribui ao agente *viriato* a crença de que na lua existe algum material constituído por granito.

```
(believes viriato ' (material lua granito))
```

Tabela 3.8: Dados Simples

No entanto, pode-se expressar informação mais complicada, como é o caso da inclusão de operadores lógicos tais como negação, quantificação, disjunção, etc.. Por exemplo, na tabela 3.9, pretende-se saber a ordem de grandeza, isto é, quantificar se um número $x$ é maior ou menor que um número $y$. Assim, a expressão $x >= y$ toma o valor lógico verdade se $x = y$ ou se $x > y$.

```
(defrelation >= (?x ?y) :=
(or (= ?y ?x) (> ?y ?x)))
```

Tabela 3.9: Termos mais complexos

Para além desta codificação efectuada pelo KIF, pode ainda codificar-se o conhecimento de um agente utilizando os operadores representados pelos símbolos "/" e "," e associar-lhes um vocabulário. Por exemplo, a expressão da tabela 3.10 indica que o agente *Domain_manager1* [28] está interessado em receber informação de um grupo do sistema SNMP de todas as máquinas do domínio administrativo. O conjunto do grupo do sistema distribuído constituído pelas máquinas é visto, neste exemplo, como uma relação KIF. Assim, cabe ao agente responder a esta proposição por forma a saber que a relação existente é virtual.

```
(interested Domain_manager1 '(system , ?x, ?y, ?z))
```

Tabela 3.10: Codificação de conhecimento

Informação mais detalhada sobre o KIF pode ser encontrada em [7], [27] ou em *http://agents.umbc.edu/aw/Topics/Communicative_Agents/KIF/index.shtml.*

Para concluir, o KIF oferece ainda outras características importantes que não se encontram entre os requisitos de outras linguagens de conteúdo (como é o exemplo do LISP, PROLOG, SQL, etc.) e que lhe auferem um predomínio sobre elas, como é o caso de:

- *Transparência*: um dos requisitos centrais do KIF é o de possibilitar meios práticos para a tradução de conhecimento do tipo declarativo em linguagens típicas de representação de conhecimento;

- *Legibilidade*: apesar do KIF não se direccionar basicamente a linguagens interactivas com as pessoas, a sua legibilidade para os humanos facilita, por um lado a descrição de representações da semântica das linguagens, por outro, a promulgação de linguagens para bases de conhecimento e, ainda, a ajuda aos humanos nos problemas da tradução de conhecimento.

- *Usabilidade*: embora o KIF inicialmente não tenha sido desenvolvido para ser utilizado em aplicações como uma linguagem de comunicação, pode também ser utilizado para este fim, se necessário.

## 3.4 Conclusão

No presente capítulo apresentaram-se os sistemas de comunicação para agentes, onde foram introduzidos alguns conceitos básicos sobre a interoperabilidade entre agentes, isto é, as linguagens de comunicação utilizadas e a sua origem. Neste ponto foram apresentadas as duas linguagens de comunicação mais utilizadas (KQML e FIPA), nomeadamente a sua sintaxe e os actos de comunicação. Uma comparação entre elas não poderia deixar de se apresentar, já que a primeira a aparecer (KQML), e as suas limitações segundo os autores referenciados na sua secção, deram origem à segunda (FIPA-ACL). Finalmente, como as linguagens de comunicação apenas servem de base ao transporte das mensagens entre os agentes, torna-se necessário a definição de uma linguagem que atribua algum significado ao conteúdo transportado pelas linguagens de comunicação: as linguagens de conteúdo, apresentando-se como exemplo a linguagem KIF.

O objectivo principal da introdução deste capítulo na presente tese de mestrado, prendeu-se com o facto de se poder dar algum suporte à compreensão da utilização da linguagem

de comunicação KQML na arquitectura multi-agentes apresentada no capítulo 4 e na explicação da definição de raíz de uma linguagem de conteúdo (MngmtCL) especificada no capítulo 5 que serve os propósitos da arquitectura.

# Capítulo 4

# Arquitectura Multi-Agente

## 4.1 Introdução

No presente capítulo faz-se a apresentação de uma arquitectura com a qual é pretendida a utilização de agentes para agir em nome de um administrador de uma rede privada e de um utilizador de uma rede pública, no sentido de negociar e estabelecer contratos de prestação de serviços de ligação. No contexto actual das redes de telecomunicações, a prestação de um serviço deste tipo estabelece-se à custa de um factor humano que torna o processo lento. Com a utilização da tecnologia dos agentes, consegue-se encurtar o processo, fazendo uma utilização mais optimizada dos recursos de rede.

Na apresentação da arquitectura faz-se uma identificação dos agentes e respectivos papéis, indicando-se ainda a linguagem de comunicação adoptada e o mecanismo de transporte das mensagens utilizado. Para além destes, são apresentados os aspectos funcionais da arquitectura, isto é, como a arquitectura funciona de um modo global. Finalmente, partindo de uma possível topologia de rede e de agentes, são apresentados e descritos possíveis cenários de comunicação.

## 4.2 Arquitectura do Sistema

No presente trabalho, a tecnologia dos agentes é orientada a cenários heterogéneos de gestão de serviços ATM, fornecendo meios para o estabelecimento de circuitos de comunicação extremo-a-extremo com diferentes graus de conhecimento. Desta forma, reduz-se a operação e a gestão de dados através da troca de porções precisas de informação, ou detectando e reagindo a alterações de estado das redes ATM [29].

O primeiro passo na definição de uma arquitectura de agentes, após a análise do domínio de um problema e da actividade da mesma, é a definição do papel dos agentes e dos protocolos de comunicação. Um papel define as características ou o comportamento esperado de um agente. Na arquitectura aqui proposta encontramos três classes de papéis, um para cada tipo de agentes: o *User Agent* (UA), o *ATM Service Provider Agent* (ASPA) e o *Facilitator Agent* (FA).

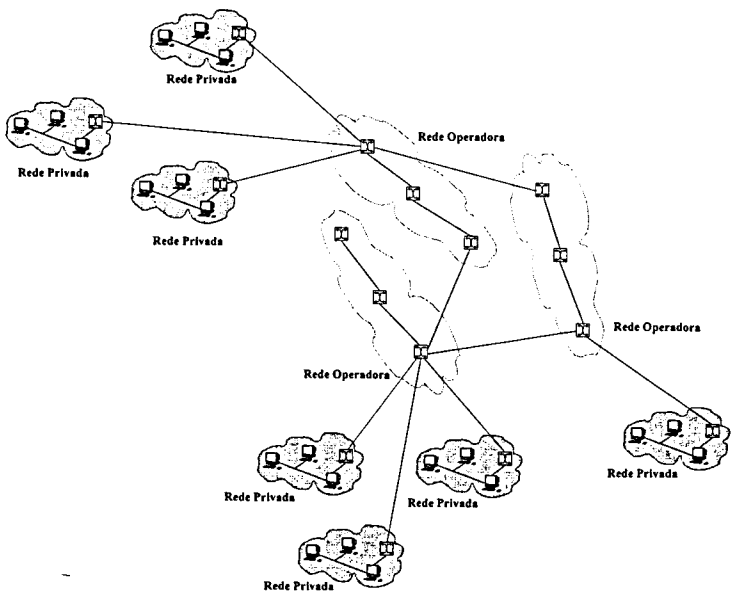



Figura 4.1: Possível topologia de rede

O agente UA é responsável por criar uma interface amigável ao administrador de uma rede privada. É sua tarefa estabelecer um contrato de tráfego com todos os agentes ASPA disponíveis, e escolher um que melhor sirva os interesses do administrador da referida rede.

Os agentes ASPA, por seu turno, são responsáveis por garantir a interoperabilidade entre as redes heterogéneas enquanto, por exemplo, monitorizam circuitos. Deve existir um agente ASPA em cada domínio de serviço ATM, que anuncia ao agente FA as suas capacidades através de uma linguagem de comunicação.

O agente FA constrói o conhecimento da comunidade de agentes através de informações deles recebidas, onde expressam as suas vontades, desejos e crenças. Por exemplo, todos os agentes ASPA informam o agente FA que estão disponíveis para processar pedidos de estabelecimento de circuitos de comunicação. Desta forma, ele pode indicar facilmente a todos os agentes UA quais os agentes ASPA podem ser contactados para prestar serviços desse género. Outra das funções de um agente FA é a de fazer chegar devidamente todas as mensagens, desde os agentes emissores aos agentes receptores.

Quando um utilizador final possui disponíveis múltiplos serviços de transporte ATM, cada um dos quais oferecidos por uma rede operadora, encontra-se perante um desafio: escolher qual operadora lhe oferece um contrato de prestação de serviços a custos mais reduzidos, o que implica ter de escolher um contrato que lhe ofereça ligações com qualidade de serviço (QdS) correspondente ou superior à por ele inicialmente especificada, garantindo custos reduzidos. Assim que a escolha é feita, o cliente (utilizador final) espera que a operadora local lhe encaminhe o tráfego pela operadora escolhida, possivelmente através da subcontratação do serviço de ligação a outras operadoras, conforme as circunstâncias.

Para ter êxito, a operadora deve estabelecer uma ligação com os seus clientes (utiliza-

dores finais). A operadora encontra-se ligada a um conjunto de redes privadas, podendo ainda, estar ligada a outras redes públicas, estabelecendo ligações de longa distância (Figura 4.1).

No caso da arquitectura aqui proposta, assume-se que é objectivo dos ASP's (*ATM Service Providers*) fornecer aos seus clientes ligações extremo-a-extremo com uma certa QdS, baseadas numa arquitectura escalável e flexível de encaminhamento de tráfego. Não serão aqui efectuadas referências mais pormenorizadas sobre essa arquitectura por não ser objectivo do trabalho. No entanto, pode ser encontrada informação sobre a referida matéria (protocolos de encaminhamento e de sinalização, especificado pelo ATM Forum) em [30] e em [31].

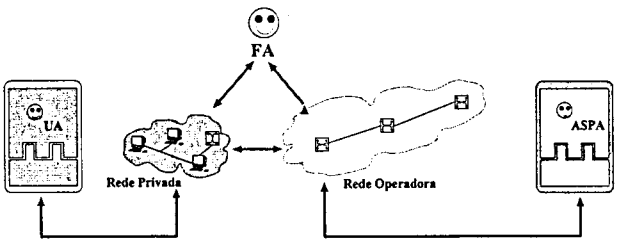



Figura 4.2: Arquitectura simplificada dos agentes

Como pode facilmente ser compreendido pela análise da figura 4.2, a arquitectura proposta é uma arquitectura bastante simples, pois o propósito principal é estabelecer um circuito de comunicação entre duas redes privadas, através de uma rede pública e de uma forma bastante mais rápida do que a tradicional contratação de serviço com operadoras.

Podemos, então, considerar na arquitectura três níveis: o primeiro constituído pelos agentes das redes privadas (UA), o segundo pelos agentes das redes públicas (ASPA), e no topo da arquitectura por um ou mais agentes FA.

Em cada rede privada deve existir um agente UA cujo objectivo é, conforme o atrás exposto, estabelecer um contrato de prestação de serviços com um agente ASPA se estiver localizado na rede local, ou aceitar um circuito negociado por outro e proceder às configurações necessárias das interfaces de rede, caso esteja localizado na rede privada final do circuito de comunicação. De notar que, apesar de um circuito de comunicação ter dois sentidos, entende-se como rede inicial aquela que inicia o contrato de prestação de serviços, e por rede final aquela que aceita o circuito de ligação. Outra função dos agentes UA é o pedido de informações sobre o tipo de serviços que podem ser prestados por uma rede operadora, bem como o custo a eles associado.

Em cada rede pública deve existir pelo menos um agente ASPA que deve aceitar (se estiver em condições de o fazer) um pedido de fornecimento de serviço, tentando manter a especificação pedida pelos agentes UA, ou propor outras em alternativa. Caso não consiga atingir a rede privada de destino, pode sempre proceder a uma subcontratação de serviço com outras redes públicas.

## 4.3 Aspectos Funcionais

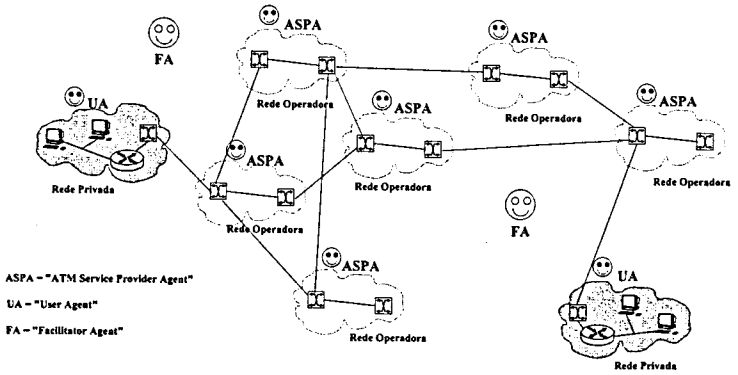



Figura 4.3: Topologia de Redes e de Agentes: Caso de Estudo

Imagine-se que, no contexto da figura 4.3, duas redes privadas (utilizadores finais), possivelmente localizadas em países diferentes, desejam estabelecer um circuito de comunicação. Imagine-se também que cada uma delas está ligada a uma rede pública local que, por seu lado, se encontra ligada a outras redes públicas.

Quando um utilizador final tenciona comunicar com outro, espera obter uma lista completa de operadoras fornecedoras de serviços ATM disponíveis para estabelecer o respectivo circuito de comunicação, um conjunto detalhado de parâmetros QdS e, ainda, os custos referentes ao serviço de encaminhamento do tráfego ao longo do circuito de comunicação.

No nosso caso, os utilizadores finais são representados pelos agentes UA e as operadoras pelos agentes ASPA. Um agente UA é responsável por contactar os agentes ASPA disponíveis por forma a obter um conjunto de propostas de contratos de prestação de serviços. Para conhecer todos os agentes ASPA disponíveis, ele deverá contactar o agente FA. No entanto, para que o agente FA possa informar correctamente sobre todos os agentes ASPA possibilitados de prestar um serviço de ligação, estes últimos devem registar-se na sua base de conhecimento. Por seu turno, os agentes ASPA são responsáveis por esboçar uma proposta de contrato de prestação de serviços após reunirem um conjunto de informações de rede. Numa primeira fase, um agente ASPA deve certificar se o utilizador final de destino se encontra acessível e depois verificar se se encontram disponíveis os recursos afectos por uma determinada especificação de serviço. Finalmente, dependendo dos resultados das tarefas anteriores, deve obter informações sobre os custos envolvidos. Se verificar que não é possível estabelecer um circuito de comunicação, deve enviar uma mensagem de negação de prestação do serviço ao agente UA que o contactou. Por outro lado, se verificar que existem meios disponíveis, deve proceder à submissão de uma proposta de contrato de prestação de serviços.

Embora a arquitectura de agentes proposta tenha um grande interesse na situação atrás descrita (pedido de estabelecimento de um circuito de comunicação), outras situações podem ser descritas, como é o caso do pedido de libertação de um circuito estabelecido,

devido a variações de custos.

Quando um circuito se encontra estabelecido, um agente UA pode desejar que outros agentes ASPA lhe forneçam diferentes propostas de contratos de prestação de serviços, por forma a reduzir os custos totais referentes à utilização de um circuito de comunicação entre duas redes privadas. Nesta situação, o agente UA deve contactar uma vez mais o agente FA para ser informado sobre todos os agentes ASPA capazes de oferecerem um serviço e optar pelo que lhe propuser um contrato com custos mais reduzidos. Se tal se verificar, deve ser estabelecida uma nova ligação até ao mesmo endereço de destino e com a mesma QdS definida anteriormente. Uma vez que existe uma variedade de caminhos possíveis desde o endereço de rede inicial até ao endereço final, (informação sobre o mecanismo de encaminhamento de tráfego em redes ATM pode ser encontrado em [32], [33] e [34]), cada um dos quais, possivelmente pertencente a outra rede operadora, o agente ASPA deve certificar-se que o circuito passa através de caminhos que envolvam custos o mais reduzidos possível. Após o estabelecimento do novo circuito, o tráfego gerado entre as duas redes privadas em questão deve ser desviado do antigo para o novo. Este procedimento envolve os agentes UA daquelas redes privadas, uma vez que é necessário associar convenientemente os novos pares VPI/VCI à interface de rede ATM (um par em cada rede). Após a reconfiguração das interfaces de rede, o circuito de comunicação antigo deve ser libertado. Desta forma, o tráfego entre as duas redes flui normalmente sem interrupções.

As mensagens trocadas pelos agentes devem possuir uma semântica bem definida. Como tal, é utilizada uma linguagem de comunicação standardizada para que, nas diferentes redes, os agentes possam interagir de acordo com as funções atribuídas a cada um. Adoptando o KQML como a linguagem de comunicção utilizada na arquitectura, conseguem-se codificar todos os elementos básicos de interacção que possam ocorrer entre os agentes.

No contexto da arquitectura, as mensagens devem ser escalonadas e orientadas a acontecimentos, sendo enviadas através de um modo assíncrono. Desta forma, o mecanismo de transporte deve suportar um endereçamento físico único, baseado nas funções atribuídas aos seus agentes. Para tal, o mecanismo deve suportar modos *"unicast"*, *"multicast"* e *"broadcast"*, facilmente implementáveis em java por intermédio de classes desenvolvidas propositadamente por forma a suportarem o KQML, como é o caso do JKQML (Java-based KQML API) [35].

## 4.4 Cenários de Comunicação

Suponhamos que a arquitectura de agentes se encontra implementada onde estão bem identificados os seus agentes. Nas redes privadas encontramos os agentes UA, nas redes públicas os agentes ASPA, e a um nível superior destes encontramos um ou mais agentes FA. Neste contexto, pretende-se aqui apresentar todos os cenários de comunicação possíveis de ocorrer entre os agentes da arquitectura, cenários esses que são descritos em termos de performativas KQML. Para melhor compreensão, apresenta-se a figura 4.4 onde se mostra uma possível topologia de arquitectura de rede e se identificam todos os seus agentes. Para além dessa identificação, mostra-se também a relação entre eles, representada na forma de

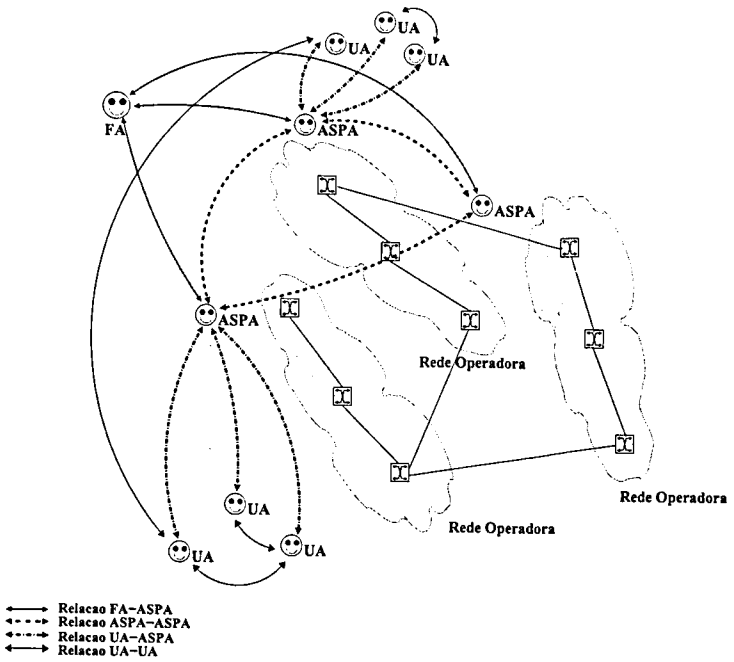



Figura 4.4: Topologia da relação global entre agentes

fluxos de informação, isto é, o sentido da troca de mensagens entre os mesmos.

### 4.4.1 Indicação de Operadoras

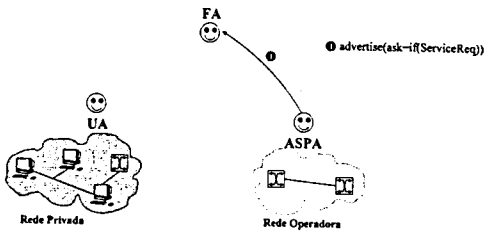



Figura 4.5: Indicação ao agente FA sobre possibilidade de prestar serviços

Numa fase inicial do funcionamento da arquitectura, os agentes das redes públicas devem informar o agente FA que estão predispostos a prestar serviços de estabelecimento dinâmico de circuitos de comunicação. Como é óbvio, um agente UA não conhece (nem tem de conhecer) todos os agentes das operadoras que o poderão servir. Para o efeito, todos os agentes ASPA comunicam ao(s) agente(s) FA que estão disponíveis para receber pedidos de fornecimento de serviços de ligação. A figura 4.5 representa este cenário, mostrando a performativa KQML utilizada (*advertise*) pelos agentes ASPA.

Em termos da forma do acto de comunicação, isto é, como é utilizado o KQML, o seu conteúdo é exemplificado no cenário 4.1: o nome da performativa é *advertise*, o seu conteúdo *ask-if(ServiceReq)*, e é efectuada por um agente ASPA ao agente FA. O significado atribuído a esta performativa é o de um agente FA saber que o seu emissor (um agente ASPA) consegue processar o seu conteúdo (uma outra performativa embutida - *ask-if*), o que em termos práticos corresponde a afirmar que está disponível para receber pedidos de negociação sobre o estabelecimento de circuitos. A comunicação entre um agente regular (UA e ASPA) e um *Facilitator Agent* é feita ao nível da camada de aplicação e este só opera essencialmente com a ontologia (*kqml-ontology*).

Na realidade, o que esta performativa indica é o seguinte: o agente ASPA informa o agente FA que está disponível para processar todo o tipo de mensagens cujo conteúdo seja uma outra performativa embutida (*ask-if*).

Como conteúdo desta última, é utilizada uma expressão **ServiceReq**, utilizada na linguagem de conteúdo desenvolvida de raíz para o contexto desta arquitectura e que se encontra especificada no capítulo 5. Como se poderá verificar nesse capítulo, esta expressão serve para pedir por uma proposta de contrato de prestação de serviços. De notar que a expressão **ServSpec** apenas tem significado na linguagem de conteúdo e ontologia especificados dentro da performativa **(:language MngmtCL :ontology CAC)**.

CENÁRIO 4.1 : **Indicação de operadora com possibilidade de prestar serviços**

```
(advertise
   :sender      ASPA
   :receiver    facilitator
   :reply-with id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
                   :reply-with id?
                   :language   MngmtCL
                   :ontology   CAC
                   :content    "ServiceReq"))
```

Para além da situação de indicação sobre a possbilidade de prestar um serviço, pode verificar-se ainda que um agente ASPA, por um qualquer motivo ligado à rede operadora que representa (por exemplo manutenção), não pode oferecer mais serviços de ligação. Neste caso, este agente deverá avisar o respectivo agente FA de tal facto. Para tal, muito similarmente à situação anterior, deverá enviar-lhe uma performativa *unadvertise* que, na sua essência, cancela a performativa *advertise* inicialmente enviada. O conteúdo desta performativa apresenta-se no cenário 4.2, abaixo representado. Analisando a sua forma, verifica-se pela palavra reservada *unadvertise* o tipo de acto de comunicação envolvido e quais os agentes envolvidos (ASPA e FA). Para que realmente o agente FA saiba que é a impossibilidade temporária de prestação de um serviço, o conteúdo da performativa deve ser o mesmo do da performativa *advertise* atrás referida. O propósito da utilização desta

performativa é o de retirar da base do conhecimento do agente FA a informação sobre a eventualidade de processar performativas *ask-if(serviceReq)* por parte daquele agente ASPA.

CENÁRIO 4.2 : **Indicação de indisponibilidade temporária de prestar serviços**

```
(unadvertise
   :sender      ASPA
   :receiver    facilitator
   :reply-with id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
                   :reply-with id?
                   :language   MngmtCL
                   :ontology   CAC
                   :content    "ServiceReq"))
```

### 4.4.2 Informação ao agente FA Sobre a Topologia dos Agentes ASPA

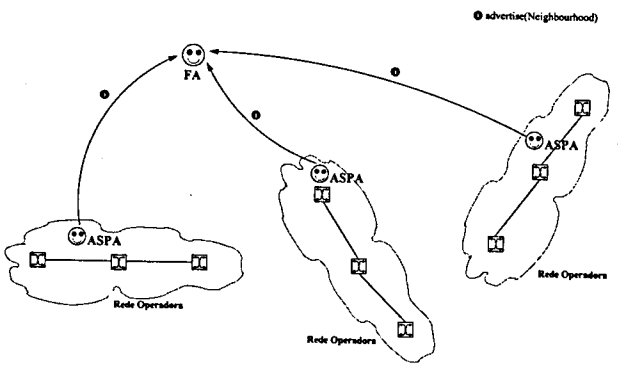



Figura 4.6: Informação sobre as redes vizinhas

Para que o agente FA possa informar convenientemente um agente UA sobre quem, numa fase inicial, lhe pode prestar um serviço pretendido, tem que ter uma noção da interligação das redes operadoras. De facto, apesar de todos os agentes ASPA poderem estabelecer circuitos, informando o agente FA nesse sentido (efectuando o *advertise* à possibilidade de estabelecer circuitos), nem todos podem prestar o serviço de ligação a todos os agentes UA. Assim, e para que o agente FA possa responder convenientemente a uma performativa do tipo *broker* efectuada por um agente UA, este tem que conhecer quais os agentes ASPA têm possibilidade imediata de prestar um serviço. Como tal, o agente FA deverá ter a noção da topologia das redes representadas pelos agentes ASPA. Para o efeito,

quando os agentes ASPA se registam, enviam uma performativa *advertise* incluindo uma expressão **Neighbourhood**, que representa os nomes dos agentes das suas redes vizinhas.

Na figura 4.6 encontra-se representada de uma forma gráfica a relação entre o agente FA e os agentes ASPA no que se refere à informação sobre a vizinhança de redes. No cenário 4.3 encontra-se a forma da performativa *advertise* enviada por um agente ASPA ao agente FA para o informar sobre os agentes seus vizinhos.

CENÁRIO 4.3 : **Informação ao agente FA sobre a vizinhança de agentes ASPA**

```
(advertise
   :sender       ASPA
   :receiver     facilitator
   :in-reply-to  id?
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      "Neighbourhood"
```

## 4.4.3 Determinação de Operadoras

Sempre que um agente UA quer pedir e negociar o estabelecimento de um circuito de comunicação, deverá saber quais os agentes ASPA, em representatividade de uma rede operadora, lhe poderão estabelecer contratos de prestação de serviços. Assim, estão previstos na arquitectura vários mecanismos para esse fim, os quais diferem apenas no tipo de performativas KQML utilizadas nos correspondentes actos de comunicação. Essas performativas englobam as chamadas performativas de *broker*, de *recommend* e de *recruit*. Inicialmente, estes actos ocorrem entre agentes UA e agentes FA e, posteriormente, entre agentes FA e ASPA, conforme é visto de seguida, onde se descreve a utilização de cada uma dessas performativas e seu significado específico no contexto da determinação de operadoras.

### 4.4.3.1 Actos de Comunicação Envolvendo Performativas "broker-one" e "broker-all"

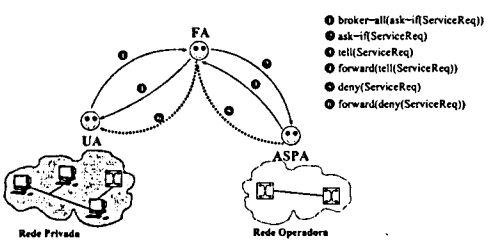



Figura 4.7: Broker-all ao facilitator

Um agente UA antes de pedir a abertura de um circuito de comunicação necessita de saber quem lhe poderá prestar esse serviço. Para tal, entra em contacto com um agente FA que lhe dará essa informação. Em termos de mensagens entre estes dois agentes, o agente UA envia ao agente FA uma performativa *broker-one* ou *broker-all* incluindo como conteúdo outra performativa (*ask-if*), aquela que o agente ASPA informou estar disponível para processar. A diferença da utilização destas duas performativas reside apenas no número de agentes ASPA que o agente FA indica ao agente UA. Utilizando uma performativa *broker-all* o agente FA indica ao agente UA todos os agentes ASPA que o informaram sobre a possibilidade de processarem pedidos de estabelecimento de circuitos. Utilizando uma performativa *broker-one*, o agente FA apenas indica um agente ASPA. O que se verifica é que, a um nível superior, utiliza-se uma performativa *broker* com a linguagem de comunicação KQML e a ontologia kqml-ontology. Como conteúdo desta, utiliza-se uma outra performativa *ask-if* com a linguagem de conteúdo MnmgtCL e a ontologia CAC por nós definida.

No conteúdo da performativa *ask-if*, inclui-se uma expressão **ServiceReq**, utilizada para "requisitar"a prestação de um serviço, ou para pedir informações sobre o estabelecimento do mesmo. O agente FA após a recepção desta performativa, e após ter pesquisado por todas as performativas *advertise* que lhe foram enviadas, envia o conteúdo desta performativa (*broker-one* ou *broker-all*) aos agentes que efectuaram o *advertise* ao conteúdo desta.

A reacção a estas performativas efectua-se da seguinte forma: o agente FA, após a recepção da performativa, procede ao envio do seu conteúdo (*ask-if(ServiceReq)*) a um ou a todos (conforme se utilize uma ou outra performativa *broker*) os agentes ASPA que efectuaram o respectivo *advertise*. Por seu lado, o(s) agente(s) ASPA deverá(ão) tentar abrir o circuito especificado pela expressão **ServiceReq** indicada no conteúdo dessas performativas. No caso de ser possível concretizar o pedido, eles enviam ao agente FA uma performativa *tell(ServiceOffer)*, onde informam ser possível abrir o circuito, indicando ainda os termos do mesmo (se novas especificações, forma de cálculo de custos, e tempo).

O agente FA após a recepção das performativas, em resposta, só tem que as fazer seguir (efectuar o seu *forward*) até ao agente UA para que ele possa decidir se pretende ou não assinar um contrato de prestação de serviços (*Service Level Agreement - SLA*). Caso o agente ASPA não consiga, por motivos diversos (comprometimento de ligações existentes, endereço final inalcançável), abrir o circuito pretendido, deve enviar ao agente FA uma performativa *untell* com a mesma expressão **ServiceReq**. De qualquer das formas, o agente FA apenas terá de fazer seguir essa performativa até ao agente UA, ficando este a saber que, apesar de o agente ASPA poder abrir circuitos, naquele momento está impossibilitado de o fazer.

O cenário 4.4 representa um acto de comunicação de uma forma exaustiva, onde se encontram as mensagens trocadas entre os agentes no sentido da indicação, por parte de um agente FA a um agente UA a pedido deste último, de todos os agentes ASPA com possibilidade de estabelecerem circuitos de comunicação. Neste acto encontramos um agente ASPA1 que consegue abrir o circuito pretendido e um outro (ASPA2) que não consegue.

CENÁRIO 4.4 : **Indicação de Operadoras disponíveis para prestar serviços de ligação**

```
(broker-all
   :sender     UA
   :receiver   facilitator
   :reply-with id?
   :language   KQML
   :ontology   kqml-ontology
   :content    (ask-if
                  :sender     UA
                  :reply-with id2
                  :language   MngmtCL
                  :ontology   CAC
                  :content    "ServiceReq"))

(ask-if
   :sender      facilitator
   :receiver    ASPA1
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")

(tell
   :sender      ASPA1
   :receiver    facilitator
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer")

(forward
   :from        ASPA1
   :sender      facilitator
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
```

*Continua na página seguinte ...*

**Indicação de Operadoras disponíveis para prestar serviços de ligação** *(continuação)*

```
   :content      (tell
                    :receiver    UA
                    :language    MngmtCL
                    :ontology    CAC
                    :content     "ServiceOffer"))
...
(ask-if
   :sender       facilitator
   :receiver     ASPA2
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "ServiceReq")

(untell
   :sender       ASPA2
   :receiver     facilitator
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "ServiceReq")

(forward
   :from         ASPA2
   :sender       facilitator
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      (untell
                    :receiver    UA
                    :language    MngmtCL
                    :ontology    CAC
                    :content     "ServiceReq"))
```

#### 4.4.3.2 Actos de Comunicação Envolvendo Performativas "recomend-one" e "recomend-all"

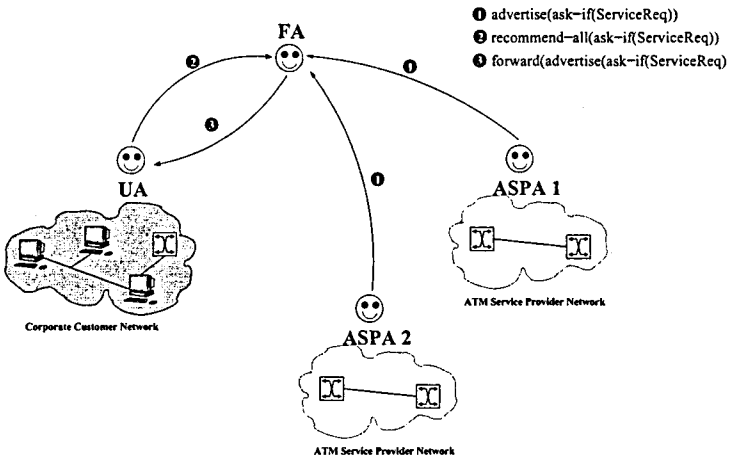



Figura 4.8: Recomendação de todas as operadoras

Caso um agente UA pretenda que um agente FA lhe recomende um agente ASPA, ele deve enviar uma performativa *recomend-one* ao agente FA, onde, como conteúdo, inclui uma outra, (*ask-if*), a utilizada em performativas *advertise* efectuadas pelos agentes ASPA como forma de indicar que aceitam pedidos para prestação de serviços.

O agente FA, após verificar no conjunto de performativas *advertise* que recebeu dos agentes ASPA, responde ao agente UA com uma performativa *forward*, utilizando como conteúdo uma performativa *advertise* efectuada por aqueles.

CENÁRIO 4.5 : **Pedido ao agente FA para recomendar um agente ASPA**

```
(recommend-one
   :sender      UA
   :receiver    facilitator
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
                   :sender    UA
                   :language    MngmtCL
                   :ontology    CAC
                   :content     "ServiceReq"))

(forward
```

*Continua na página seguinte ...*

**Pedido ao agente FA para recomendar um agente ASPA** *(continuação)*

```
   :from         ASPA
   :to           UA
   :sender       facilitator
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      (advertise
                    :sender     ASPA
                    :receiver   UA
                    :reply-with id?
                    :language   KQML
                    :ontology   kqml-ontology
                    :content    (ask-if
                                   :sender      UA
                                   :receiver    ASPA
                                   :in-reply-to id?
                                   :language    MngmtCL
                                   :ontology    CAC
                                   :content     "ServiceReq")))
```

No caso de um agente UA querer que o agente FA lhe recomende todos os agentes ASPA, este deve enviar uma performativa *recomend-all* em vez de *recomend-one*. Estas duas performativas são muito idênticas, diferindo apenas neste aspecto: no caso da performativa *recomend-one*, apesar do agente FA procurar no conjunto de mensagens utilizando performativas *advertise* que lhe foram enviadas, apenas recomenda um agente ASPA; no caso da utilização de uma performativa *recomend-all*, o mesmo recomenda todos os agentes que efectuaram o mesmo *advertise*, como pode ser facilmente verificado na análise do cenário 4.6, abaixo representado.

CENÁRIO 4.6 : **Pedido ao agente FA para recomendar todos os agentes ASPA**

```
(recommend-all
   :sender       UA
   :receiver     facilitator
   :in-reply-to  id?
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      (ask-if
                    :sender    UA
```

*Continua na página seguinte ...*

**Pedido ao agente FA para recomendar todos os agentes ASPA** *(continuação)*

```
                  :language    MngmtCL
                  :ontology    CAC
                  :content     "ServiceReq"))

(forward
   :from        ASPA1
   :to          UA
   :sender      facilitator
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (advertise
                  :sender     ASPA1
                  :receiver   UA
                  :reply-with id?
                  :language   KQML
                  :ontology   kqml-ontology
                  :content    (ask-if
                                 :sender      UA
                                 :receiver    ASPA1
                                 :in-reply-to id?
                                 :language    MngmtCL
                                 :ontology    CAC
                                 :content     "ServiceReq")))
...

(forward
   :from        ASPA2
   :to          UA
   :sender      facilitator
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (advertise
                  :sender     ASPA_n
                  :receiver   UA
```

*Continua na página seguinte ...*

**Pedido ao agente FA para recomendar todos os agentes ASPA** *(continuação)*

```
                :reply-with id?
                :language   KQML
                :ontology   kqml-ontology
                :content    (ask-if
                               :sender      UA
                               :receiver    ASPA_n
                               :in-reply-to id?
                               :language    MngmtCL
                               :ontology    CAC
                               :content     "ServiceReq"))
```

#### 4.4.3.3 Actos de Comunicação Envolvendo Performativas "recruit-one" e "recruit-all"

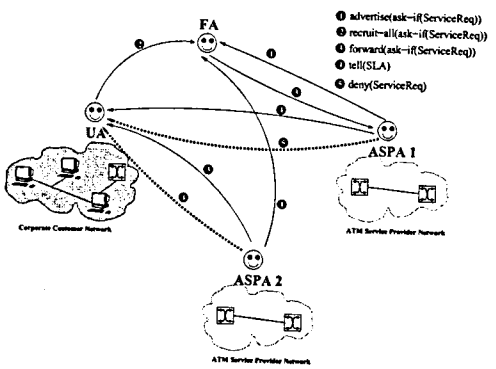



Figura 4.9: Recrutamento de todos agentes ASPA

Outra situação que pode ocorrer neste contexto de determinação de operadoras é a de um agente UA desejar que seja o agente FA a propor-lhe s agentes ASPA, isto é, que seja ele a recrutar os agentes ASPA para o fornecimento de serviços. Assim, dependendo do tipo de recrutamento, empregam-se as performativas *recruit-one* e *recruit-all*, diferindo estas apenas no número de agentes ASPA recrutados.

Se um agente UA quiser que seja o agente FA a sugerir um agente ASPA, ele terá que enviar uma performativa *recruit-one* onde, incluindo no seu conteúdo uma performativa *ask-if* com uma expressão **ServiceReq**, especifica o serviço pretendido. Por seu turno, o agente FA após a recepção desta performativa verifica quais os agentes ASPA fizeram o *advertise* à possibilidade de prestação de serviços e "recruta" um. Por outras palavras, o agente FA inicia o processo de indicação ao agente UA de um potencial agente ASPA, capaz de lhe prestar um serviço de ligação. Este processo composto por duas fases: a primeira fase, corresponde ao envio de uma performativa *forward* ao agente ASPA recrutado com o mesmo conteúdo da performativa *recruit-one* recebida inicialmente; a segunda,

corresponde à resposta do agente ASPA directamente ao agente UA que pediu ao agente FA para lhe recrutar um agente fornecedor de serviços. Aqui, o agente ASPA recrutado deve primeiro verificar se consegue fornecer o serviço pretendido. Caso tal seja possível, responder-lhe-á com uma performativa *tell* utilizando uma expressão **ServiceOffer** com os termos do serviço inicialmente pretendido, ou com outro tipo de serviço. No caso de se ver impossibilitado de prestar o serviço, em vez de enviar como resposta esta performativa *tell*, envia uma outra (*untell*(**ServiceReq**)), pretendendo com ela dizer que percebeu a mensagem mas não a pode processar, isto é, abrir o circuito pretendido.

No caso de um agente UA pretender que sejam "recrutados" todos os agentes, a performativa utilizada é a *recruit-all*, sendo o processo semelhante, diferindo apenas no número de agentes ASPA envolvidos. A figura 4.9 pretende representar através de uma forma gráfica o cenário global do pedido de recrutamento de todos os agentes ASPA, e o cenário 4.7 apresenta as mensagens trocadas entre os agentes onde, se verifica que um agente ASPA1 consegue abrir o circuito especificado e um agente ASPA2 não.

CENÁRIO 4.7 : **Pedido ao agente FA para recrutar todos os agentes ASPA**

```
(recruit-all
   :sender      UA
   :receiver    facilitator
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
                   :sender    UA
                   :language  MngmtCL
                   :ontology  CAC
                   :content   "ServiceReq"))

(forward
   :from        UA
   :to          ASPA1
   :sender      facilitator
   :receiver    ASPA1
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
                   :sender      UA
                   :receiver    ASPA1
                   :in-reply-to id?
                   :reply-with  id?
                   :language    MngmtCL
```

*Continua na página seguinte ...*

**Pedido ao agente FA para recrutar todos os agentes ASPA** *(continuação)*

```
                    :ontology    CAC
                    :content     (ServiceReq)))

(tell
   :sender       ASPA1
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      (ServiceOffer))
...
(forward
   :from         UA
   :to           ASPA2
   :sender       facilitator
   :receiver     ASPA
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      (ask-if
                    :sender      UA
                    :receiver    ASPA2
                    :in-reply-to id?
                    :reply-with  id?
                    :language    MngmtCL
                    :ontology    CAC
                    :content     (ServiceReq)))

(untell
   :sender       ASPA2
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      (ServiceReq))
```

### 4.4.4 Pedido de Informações Sobre Serviços

Se um agente UA não sabe que tipo de serviços um agente ASPA pode facultar, ele terá que lhe pedir informações sobre eles. Para o efeito, a arquitectura prevê várias formas: um agente UA pode, por um lado querer que um agente ASPA lhe envie de uma só vez informações sobre os serviços que é capaz de oferecer, podendo enviar tudo numa mensagem única, ou em várias e, por outro, querer que os agentes ASPA lhe vão enviando informações sobre os serviços à medida que eles vão estando possíveis de prestar. Assim, de seguida apresentam-se as três formas prvistas na arquitectura para se pedirem informações sobre serviços, evidenciando as diferenças resultantes da utilização das respectivas distintas performativas.

#### 4.4.4.1 Pedido de Informações Sobre Serviços Envolvendo Uma Única Mensagem de Resposta

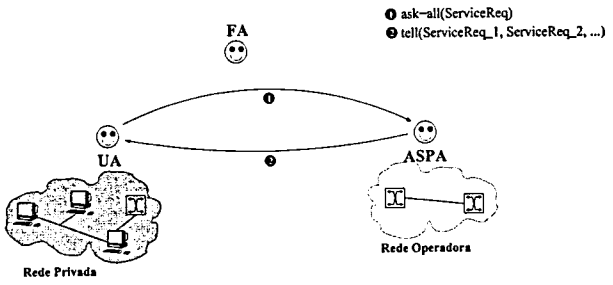



Figura 4.10: Pedido de informação sobre todos os tipos de serviço possíveis de serem prestados, utilizando uma performativa ask-all

O agente UA envia directamente a um agente ASPA uma performativa *ask-all* onde, como conteúdo, utiliza uma expressão **ServiceReq**. Com esta forma de procedimento, ele pretende receber do agente ASPA envolvido neste acto de comunicação todas as informações disponíveis sobre os serviços de ligação possíveis de serem prestados. A figura 4.10 representa de uma forma gráfica todo este processo. Em resposta, o agente ASPA envia-lhe uma performativa *tell*, passando no seu conteúdo as informações sobre todos os tipos de serviços de ligação, formalizadas num conjunto de expressões **ServiceOffer**.

CENÁRIO 4.8 : **Pedido de informações sobre todos os serviços disponíveis**

```
(ask-all
   :sender       UA
   :receiver     ASPA
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
```

*Continua na página seguinte ...*

**Pedido de informações sobre todos os serviços disponíveis** *(continuação)*

```
   :ontology    CAC
   :content     "ServiceReq")

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer_1, ServiceOffer_2,..., ServiceOffer_N")
```

#### 4.4.4.2 Pedido de Informações Sobre Serviços Envolvendo Múltiplas Mensagens de Resposta

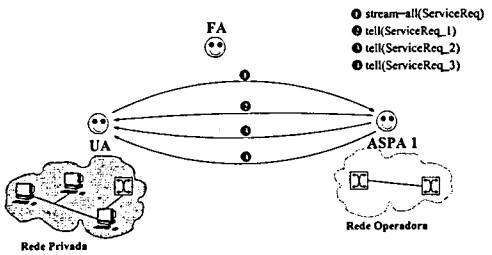



Figura 4.11: Pedido de informação sobre todos os tipos de serviço possíveis de serem prestados, utilizando uma performativa stream-all

Embora o propósito da utilização desta performativa seja igual ao da performativa *ask-all* descrita anteriormente, a diferença reside na forma do procedimento da resposta do agente ASPA. Assim, o agente UA envia uma performativa *stream-all*, de forma idêntica à performativa *ask-all* utilizada. Em resposta, o agente ASPA envia, não uma performativa *tell*, onde, no conteúdo, indica todos os tipos de serviço, mas várias, cada uma contendo a informação de um único serviço, conforme se exemplifica no cenário 4.9 a seguir representado.

CENÁRIO 4.9 : **Outra forma de pedir informações sobre todos os serviços disponíveis**

```
(stream-all
   :sender      UA
   :receiver    ASPA
   :in-reply-to id?
```

*Continua na página seguinte ...*

**Outra forma de pedir informações sobre todos os serviços disponíveis** *(continuação)*

```
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq)

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer_1)

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer_2)  ...

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer_N)
```

A figura 4.11 pretende representar graficamente todo este outro processo que um agente UA pode utilizar, em alternativa, para receber informações sobre serviços de ligação.

#### 4.4.4.3 Subscrição de Recepção de Informações Sobre Serviços

Nesta última situação, e não fugindo ao pedido de informações sobre serviços, um agente UA pretende que um agente ASPA lhe vá enviando informações sobre serviços à medida que eles vão sendo possíveis de prestar, sem ter que estar sempre a pedir por eles. Aqui entra-se num contexto um pouco diferente, no qual é feita uma subscrição ao envio sistemático de

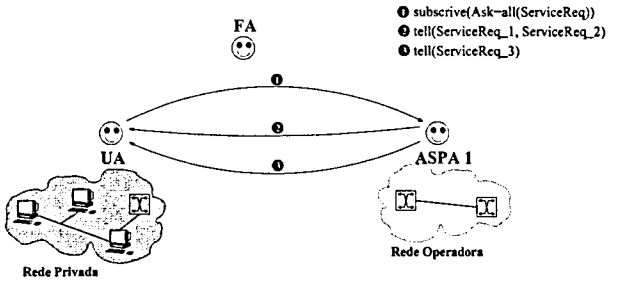



Figura 4.12: Subscrição de recepção de informação sobre todos os tipos de serviço

informações sobre serviços.

Para o propósito referido, um agente UA deve enviar uma performativa *subscribe* a um ou mais agentes ASPA, dependendo do número de agentes dos quais deseja receber as informações. Face à recepção desta performativa, os agentes ASPA vão informando sobre todos os serviços que lhes vai sendo possível prestar, facto este formalizado através do envio de uma série de performativas *tell* como as indicadas no cenário 4.10, onde são passadas, através de expressões **ServiceOffer**, as informações sobre os serviços.

CENÁRIO 4.10 : **Pedido de actualização sistemática de informações sobre todos os serviços disponíveis**

```
(subscribe
   :sender      UA
   :receiver    ASPA
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content
      (ask-all
         :sender      UA
         :receiver    ASPA
         :in-reply-to id?
         :reply-with  id?
         :language    MngmtCL
         :ontology    CAC
         :content     "ServiceReq"))
(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
```

*Continua na página seguinte ...*

**Pedido de actualização sistemática de informações sobre todos os serviços disponíveis** *(continuação)*

```
   :language     MngmtCL
   :ontology     CAC
   :content      "ServiceOffer_1, ServiceOffer_2")

(tell
   :sender       ASPA
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "ServiceOffer_N")
```

Analisando as performativas utilizadas por um agente UA nos cenários 4.8, 4.9 e 4.10, notamos diferenças na forma em que os agentes ASPA reagem. No caso do cenário 4.8, um agente ASPA utiliza apenas uma mensagem onde são passadas as informações de todos os serviços. No caso do cenário 4.9, os agentes ASPA utilizam uma mensagem para cada tipo de serviço. Finalmente, no caso do cenário 4.10, um agente ASPA vai enviando sistematicamente as informações dos serviços, podendo utilizar para este fim uma única ou variadas mensagens.

### 4.4.5 Pedido de Informação Sobre Custos

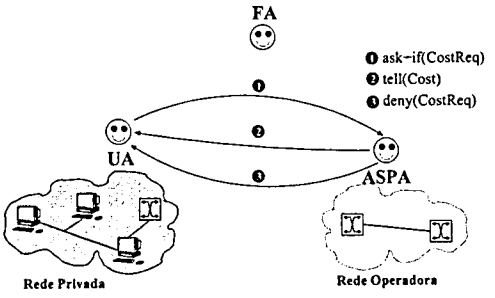



Figura 4.13: Pedido de informação sobre custos

Um agente UA, conhecendo à partida quais agentes ASPA lhe podem fornecer um serviço de ligação, pode desejar saber qual o custo do estabelecimento de um determinado circuito de comunicação. Assim, pede directamente ao agente ASPA para ser informado sobre um custo. Para tal, envia-lhe uma performativa *ask-if* onde, no conteúdo, utiliza uma expressão **CostReq** com uma expressão **ServSpec** encapsulada para caracterizar o serviço pretendido. O agente ASPA, após a recepção desta performativa, deverá passar ao

agente UA uma expressão que utilizará como função de cálculo de custos. Para tal, terá que verificar primeiro se pode estabelecer o circuito pretendido. Em caso afirmativo, enviará uma performativa *tell*, onde passará uma expressão **Cost** que contém uma expressão de cálculo de custos e os respectivos parâmetros. Uma vez que os custos associados a um serviço poderão ter várias métricas, torna-se mais fácil passar uma função devidamente parametrizada por forma a que seja o agente UA a calcular os custos de um serviço. Por outro lado, se o agente ASPA não possui um mecanismo *"off-line"* para saber o custo associado a um circuito e tiver que o abrir separadamente, contudo não conseguindo, enviará uma performativa *untell* utilizando no seu conteúdo a mesma expressão **CostReq** recebida.

O cenário 4.11 pretende representar as mensagens envolvidas nos actos de comunicação entre um agente UA e um agente ASPA para o pedido de informação sobre o custo de um serviço. A figura 4.13 pretende identificar graficamente todos os actos de comunicação envolvidos neste processo de pedido de informação de custos associados a um serviço de ligação.

CENÁRIO 4.11 : **Pedido de informação sobre custos associados a um serviço**

```
(ask-if
   :sender       UA
   :receiver     ASPA
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "CostReq")

(tell
   :sender       ASPA
   :receiver     UA
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "Cost")
```

### 4.4.6 Indicação de Estabelecimento de Um Circuito de Comunicação

Sempre que um agente UA pretende estabelecer um circuito de comunicação, e antes de iniciar o processo de contratação do correspondente serviço a uma operadora, deve avisar o agente UA da rede destinatária sobre esse facto. Assim, cria-se uma relação UA-UA onde o primeiro agente informa o segundo que deseja criar um circuito de comunicação entre as duas redes e pede para aceitar o serviço de ligação. Desta forma, esse segundo agente estará atento e saberá, quando um circuito for aberto, quem o pediu.

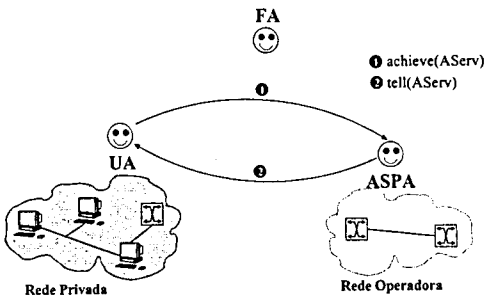



Figura 4.14: Indicação de pretensão de estabelecer circuito de comunicação

Para este efeito, imaginemos a existência de dois agentes $UA_1$ e $UA_2$, onde o primeiro deseja que uma rede pública estabeleça um circuito entre as redes que eles representam. Para que o circuito seja efectivamente estabelecido, o agente UA da outra rede privada, ao aperceber-se que existe um circuito estabelecido por uma rede pública, deve proceder à configuração das interfaces de rede, por forma a que o tráfego flua correctamente entre aquelas redes privadas.

Assim, o primeiro agente UA envia uma performativa *achieve* contendo uma expressão **AServ** ao segundo agente. A utilização de uma performativa deste tipo implica um pedido do agente emissor ao agente receptor para concretizar a tarefa especificada no conteúdo da mensagem. A utilização no conteúdo da performativa de uma expressão **AServ** representa, no contexto da linguagem de conteúdo **MngmtCL**, o pedido de aceitação de estabelecimento de um circuito de comunicação entre as redes dos dois agentes UA, e a informação sobre o endereço ATM e o par VPI/VCI utilizado no comutador ATM do lado da rede do agente emissor. O agente $UA_2$, deverá confirmar a recepção desta última e estar atento ao estabelecimento do circuito. Para tal, deverá informar o agente $UA_1$ que recebeu a performativa *achieve* anterior, enviando-lhe uma outra performativa *tell* onde utilizará no conteúdo a mesma expressão **AServ**.

A figura 4.14 mostra o cenário global deste tipo de relação entre os dois agentes UA e o cenário 4.12 as mensagens KQML trocadas neste contexto.

CENÁRIO 4.12 : **Indicação de estabelecimento de um circuito**

```
(achieve
    :sender       UA1
    :receiver     UA2
    :in-reply-to  id?
    :reply-with   id?
    :language     MngmtCL
    :ontology     CAC
    :content      "AServ")
```

*Continua na página seguinte ...*

**Indicação de estabelecimento de um circuito** *(continuação)*

```
(tell
    :sender       UA2
    :receiver     UA1
    :in-reply-to  id?
    :reply-with   id?
    :language     MngmtCL
    :ontology     CAC
    :content      "AServ")
```

### 4.4.7 Pedido de Estabelecimento de Circuitos

Apesar de, quando um agente UA deseja estabelecer um circuito deva saber que agentes ASPA podem prestar um serviço de ligação e estes últimos poderem submeter um contrato de prestação de serviços, aqui parte-se do princípio que os agentes UA já conhecem todas (ou algumas) operadoras que podem prestar o referido serviço. Assim, eles pedem directamente aos respectivos agentes ASPA sobre a possibilidade de prestação de um serviço, deixando de existir comunicação directa com o agente FA. Como foi referido no capítulo 3 as funções de um agente FA referem-se à implementação dos variados serviços de comunicação, tais como a manutenção do registo de nomes dos agentes da arquitectura, ao seguimento das mensagens dos respectivos serviços, ao encaminhamento de mensagens baseando-se no seu conteúdo, e à comparação da informação dos agentes. Assim, torna-se óbvio constatar que um agente UA precisa do agente FA para conhecer a existência dos agentes das redes públicas disponíveis. A partir do momento em que conhece essa informação, e porque além do nome dos agentes, o agente FA indica também a forma de uma mensagem lá chegar, deixa de ser necessário qualquer acto de comunicação entre o agente UA e este último, podendo o primeiro entrar em contacto directo com os agentes ASPA e trocar mensagens com eles.

Neste sentido, são de seguida descritos todos os actos referentes a este contexto, partindo-se da verificação da possibilidade de prestar um serviço e chegando à assinatura ou negação do contrato de prestação do mesmo - "*Service Level Agreement* - SLA".

#### 4.4.7.1 Verificação da Possibilidade de Prestação de Serviço

Através de uma performativa *ask-if* contendo uma expressão **ServiceReq**, um agente UA pergunta directamente ao agente ASPA da rede operadora que ele pretende contratar se este pode abrir um circuito com as características designadas por esta expressão, isto é, a especificação do serviço pretendido (efectivada pelo encapsulamento de uma expressão **ServSpec**, o custo mínimo **Cost**(se se aplicar) e a duração **TimeInterval** pretendidos.

Por seu lado, o agente ASPA após a recepção desta performativa, irá verificar se consegue fornecer o serviço, tentando reservar os respectivos recursos se for bem sucedido, e enviar uma performativa *tell* onde, utilizando uma expressão **ServiceOffer**, indica as condições (as especificadas pelo agente UA, ou outras se não forem possíveis aquelas) do

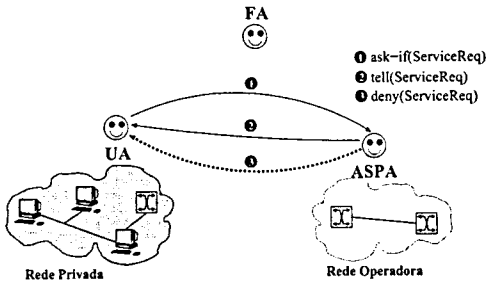



Figura 4.15: Pedido para verificar se é possível a abertura de um circuito

serviço a prestar. Como nem sempre poderá ser possível um agente ASPA conseguir prestar um serviço, ele deverá responder negativamente ao pedido de verificação da possibilidade de o prestar, enviando uma performativa *untell*, utilizando a mesma expressão recebida na performativa *ask-if* anterior, e indicando ao agente UA que, de momento, não poderá efectuar a abertura do circuito pretendido. O formato da mensagem trocada neste contexto é idêntica ao da afirmação da possibilidade de prestar um serviço, diferindo apenas no tipo de performativa utilizada: em vez de uma performativa *tell*, é utilizada uma performativa *untell*.

Atente-se ao cenário 4.13, especificado em termos do KQML, onde se mostra o formato das mensagens trocadas entre estes dois agentes no contexto do pedido e verificação de possibilidade de prestação de um serviço de ligação. Aqui encontramos dois tipos de resposta de um agente ASPA: uma performativa *tell* indicando possibilidade de prestar o serviço, e uma performativa *untell* negando a prestação do mesmo.

CENÁRIO 4.13 : **Verificação da possibilidade de prestação de serviço**

```
(ask-if
   :sender      UA
   :receiver    ASPA
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq"))

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer"))
```

*Continua na página seguinte ...*

**Verificação da possibilidade de prestação de serviço** *(continuação)*

```
Ou, na impossibilidade de prestar um serviço,

  (untell
     :sender      ASPA
     :receiver    UA
     :in-reply-to id?
     :reply-with id?
     :language    MngmtCL
     :ontology    CAC
     :content     "ServiceReq"))
```

#### 4.4.7.2 Contrato de Prestação de Serviço

Após verificação das operadoras que podem prestar um serviço pretendido, e do conhecimento dos termos do mesmo, um agente UA, na eventualidade de recepção de várias propostas, terá que optar pela que melhor lhe sirva e proceder à assinatura do respectivo contrato de prestação de serviços - **SLA**.

1. **Assinatura do contrato de prestação de serviços, "*SLA*"**

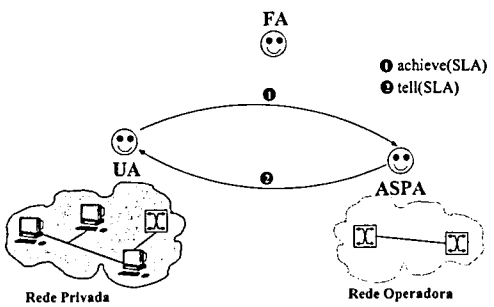



Figura 4.16: Acordo com o "Service Level Agreement" - SLA

Estando de acordo com o *Service Level Agreement* proposto por determinado agente ASPA, um agente UA terá que informar o mesmo que pretende utilizar o serviço por ele proposto. Para o efeito, deverá enviar uma performativa *achieve* utilizando como conteúdo uma expressão **SLA**, pedindo desta forma ao agente ASPA receptor da mensagem que processe o seu conteúdo, isto é, dar a perceber que se vai proceder à assinatura de um contrato de prestação de serviços de ligação. O agente ASPA, após a recepção desta performativa, deverá manter aberto o circuito entretanto reservado, e responder com uma outra performativa (*tell*) utilizando a mesma expressão **SLA**,

informando a efectivação da assinatura de um contrato entre aquelas duas partes e que o circuito se encontra aberto e pronto a ser utilizado.

No cenário a seguir representado (4.14) encontramos as mensagens trocadas entre um agente UA e um agente ASPA, em seguimento do acordo com os termos do contrato de prestação de um serviço de ligação, para a efectivação do referido contrato.

CENÁRIO 4.14 : **Assinatura de um contrato de prestação de serviços**

```
(achieve
   :sender       UA
   :receiver     ASPA
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "SLA")

(tell
   :sender       ASPA
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "SLA"))
```

2. **Negação do contrato de prestação de serviços - SLA**

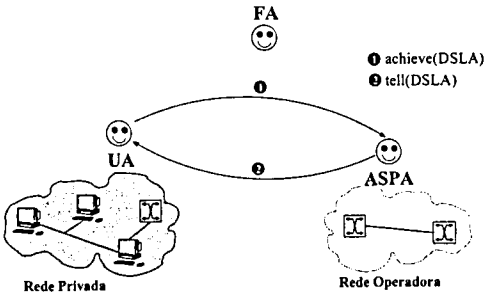



Figura 4.17: Negação do SLA

Contrariamente ao atrás exposto, isto é, caso o agente UA não concorde com os termos do serviço proposto pelos agentes ASPA, ele deverá negar o mesmo, enviando-lhes para o efeito uma performativa *achieve*, idêntica à utilizada no acordo dos termos.

A diferença entre elas reside apenas na expressão passada no conteúdo: é utilizada uma expressão **DSLA** em vez de uma expressão **SLA**. Desta forma, os agentes ASPA que receberem este género de performativas sabem que não existirá qualquer vínculo a um possível contrato de prestação de serviços entre estas duas partes (agentes UA e ASPA), devendo de seguida libertar os recursos afectos aos circuitos entretanto abertos. Por outro lado, os agentes ASPA terão ainda de confirmar a recepção da performativa, respondendo para o efeito com uma performativa *tell(DSLA)*, onde utilizarão a mesma expressão **DSLA**, entretanto recebida.

### 4.4.8 Informação Sobre Interface Utilizada na Rede de Destino

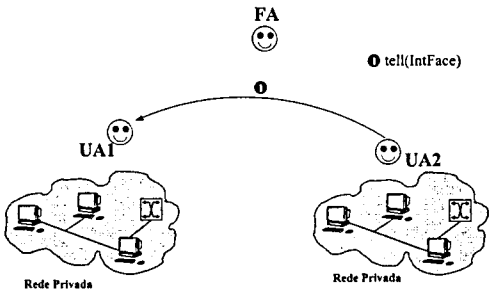



Figura 4.18: Indicação ao agente UA contratante sobre par VCI/VPI

Após a contratação e assinatura de um SLA, o agente UA da rede destinatária deverá detectar que foi criado um circuito de comunicação, tendo que, por um lado, configurar a sua interface de rede utilizando os respectivos pares VPI/VCI, e por outro, após essa configuração, informar o agente UA contratante sobre essa interface. Neste sentido, o agente UA da rede destinatária enviará ao agente UA contratante do serviço uma performativa *tell* onde, no contéudo, passa uma expressão **IntFace**, através da qual ficará a conhecer qual o par VPI/VCI do circuito de comunicação utilizado no lado da rede de destino.

CENÁRIO 4.15 : **Informação sobre interface de rede utilizada pela rede de destino**

```
(tell
    :sender       UA2
    :receiver     UA1
    :in-reply-to  id?
    :reply-with   id?
    :language     MngmtCL
    :ontology     CAC
    :content      "IntFace")
```

A figura 4.18 mostra, de uma forma gráfica, todo o cenário global desta relação de

informação sobre a interface de rede utilizada no outro extremo do circuito.

### 4.4.9 Indicação de Fim de Ligação

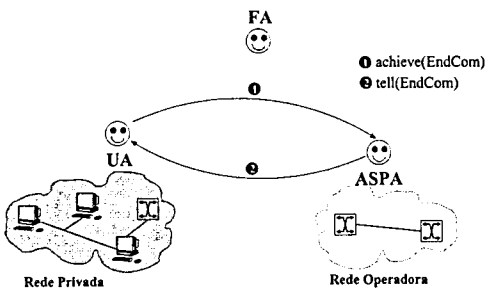



Figura 4.19: Indicação de pretensão de terminar ligação

No caso de não ser utilizada sinalização na comunicação, o agente UA contratante deverá indicar ao agente UA da rede de destino que a ligação vai terminar e pedir-lhe para libertar a interface de rede então por ele utilizada. Assim, o agente UA contratado envia uma performativa *achieve* com uma expressão **EndCom** no conteúdo, indicando-lhe qual o endereço ATM e o par VPI/VCI afecto ao circuuito que vai ser desligado. Em seguimento da recepção desta performativa, o agente UA da rede de destino deverá acusar recepção da mesma e libertar a interface de rede referente àquele circuito criado entre as duas redes. Para acusar a sua recepção, o agente UA da rede de destino enviará uma outra performativa (*tell*) com a mesma expressão **EndCom** no seu conteúdo.

O cenário 4.16 pretende representar, em termos das mensagens KQML trocadas entre estes dois agentes UA, o cenário da indicação de terminação de um serviço de ligação.

CENÁRIO 4.16 : **Indicação de fim de serviço de comunicação**

```
(achieve
   :sender      UA1
   :receiver    UA2
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "EndCom"))

(tell
   :sender      UA2
   :receiver    UA1
   :in-reply-to id?
```

*Continua na página seguinte ...*

**Indicação de fim de serviço de comunicação** *(continuação)*

```
      :reply-with  id?
      :language    MngmtCL
      :ontology    CAC
      :content     "EndCom"))
```

### 4.4.10 Fim do Contrato de Prestação de Serviço

O contrato de prestação de serviços cessará após o tempo acordado expirar, ou por um pedido expresso por um agente UA.

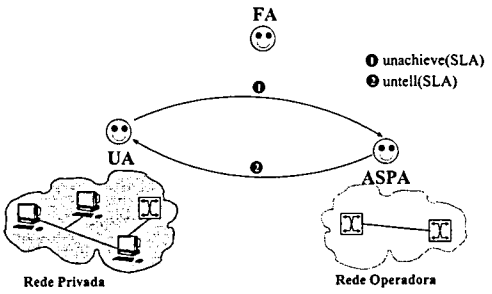



Figura 4.20: Fim do contrato de prestação de serviço

Após o tempo acordado expirar, o agente ASPA contratado deverá enviar uma mensagem ao agente UA contratante, informando-o que o contrato está a chegar ao fim. esta mensagem é formalizada com uma performativa **untell** com a mesma expressão **SLA** utilizado na assinatura do contrato de prestação do serviço, e efectuada por intermédio de uma performativa *achieve(SLA)*. Por outro lado, se o fim de contrato for efectuado a pedido de um agente UA, este terá que informar o agente ASPA acerca dessa sua vontade. Assim, ele deverá enviar-lhe uma performativa *unachieve*, utilizando uma expressão **SLA** recebida no início do estabelecimento do contrato, para que o agente ASPA contratado proceda ao fim do contrato de prestação de serviço. Em resposta, este envia-lhe uma performativa *untell*, indicando, através da utilização da mesma expressão **SLA** recebida, que o contrato cessou (ver cenário 4.17).

CENÁRIO 4.17 : **Pedido de conclusão de prestação de serviços**

```
(unachieve
   :sender      UA
   :receiver    ASPA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
```

*Continua na página seguinte ...*

**Pedido de conclusão de prestação de serviços** *(continuação)*

```
    :content    "SLA"))

  (untell
    :sender      ASPA
    :receiver    UA
    :in-reply-to id?
    :reply-with  id?
    :language    MngmtCL
    :ontology    CAC
    :content    "SLA"))
```

### 4.4.11 Indicação de Mudança de Operadora

Uma outra situação possível de se verificar no funcionamento da arquitectura é a de, a partir de determinado momento, um agente UA contratante pretender que seja outra rede operadora a estabelecer um circuito de ligação entre as mesmas redes privadas (por obviamente envolver custos menores), no entanto sem interromper a comunicação entre elas. Para este fim, o agente UA contratante deve informar primeiro o agente UA da rede de destino que vai mudar de operadora e pedir-lhe para, após o estabelecimento do novo circuito, reconfigurar a sua interface de rede. Com este intuito, o primeiro agente UA envia ao segundo uma performativa *achieve* utilizando como conteúdo uma expressão **ChOper**. Como se poderá constatar no capítulo 5, esta expressão é utilizada para indicar a mudança de operadora no fornecimento de um circuito de comunicação, identificando os pares VPI/VCI a ele afectos. Por seu lado, o agente UA da rede de destino deve responder acusando a recepção dessa performativa, utilizando para o efeito uma outra performativa *tell*, onde utiliza a mesma expressão **ChOper** recebida (cenário 4.18). A partir deste momento, o agente UA da rede de destino estará atento à abertura de um novo circuito devendo, após verificar o estabelecimento do novo circuito, reconfigurar a interface da rede inicial, associando-lhe para o novo par VPI/VCI. Posteriormente, deverá informar uma vez mais o agente UA contratante sobre a nova interface utilizada. Esta informação é efectuada por intermédio do envio de uma performativa *tell* semelhante à indicada no cenário 4.15.

CENÁRIO 4.18 : **Indicação de mudança de operadora**

```
  (achieve
    :sender      UA1
    :receiver    UA2
    :in-reply-to id?
    :reply-with  id?
    :language    MngmtCL
    :ontology    CAC
    :content    "ChOper"))
```

*Continua na página seguinte ...*

**Indicação de mudança de operadora** *(continuação)*

```
(tell
   :sender       UA2
   :receiver     UA1
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content     "ChOper"))
```

### 4.4.12 Subcontratação de Serviços

Se o endereço final do circuito de comunicação pedido por um determinado agente UA for inalcançável devido ao facto da rede de destino se encontrar fora do domínio de um agente ASPA, este terá que perguntar aos seus agentes vizinhos se conseguem estabelecer um circuito de comunicação até esse endereço. Desta forma, o serviço passará a ser subcontratado. Assim, na presente secção descrever-se-á a forma como um agente ASPA procede para subcontratar um serviço de ligação, passando por uma fase inical de verificação de quem lhe poderá prestar melhor o serviço, até à efectivação da subcontratação do mesmo.

As relações entre os agentes ASPA envolvidos no processo global da subcontratação de um serviço de ligação são muito semelhantes às que se encontram entre os agentes UA e ASPA quando os primeiros contratam o mesmo serviço. A diferença reside exclusivamente nos agentes envolvidos (apenas intervêm agentes ASPA). Por este motivo não são aqui apresentados de uma forma exaustiva os correspondentes actos de comunicação.

#### 4.4.12.1 Verificação de Disponibilidade de Prestar Serviços de Ligação

Para saber quem efectivamente pode prestar o serviço de estabelecimento do circuito de comunicação, desde a sua rede até ao endereço final especificado por um agente UA, um agente ASPA envia uma performativa *ask-if* (figura 4.22, legenda 1) aos seus agentes vizinhos, incluindo no seu conteúdo uma expressão **ServiceReq**. Esta expressão possui os mesmos parâmetros recebidos por parte de um agente UA, isto é, a especificação do serviço pretendida e o tempo de ligação desejado. Em resposta, o agente ASPA receptor da performativa (aquele que possivelmente será subcontratado) irá tentar abrir o circuito desejado. Se tal for directamente possível, enviará uma performativa *tell*, com uma expressão **ServiceReq** (figura 4.22, legenda 2) onde estabelece os termos (nova especificação do serviço se necessário, custo e tempo de ligação possível). Por outro lado, se este agente ASPA não conseguir por si atingir o endereço pedido, irá perguntar aos agentes ASPA seus vizinhos se, por sua vez, conseguem estabelecer um circuito de comunicação até ao mesmo endereço, procedendo a outra subcontratação de serviço. Como pode facilmente ser aferido, o processo repetir-se-á até que se consiga atingir o endereço final, ou, por outro lado, se o endereço for inalcançável, proceder-se-á à negação da subcontratação de serviços e,

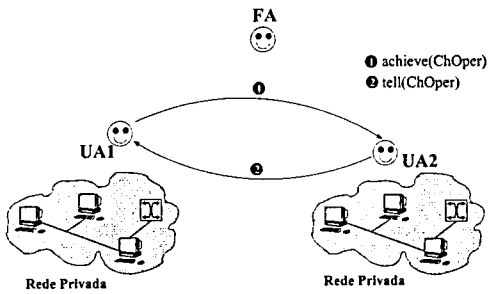



Figura 4.21: Indicação de pretensão de contratação de serviço a outra operadora

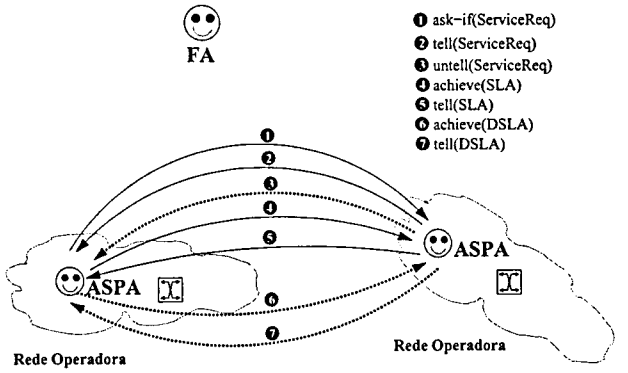



Figura 4.22: Cenário global do pedido de subcontratação de um serviço

consequentemente à informação do agente UA que pediu o serviço sobre a impossibilidade do estabelecimento do mesmo.

Por outro lado, se, por motivos de impossibilidade de alcançar o endereço especificado, ou por questões de comprometimento dos circuitos já estabelecidos, um agente ASPA subcontratado não puder prestar o serviço, ele deverá negar a subcontratação do mesmo, enviando para o efeito, e como resposta, uma performativa *untell*, utilizando no seu conteúdo a mesma expressão **ServiceReq** (figura 4.22, legenda 3), recebido inicialmente.

#### 4.4.12.2 Efectivação da Subcontratação de Serviço

A partir deste momento, um agente ASPA que pretenda uma subcontratação de um serviço, está em condições de saber qual/quais dos seus agentes ASPA vizinhos poderão estabelecer o circuito inicialmente especificado por um agente UA. Fazendo uso da informação recebida na expressão **ServiceReq**, em resposta à performativa *ask-if* inicial, o agente ASPA deverá optar pelo serviço mais barato disponibilizado. Nestas condições, proceder-se-á à assinatura de um contrato de subcontratação de serviços, muito semelhante ao contrato de prestação de serviço - "*Service Level Agreement*" - entre um agente UA e um agente ASPA que a seguir se descreve.

1. **Assinatura do contrato de subcontratação de serviços**
   Estando de acordo com o "*Service Level Agreement*" proposto por um agente ASPA de uma rede sua vizinha, o agente ASPA subcontratante deverá informar o agente subcontratado que pretende utilizar o serviço proposto. Para tal, enviar-lhe-á uma performativa *achieve* utilizando uma expressão **SLA** (figura 4.22, legenda 4), indicando o desejo de assinar um contrato com os termos passados naquela expressão. O agente ASPA proponente dos referidos termos, após a recepção dessa performativa, deverá manter aberto o circuito de comunicação (que entretanto se encontra reservado), e responder com uma outra performativa *tell* fazendo uso da mesma expressão **SLA** (figura 4.22, legenda 5) indicando desta forma que foi assinado um contrato de prestação de serviços e que o circuito pedido se encontra aberto e disponível para utilização.

2. **Negação do contrato de subcontratação de serviços - SLA**
   De forma análoga à negação da prestação de serviço por parte de um agente UA, o agente ASPA que não concorde com os termos de subcontratação propostos, deverá expressar essa sua atitude, negando-os. Para o efeito, deverá enviar uma performativa *achieve* (figura 4.22, legenda 6) utilizando no conteúdo uma expressão **DSLA** (aquela que representa a negação do serviço proposto). Em seguimento da recepção desta última performativa, todos os agentes ASPA seus receptores da mesma deverão responder em conformidade, acusando a sua recepção e informar que não existe vínculo de qualquer espécie entre eles. Esta resposta é efectuada através do envio de uma performativa *tell* com a mesma expressão **DSLA** entretanto recebida (figura 4.22, legenda 7).

A figura 4.22 representa graficamente os actos de comunicação referentes ao cenário global do pedido de subcontratação de um serviço de ligação, encontrando-se os mesmos formalizados em termos da linguagem de conteúdo KQML em anexo nos actos de comunicação (Apêndice A.21 da página 135, Apêndice A.22 da página 135 e Apêndice A.23 da página 136).

### 4.4.13 Fim do Contrato de Subcontratação de Serviço

Tal como o especificado no fim de contrato de prestação de serviço entre agente UA e um agente ASPA, o contrato de subcontratação de serviço cessará quando o tempo acordado expirar, ou na sequência do pedido de fim de contrato de prestação de serviço (efectuado exclusivamente por um agente UA). Dependendo do tipo de ligação estabelecida, isto é, se se utilizou ou não sinalização, o processo de fim de contrato de subcontratação de serviço variará. Assim, e partindo do princípio que não é utilizada sinalização, já que com esta o processo é simples, a relação entre os agentes que a seguir se descreve não existe.

O agente ASPA que quiser terminar a subcontratação de serviço deverá enviar uma performativa *unachieve* utilizando como conteúdo a expressão **SLA** que representa o contrato de prestação de serviço em questão, pedindo desta forma ao agente ASPA subcontratado

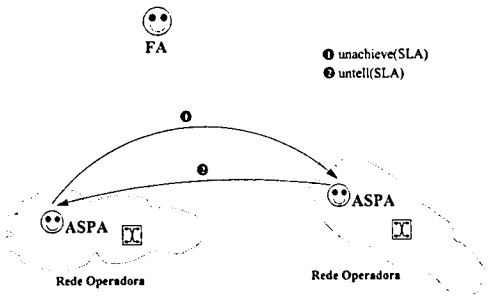



Figura 4.23: Fim do contrato de sub-prestação de serviços

para proceder à libertação do circuito reservado e proceder ao fim da contabilização dos custos envolvidos na subcontratação do serviço. Por outro lado, o agente ASPA subcontratado, após a recepção desta performativa, e após a libertação do respectivo circuito de comunicação, deve enviar ao agente ASPA que o subcontratou uma performativa *untell*, incluindo no seu conteúdo a mesma expressão **SLA** recebida. A partir deste momento, encontramo-nos perante o fim de um contrato de subcontratação de serviços. O agente ASPA contratado por um agente UA (e ao mesmo tempo subcontratante de um serviço) só tem que indicar a esse mesmo agente UA que o circuito de ligação pedido vai ser encerrado (ver o já explicado no cenário 4.17).

## 4.5 Conclusão

Este capítulo apresentou uma arquitectura baseada na tecnologia dos agentes para optimizar a utilização de recursos das redes de telecomunicações quando, através de uma forma automatizada e dinâmica, se estabelecem e concretizam contratos de prestação de serviços de ligação. Os circuitos de comunicação resultantes da prestação destes serviços são estabelecidos entre redes privadas através de uma ou mais redes públicas de área alargada. Inicialmente foi feita uma descrição da arquitectura e apresentados os seus aspectos funcionais. Por fim, foram descritos os cenários de comunicação previstos onde se mostrou como pode ser utilizada a linguagem de comunicação KQML, acente em mecanismos aasíncronos para o transporte de mensagens (por exemplo SMTP, HTML), no envio e recepção dos actos de comunicação dos agentes.

# Capítulo 5

# Linguagem de Conteúdo

## 5.1 Introdução

As linguagens de conteúdo fornecem os aspectos sintáticos para a representação da troca de informação e do conhecimento. Assim, elas são um veículo para a expressão de conhecimento. No entanto, elas devem ser suficientemente abertas para permitirem a representação desse conhecimento. O facto das linguagens deverem ser abertas significa que elas devem aceitar novos vocábulos, isto é, novas palavras, termos e expressões, muitas vezes requeridas na aplicação de novas ontologias. Neste contexto, quando se utiliza uma linguagem de comunicação, deve ser escolhida uma linguagem de conteúdo, embebida nas mensagens trocadas entre os agentes. Uma coisa é possuirmos um meio de transporte dos actos de comunicação (no nosso caso efectuado pelo KQML), e outra é a linguagem de conteúdo, aquela que permite aos agentes trocar o conhecimento, e actuar em concordância com os papéis a eles atribuídos no contexto de determinada arquitectura.

Como o seu nome propriamente indica, uma linguagem de conteúdo é a que representa o conhecimento que os agentes possuem, por um lado através de expressões lógicas, utilizando na maior parte dos casos uma sintaxe semelhante ao LISP (como é o caso do KIF) e, por outro, através de objectos responsáveis pelas acções a serem concretizadas no contexto da mesma.

Assim, é propósito do presente capítulo dar uma visão global da linguagem de conteúdo utilizada na arquitectura, desenvolvida propositadamente para responder às suas necessidades, a qual se nomeou de **MngmtCL** - "*Management Content Language*", passando à especificação dos seus componentes (designados por expressões da linguagem), da sua sintaxe e da sua organização estruturada.

Outro aspecto que vai ser tido em conta é o de que o núcleo central da arquitectura é, como já foi referido, o pedido de informações, a negociação e o estabelecimento de contratos de prestação de serviços de ligação. Portanto, torna-se necessário caracterizar esses serviços, processo que será designado por *especificação de serviço*, ao qual será atribuída uma secção no presente capítulo.

Finalmente, para assimilação e compreensão da linguagem de conteúdo, apresenta-se

um cenário de aplicação real na arquitectura que nos possibilitará, por um lado, verificar na linguagem de comunicação KQML o pedido de informações sobre um serviço e, por outro, como os agentes ASPA utilizam a linguagem de conteúdo para indicar aos agentes FA que estão disponíveis para negociarem e estabelecerm contratos de prestação de serviços de ligação a pedido dos agentes UA.

Um dos contras da utilização das linguagens de conteúdo, obviamente em conjunto com as linguagens de comunicação, é a de não poderem ser excessivamente genéricas, sob o risco de possuirmos uma linguagem extremamente complexa, muito próxima da linguagem natural. Para isso, a linguagem de conteúdo especificada deve ter um âmbito de utilização preciso. É neste sentido que se utilizam ontologias que especificam em que contexto a linguagem é válida, ou seja, que a semântica dos termos utilizados na linguagem tem um valor preciso unicamente nesse contexto de utilização. Deste modo, é definida uma ontologia - **CAC** - que, estando inserida no contexto do estabelecimento de circuitos entre redes privadas através de redes públicas, deve respeitar as funções de gestão do tráfego em redes ATM, definidas pelo ITU-T e o ATM Forum. O seu nome provém da função de controlo de ligação - *"Connection Admission Control - CAC"* [36], segundo a qual, um circuito é estabelecido se se encontrarem disponíveis os recursos suficientes extremo-a-extremo do circuito, e se uma nova ligação não afectar a qualidade de serviço de outras existentes.

## 5.2 Visão Geral da Linguagem

A inter-relação entre os agentes da arquitectura, isto é, o conhecimento das capacidades e papéis a desempenhar por cada um deles, é concretizada, como já foi referido, por intermédio de uma linguagem de comunicação, responsável por encaminhar as crenças, as actividades e os estados dos mesmos agentes. Pode-se até afirmar com segurança que a linguagem de comunicação utilizada - KQML - é o protocolo de comunicação utilizado no contexto da arquitectura, dado não ser mais do que o meio de levar, ou encaminhar correctamente, desde a sua origem até ao seu destino, a informação aos agentes.

No entanto, torna-se imprescindível que tenham uma forma de interpretar o conteúdo de um acto de comunicação, devendo para tal conhecer a forma de o fazer. É neste contexto que uma linguagem de conteúdo desempenha o seu papel, permitindo que os agentes de uma arquitectura saibam efectivamente processar o conteúdo de uma mensagem, e reagir conforme o que se espera com o envio da mesma.

Assim, e como uma linguagem de conteúdo deve ser especificada em concordância com as necessidades de determinada arquitectura ou plataforma multi-agente, implementa-se aqui uma linguagem - "**MngmtCL**" (do inglês "*Management Content Language*") - por forma a que, dentro da gestão global da arquitectura, os agentes correspondam ao que lhes é solicitado, desempenhando as funções dos papéis que lhes foram atribuídos.

Esta linguagem é uma linguagem que utiliza expressões, cada uma delas com uma estrutura bem definida, conforme poderá ser visto na secção 5.3. De momento, a linguagem está especificada para que os agentes possam trocar as informações (formalizadas em termos

de expressões) com o objectivo de negociar e contratar serviços de ligação. Desta forma, ela é constituída por dez componentes principais (ver tabela 5.1), e por uma função - **fcost()** - para possibilitar o cálculo de custos associados a um possível serviço de ligação.

| **Componente** | **Significado** |
|---|---|
| ServiceReq | *"Service Request"* |
| ServiceOffer | *"Service Offer"* |
| SLA | *"Service Level Agreement"* |
| DSLA | *"Deny Service Level Agreement"* |
| CostReq | *"Cost Request"* |
| Neighbourhood | Informação de Agentes vizinhos |
| AServ | *"Accept Service"* |
| IntFace | *"Interface"* |
| ChOper | *"Change Operator"* |
| EndCom | *"End Communication"* |

Tabela 5.1: Sumário dos principais componentes da linguagem de contéudo e seu significado dentro da arquitectura

Os componentes da linguagem resumem-se a expressões híbridas na medida em que são constituídas por outras através de encapsulamento. Assim, podemos distribuí-las segundo uma estrutura, desde o nível de base designado por *nível de objectivo semântico* onde se encontram representados as expressões passadas no conteúdo de um acto de comunicação, até aos subníveis interiores (*nível de especificação de conteúdo semântico*, *nível de qualificação de serviços*, *nível de parametrização de serviços*, *nível de caracterização paramétrica de serviços*, *nível de identificação* e de *QdS* e *nível de classe de serviço*) onde se distribuem todas as restantes. (figura 5.1). A salientar que os níveis *de objectivo semântico*, *de especificação de conteúdo semântico*, *de qualificação de serviços* e *de parametrização de serviços* correspodem aos níveis de base da linguagem, e os restantes aos níveis tecnológicos. A razão da separação das expressões da linguagem segundo estas duas classes de níveis deve-se ao facto de tornar a linguagem de conteúdo mais flexível, tornando-a independente das tecnologias do meio físico que implementarão um serviço desejado. Assim, os níveis de base são aqueles que não dependem das tecnologias, e os níveis tecnológicos são, como a sua designação sugere, muito dependentes das tecnologias/protocolos utilizada(o)s ao nível do transporte físico dos dados.

A um nível superior, isto é, ao da própria linguagem de comunicação entre os agentes, a troca de mensagens é efectuada na seguinte forma: respeitando a especificação do KQML, existe um nome que identifica o tipo de acto de comunicação (**performative**); uma identificação dos agentes intervenientes (**:sender :receiver :from :to**), identificação da sequência dos actos (**:in-reply-to :reply-with**); uma identificação da linguagem de conteúdo - **MngmtCL** e ontologia - **CAC**, com significado na arquitectura proposta (**:language :ontology**) e, finalmente, as expressões da linguagem utilizadas, isto é, o conteúdo

propriamente dito dos actos de comunicação (**:content**). Para melhor compreensão do exposto, atente-se à tabela **Actos de Comunicação** 5.2 a seguir indicada.

ACTOS DE COMUNICAÇÃO 5.2 : **Aspecto geral das performativas trocadas entre os agentes**

```
performative(:sender      <...>
             :from        <...>
             :to          <...>
             :receiver    <...>
             :in-reply-to <...>
             :reply-with  <...>
             :language    MngmtCL
             :ontology    CAC
             :content     "<...>")
```

## 5.3 Expressões da Linguagem e sua Sintaxe

Na presente secção encontram-se descritas pormenorizadamente todas as expressões da linguagem de conteúdo, segundo uma forma estruturada onde se definem determinados níveis e subníveis de acordo com o seu propósito. A relembrar que sintaticamente a linguagem define-se à custa de expressões de base, as quais vam sendo constituídas por outras. Daí a necessidade da existência de níveis e subníveis nas expressões desta linguagem de conteúdo.

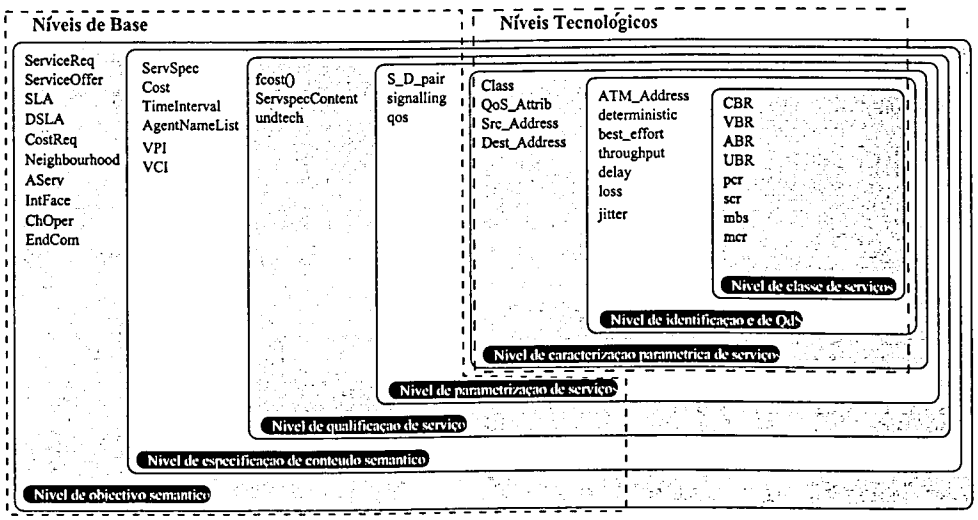



Figura 5.1: Expressões da linguagem de conteúdo e sua estrutura

### 5.3.1 Níveis de Base

As expressões dos níveis de base, são aquelas que são independentes das tecnologias e dos protocolos utilizados ao nível do transporte físico das mensagens e representam o acto primário de um agente. Por este motivo estas expressões são híbridas e, logo, mais complexas. Aqui encontramos quatro subníveis: o primeiro, e o da base da linguagem de conteúdo, é designado de *Nível de objectivo semântico*; o segundo de *Nível de especificação de conteúdo semântico*; o terceiro de *Nível de qualificação de serviços*; e o quarto de *Nível de parametrização de serviços.* Quanto aos níveis dependentes das tecnologias e protocolos, estes são compostos por três subníveis, designados respectivamente por *Nível de caracterização paramétrica, Nível de identificação e de QdS* e, finalmente, *Nível de classe de serviço.*

#### 5.3.1.1 Nível de objectivo semântico

O primeiro nível da linguagem é constituído por todas as expressões utilizadas no conteúdo de um acto de comunicação efectuado pelos agentes da arquitectura. Por este motivo, este nível constitui a base da estrutura das expressões da linguagem de conteúdo. Possuindo cada uma delas um objectivo semântico que correponde ao propósito da sua utilização em determinado acto de comunicação da arquitectura multi-agentes descrita no capítulo 4, de seguida são indicadas cada uma delas expressões, descrevendo-se a sua sintaxe.

1. **ServiceReq**

   A expressão **ServiceReq** é utilizada pelos agentes UA serve, tanto para efectuar um pedido de prestação de serviço (abertura de um circuito de comunicação), como pedir por informações um agente ASPA sobre serviços que são possíveis de fornecer.

   Sendo uma expressão híbrida, como parâmetros admite outras três expressões: uma expressão **ServSpec** que especifica a ligação desejada (utilizadores finais, tipo de circuito, classe de circuito e alguns parâmetros de qualidade de serviço); uma expressão **Cost** que especifica tanto o custo máximo que um agente deseja como o custo que um agente ASPA propõe; e a expressão **TimeInterval** que especifica o tempo pretendido no estabelecimento do circuito. A tabela 5.3 mostra a constituição desta expressão.

```
ServiceReq(ServSpec,
           Cost,
           TimeInterval)
```

Tabela 5.3: Sintaxe da expressão ServiceReq

2. **ServiceOffer**

   A expressão **ServiceOffer** muito semelhante sintaticamente à expressão *ServiceReq* (apenas difere no propósito de utilização na linguagem de conteúdo) é utilizada por um agente ASPA para, por um lado, submeter um contrato de prestação de serviços e, por outro, para fornecer as informações pedidas sobre um ou mais tipos possíveis de prestação dos mesmos serviços.

   Analisando a sintaticamente a expressão, é também constituída por três expressões que a parametrizam: uma expressão **ServSpec** que especifica a ligação desejada (utilizadores finais, tipo de circuito, classe de circuito e alguns parâmetros de qualidade de serviço); uma expressão **Cost** que especifica tanto o custo máximo que um agente deseja, como o custo que um agente ASPA propõe; e uma expressão **TimeInterval** que especifica o tempo pretendido no estabelecimento de um circuito. A tabela 5.4 mostra a constituição desta expressão.

```
ServiceOffer(ServSpec,
             Cost,
             TimeInterval)
```

Tabela 5.4: Sintaxe da expressão ServiceOffer

3. **SLA**

   A expressão **SLA** é utilizada por um agente UA para informar os agentes ASPA que aceita (ou nega) os termos da proposta de um contrato de prestação serviços, por concordar com a informação recebida na sequência da utilização anterior de uma expressão de base **ServiceReq**. Por este motivo, esta expressão corresponde ao chamado *Service Level Agreement*. Tal como a expressão anterior, esta é também constituída pelas três expressões **ServSpec**, **Cost** (sendo esta opcional, é utilizada para passar uma função de cálculo de custos associados ao serviço especificado por **ServiceReq**) e **TimeInterval**. Como se pode verificar na tabela 5.5, onde se mostra a sintaxe da expressão, ela é igual segundo uma visão sintática às expressões **ServiceReq** e **ServiceOffer**, diferindo na finalidade da sua utilização na linguagem de conteúdo.

4. **DSLA**

   A expressão **DSLA** é utilizada por agente UA para informar o ASPA agent que não concorda com os termos por ele propostos e, em consequência, não pretende contratar esse serviço. De uma forma generalisada, pode ser dito que esta expressão é a antítese da expressão **SLA** que anteriormente se descreveu. A sua construção sintática encontra-se representada na tabela 5.6, onde se verifica que os seus

```
SLA(ServSpec,
    Cost,
    TimeInterval)
```

Tabela 5.5: Sintaxe da expressão SLA

parâmetros são idênticos aos dessa expressão **SLA**. Tal como nessa expressão, esta admite também como opcional o parâmetro para a passagem da forma de cálculo de custos (expressão *Cost*) e o parâmetro que define o tempo de ligação pretendido (expressão **TimeInterval**) por poder serem desnecessários, neste contexto.

```
DSLA(ServSpec,
     Cost,
     TimeInterval)
```

Tabela 5.6: Sintaxe da expressão DSLA

5. **CostReq**

   Esta expressão é utilizada por um agente UA para pedir informações sobre o custo envolvido no possível estabelecimento de um circuito de comunicação. Como parâmetro único admite uma expressão **ServSpec** que representa a especificação do serviço sobre a qual é pretendido saber o custo.

```
CostReq(ServSpec)
```

Tabela 5.7: Sintaxe expressão CostReq

6. **Neighbourhood**

   A expressão **Neighbourhood** é utilizada pelos agentes ASPA para informar o agente FA sobre quem são os seus vizinhos. Desta forma, o FA deverá ter uma ideia sobre a topologia da arquitectura dos agentes. Como parâmetros, admite uma lista, (ou sequência), de nomes de agentes ASPA, representada por uma outra expressão **AgentNameList**.

```
Neighbourhood(AgentNameList)
```

Tabela 5.8: Sintaxe da expressão Neighbourhood

7. **AServ**

   A expressão **AServ** é utilizada por um agente UA para informar outro para aceitar um serviço de ligação entre as redes que eles representam. Como parâmetro, a classe encapsula uma expressão **ServSpec**, onde, a título informativo, especifica o circuito que vai ser estabelecido.

   Na tabela 5.9 pode ser encontrada a definição da expressão.

```
AServ(ServSpec)
```

Tabela 5.9: Sintaxe da expressão AServ

8. **IntFace**

   A expressão **IntFace** é utilizada por um agente UA para indicar a outro (o agente contratante) que par VPI/VCI irá utilizar para configurar o circuito de comunicação. Como parâmetros, a classe encapsula outras duas expressões , **VPI** e **VCI**, as quais pretendem representar esse mesmo par.

   Na tabela 5.10 encontra-se definida a sua constituição.

```
IntFace(VPI,VCI)
```

Tabela 5.10: Sintaxe da expressão IntFace

9. **ChOper**

   A expressão **ChOper** é utilizada por um agente UA para pedir a outro (o agente da rede de destino) que aceite a ligação, uma vez que pretende contratar um serviço a um outro agente ASPA, isto é, passar a utilizar outra rede pública, para, após o estabelecimento do novo circuito, reconfigurar convenientemente a interface de rede. Como parâmetros, a expressão é constutuída por outras duas expressões,**VPI** e **VCI**, as quais representam o par inicialmente utilizado e que irá ser afectado devido à utilização de um novo serviço.

   Na tabela 5.11 encontra-se definida a sintaxe da expressão.

```
ChOper(VPI,VCI)
```

Tabela 5.11: Sintaxe da expressão ChOper

10. **EndCom**

    A expressão **EndCom** é utilizada por um agente UA (o agente contratante de um serviço) para informar o agente da rede destino que vai terminar a comunicação e pedir-lhe para libertar a interface de rede utilizada. De notar que esta expressão é apenas utilizada em circuitos de comunicação sem sinalização. Como parâmetros, encapsula duas expressões, **VPI** e **VCI**, que representam, de igual modo às expressões anteriores, o par VPI/VCI da rede de destino em utilização.

    Na tabela 5.12 encontra-se a construção sintática da expressão.

```
EndCom(VPI,VCI)
```

Tabela 5.12: Sintaxe da expressão EndCom

#### 5.3.1.2 Nível de especificação do conteúdo semântico

Neste nível encontramos todas as expressões que constituem as expressões do nível do objectivo semântico, aquelas a partir das quais se implementam todos os actos de comunicação entre os agentes da arquitectura.

1. **Cost**

   A expressão **Cost** é uma expressão constituinte das expressões de base situadas ao nível de objectivo semântico, e é utilizada por um agente ASPA após a recepção de um pedido de informação sobre o custo de um determinado serviço. É através dela que é passada a expressão para o cálculo de um custo associado a um serviço desejado, juntamente com o valor dos respectivos parâmetros. A constituição da expressão vai depender muito das políticas de tarifação utilizadas pelas operadoras e está muito ligado a uma função de cálculo de custos adoptado por determinada operadora *-fcost()* - e está de acordo com as políticas de tarifação adoptadas (ver secção 5.3.3.1).

   No contexto do exemplo de tarifação dado nessa secção, esta expressão é parametrizada através de cinco expressões: **fcost**, **PC**, **PA_Q**, **PCR** e **TimeInterval**. A expressão **fcost** é uma expressão passada por um agente ASPA a um agente UA, e utilizada por este último como função para cálculo do custo associado a um serviço de ligação.

Como apenas a expressão de cálculo é insuficiente, torna-se necessário também passar os valores dos parâmetros nela utilizados, (**PC**, **PA_Q**, **PCR** e **TimeInterval**), que correspondem, respectivamente, a um custo de estabelecimento de uma ligação, ao custo associado à utilização de recursos afectos pela qualidade de serviço (QdS), ao ritmo e pico de células negociado e, finalmente, ao tempo de duração do circuito de comunicação.

A tabela a seguir representada (5.13) pretende mostrar a constituição da expressão **Cost** no contexto da tarifação exemplificada na secção 5.3.3.1.

```
Cost(fcost(),
     PC,
     PA_Q,
     PCR,
     TimeInterval)
```

Tabela 5.13: Sintaxe da expressão Cost

2. **ServSpec**

   Esta expressão é responsável por representar um serviço sobre o qual é pretendido obter informações, ou é pedida a sua negociação e estabelecimento. Por fazer parte da especificação de um serviço, à qual se dedica uma secção mais à frente (secção 5.4), ela não é aqui descrito, mas sim nessa secção.

3. **TimeInterval**

   A expressão **TimeInterval** é uma expressão integrante das expressões de base **ServiceReq**, **ServiceOffer**, **SLA** e **DSLA**, servindo para indicar o parâmetro tempo quer, quando se pedem informações sobre tipos de serviços e custos associados, quer para se submeter, negociar e efectivar propostas de prestação de serviços de ligação no contexto da arquitectura apresentada no capítulo 4.

4. **AgentListName**

   Para que o(s) agente(s) FA possa(m) saber quais os agentes ASPA são vizinhos uns dos outros, isto é, saber como as redes operadoras estão interligadas, torna-se necessário que cada um deles o(s) informem convenientemente neste sentido. Desta forma, sempre que um agente UA lhe(s) pedir para indicar possíveis agentes ASPA para uma eventualidade prestação de um serviço de ligação, eles poderão dar uma resposta conveniente, de acordo com o conhecimento que possuem da topologia geral dos agentes. Assim, a expressão **AgentListName** é utilizada dentro da expressão do nível do objectivo semântico da linguagem - **Neighbourhood** - pelos agentes ASPA

para indicarem ao(s) agente(s) FA quem são os agentes ASPA seus vizinhos. A expressão resume-se a uma cadeia de caracteres alfanuméricos separados por *vírgulas* (",") onde se indicam os nomes de todos esses agentes (veja-se o exemplo indicado na tabela 5.14).

```
AgentNameList = ''ASPA1, ASPA2, ASPA3''
```

Tabela 5.14: Exemplo utilizando uma expressão AgentNameList

5. **VPI**

   A expressão **VPI** é uma expressão utilizada na expressão do nível do objectivo semântico da linguagem - **IntFace** - e representa parte do par VPI/VCI, nomeadamente correspondendo ao chamado *"Virtual Path Identifier"*, necessário para a configuração das interfaces de rede associadas aos circuitos de comunicação.

6. **VCI**

   Tal como uma expressão **VPI**, a expressão **VCI** é utilizada na expressão do nível do objectivo semântico da linguagem - **IntFace** - e representa parte do par VPI/VCI, nomeadamente correspondendo ao chamado *"Virtual Chanel Identifier"*, necessário para a congifuração das interfaces de rede associadas aos circuitos de comunicação.

#### 5.3.1.3 Nível de qualificação de serviços

Neste nível situam-se as expressões que constituem a expressão utilizada na especificação de um serviço (**ServSpec**). Por este motivo, este nível corresponde à qualificação de serviços possíveis de ser prestados pelas redes operadoras. As expressões deste nível correspondem, à definição da tecnologia/protocolo a utilizar ao nível físico do transporte das mensagens entre os agentes (por exemplo ATM, IP, etc.) - expressão **undtech**, e à especificação propriamente dita do serviço - expressão **ServSpecContent**, onde se identificam os endereços das redes entre as quais é pretendido um circuito de comunicação, parâmetros de QdS, e se é pretendida a utilização de sinalização.

Por se ter dedicado uma secção para a especificação de um serviço, estas expressões encontram-se descritas na secção 5.4 deste capítulo.

Para além destas, encontramos também neste nível uma expressão **fcost** que corresponde a uma função de cálculo de custos associados a um serviço. Também aqui não se faz uma sua descrição, devido ao facto de estar relacionada com políticas de tarifação das próprias operadoras. Assim, preferiu-se dar-se uma especial atenção a esta função na secção 5.3.3 deste capítulo.

#### 5.3.1.4 Nível de parametrização de serviços

Neste nível encontramos todas a expressões que parametrizam a expressão **ServSpecContent** do nível de qualificação de serviços. Entre elas encontramos a expressão **S_D_Pair** utilizada para identificar as redes dos extremos do circuito de comunicação, a expressão **signalling** para indicar se é pretendida ou não a utilização de sinalização, e a expressão **qos** para se definirem os parâmeteros da qualidade de serviço. Estas expressões, por constituirem a expressão de base para a definição de um serviço de ligação, encontram-se descritas aprofundamente na secção 5.4 onde, como já foi referido, se faz a introdução ao processo da especificação de um serviço de ligação.

### 5.3.2 Níveis Tecnológicos

Quando se definiram as expressões da linguagem, um dos cuidados que foram tidos foi o de as definir segundo uma estrutura tendo em atenção a sua relação com os próprios actos de comunicação dos agentes. Assim, para além das expressões de base atrás referidas, aquelas que se identificam com o conteúdo das mensagens trocadas, isto é, com os actos de comunicação dos agentes, e aquelas que as prefazem num todo, foi necessário definir outras que, estando muito dependentes das tecnologias do transporte físico das mensagens, se caracterizaram como pertencentes à designação de níveis tecnológicos. Aqui, os níveis tecnológicos são compostos por três subníveis, designados respectivamente por *Nível de caracterização paramétrica*, *Nível de identificação e de QdS* e, finalmente, *Nível de classe de serviço*, que a seguir se descrevem.

#### 5.3.2.1 Nível de caracterização paramétrica de serviços

Neste nível estão definidas as expressões que caracterizam os parâmetros de todo um serviço desejado. Entre ele encontramos as expressões **Class**, **QoS_Attrib**, **Src_Address** e **Dest_Address**, que constituem a globalidade da expressão base para a definição de um serviço. De forma análoga a essa expressão, também estas serão descritas na secção 5.4 onde, se contextualizam e se apresenta a sua sintaxe.

#### 5.3.2.2 Nível de identificação e de QdS

A este nível foram atribuídas as expressões que identificam os extremos de um circuito a ser estabelecido, assim como os parâmetros utilizados na definição de uma qualidade de serviço pretendida. Aqui, encontramos as expressões **ATM_Address** para se definir o endereço ATM dos extremos de um circuito; e **Deterministic**, **Best_Effort**, **throughput**, **delay**, **loss** e **jitter** para se definirem os parâmetros da QdS. Todas estas expressões constituem as expressões do nível de caracterização paramétrica de serviços e, de igual forma a estas últimas, não será aqui efectuada a sua representação sintática, por ela se encontrar especialmente descrita na secção 5.4.

#### 5.3.2.3 Nível de classe de serviço

Este é o último nível da estrutura das expressões da linguagem de conteúdo, e nele encontramos as expressões utilizadas para definir toda uma classe de serviço desejado: que tipo de ligação é desejada, determinística (CBR ou VBR); ou o melhor possível (ABR ou UBR). Para além destas, foram aqui também inseridas outras expressões utilizadas na definição de parâmetros de throughput (pcr, scr, mbs e mcr).

Não fugindo à metodologia optada para a apresentação da linguagem de conteúdo, também a descrição destas expressões se encontra na secção 5.4 deste capítulo.

### 5.3.3 Funções Utilizadas

#### 5.3.3.1 A Função fcost()

Não sendo objecto da presente tese falar sobre tarifação em redes ATM, apenas é apresentada, e a título demonstrativo, uma função possível de ser utilizada, representada pela equação 5.1 (documentação sobre a tarifação em redes ATM pode ser encontrada por exemplo em [37] e em [38]).

Num ambiente onde o importante é a utilização dos recursos, não faz muito sentido utilizar uma tarifação baseada no débito gerado, mas muito mais na reserva e utilização dos próprios recursos da rede. Neste sentido, a função *fcost()* é uma expressão encapsulada na expressão **Cost**, passada por um agente ASPA como conteúdo da linguagem após um pedido de informação sobre custos associados a um serviço de ligação. Uma vez que os custos dependem de vários factores, entre os quais as políticas de tarifação das redes operadoras, torna-se mais fácil um agente ASPA passar a função de cálculo de custos com os devidos parâmetros e, por seu turno, o agente, após a recepção da função, calcular o custo a ele associado.

Uma vantagem inerente a esta forma de procedimento é a de libertar o agente ASPA desta tarefa, ocupando-se apenas das tarefas de abertura (ou tentativa de abertura) e vigilância de circuitos.

$$fcost = PC + PA(Q) * PCR * TimeInterval \tag{5.1}$$

Nesta expressão o parâmetro *PC* representa o custo do estabelecimento de uma ligação; o parâmetro *PA(Q)* o custo associado à utilização de recursos afectos pela qualidade de serviço (QdS); o parâmetro *PCR* ao ritmo das células negociado; e o parâmetro *TimeInterval* à duração do circuito de comunicação.

## 5.4 Especificação de Serviço

Todos os actos de comunicação existentes nesta arquitectura têm o objectivo principal o estabelecimento de um circuito de comunicação entre duas redes operadoras privadas, por intermédio de um agente UA a pedido de um adminitrador dessas redes. Este circuito deverá corresponder às necessidades expressas pelo administrador, nomeadamente a uma

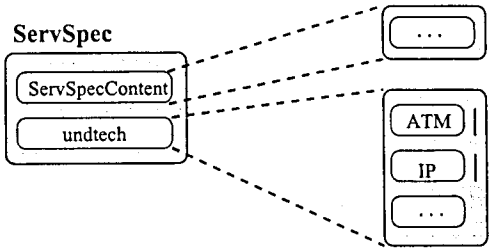



Figura 5.2: Constituição da expressão ServSpec

especificação de serviço por ele definida, ou por propostas apresentadas pelos agentes das redes públicas (agentes ASPA) que estão sujeitas a negociação antes de serem devidamente estabelecidas. Por este motivo, torna-se necessário especificar um serviço, processo esse que é aqui apresentado e explicado. Assim, sempre que um cliente pretenda que uma operadora lhe estabeleça um determinado circuito de comunicação, deve indicar todos os parâmetros necessários (tipo de tecnologia/protocolo, endereço da rede final, tipo e classe de ligação, qualidade de serviço, custo mínimo e tempo de duração da ligação). Neste sentido, é definida na linguagem de conteúdo a expressão **ServSpec**, com o propósito de ser utilizada quer por agentes UA para especificar o serviço pretendido, quer por agentes ASPA para propor um serviço passível de ser negociado com os agentes UA, podendo ou não, resultar na celebração de um contrato de prestação de serviços (*Service Level Agreement - SLA*) entre estas duas entidades.

A especificação de um serviço pode ser vista de uma forma simplificada (figura 5.2), onde se classifica a estrutura de parâmetros utilizada, quer por um agente de uma rede privada (UA), quer por um agente de uma rede operadora (ASPA).

A expressão **ServSpec** possui encapsulada outras duas expressões **ServSpecContent** e **undtech** das quais vão emergir todos os restantes parâmetros para a especificação de um serviço de ligação (tabela 5.15). A expressão **ServSpecContent** representa todos os parâmetros de especificação de um serviço, possuindo formas distintas, consoante se trate de uma tecnologia/protocolo ATM, IP, ou outras. A expressão **undtech** é aquela que representa a tecnologia/protocolo ao nível da camada de transporte a utilizar. Existindo uma série de protocolos e tecnologias, prevê-se nesta arquitectura a utilização do **ATM**, embora a linguagem seja especificada para poder utilizar no futuro outras tecnologias e outros protocolos, como é o exemplo do **IP**.

Como referido anteriormente, a especificação do serviço de ligação vai depender muito da tecnologia e do protocolo utilizado. Por este motivo é definido o parâmetro **undtech** que nos permitirá especificar a tecnologia/protocolo a utilizar. Desta forma, e em conjunção com a correcta parametrização da expressão **ServSpecContent**, consegue-se justificar a estrutura utilizada no componente **ServSpec** com o objectivo de tornar a linguagem mais aberta, dotando-a de capacidade de adaptação a novas realidades tecnológicas que possam vir a surgir, tornando a própria arquitectura multi-agente numa plataforma flexível.

```
ServSpec(ServSpecContent(),
        undtech(ATM |
        IP |
        <...>))
```

Tabela 5.15: Sintaxe da expressão ServSpec

Para se caracterizar um serviço pretendido é necessário, para além de indicar a tecnologia/protocolo a utilizar, especificá-lo por forma a possibilitar, se necessário, a identificação das redes entre as quais deverá ser estabelecido o correspondente circuito e indicar se é pretendida a utilização de sinalização, a classe de serviço e os respectivos parâmetros de QdS (qualidade de serviço).

Neste contexto, define-se a expressão **ServSpecContent**, responsável por caracterizar um serviço de ligação. Esta é também uma expressão híbrida na medida em que é constituída por outras expressões (tabela 5.16): **S_D_pair**, **signalling** e **qos**.

Antes de se proceder à descrição destes parâmetros (ou expressões), torna-se necessário informar que não serão feitas referências de índole mais técnica, isto é, não se pormenorizarão os seus conceitos. Apenas serão utilizados a um nível superior, o da linguagem de conteúdo, para possibilitar que os agentes da arquitectura peçam e forneçam informações, e estabeleçam contratos de prestação de serviço. A sua tecnologia, portanto, abaixo do nível da arquitectura dos agentes sai fora do âmbito do trabalho desenvolvido na presente tese de mestrado.

Passando efectivamente à descrição construtiva dos referidos parâmetros, eis a sua constituição:

O parâmetro **S_D_pair** é uma expressão que pretende identificar o par *"Source-Destination"*, isto é, os endereços das duas redes privadas entre as quais se pretende estabelecer a ligação. Dentro desta, encapsulam-se outras duas, *src_address* e *dest_address* que, como parâmetros, admitem o endereço das duas redes e, opcionalmente, o par VPI/VCI para configuração das interfaces de rede, cada um deles também definidos em termos de uma expressão.

O parâmetro **signalling**, do tipo boleano, é utilizado para indicar se se pretende utilizar ou não, sinalização na ligação.

No que diz respeito à definição da QdS desejada, é definida uma expressão **qos**, constituída por outras duas: **class** e **qos_attrib**. A expressão **class** é utilizada para se definir a classe do serviço de ligação. A expressão **qos_attrib** é utilizada para se definirem os atributos da classe de serviço pretendido. Nesta última encontramos encapsulados outras expressões - **throughput**, *"delay"*, *"loss"* e *"jitter"* - para respectivamente se indicar: parâmetros de **throughput**, *"Maximum Cell Transfer Delay"*, *"Cell Loss Ratio"* e *"Cell Delay Variation"*.

Para definição da classe de serviço, isto é, a sintaxe da expressão **class**, encapsulam-se duas expressões facultativas, onde apenas uma delas é obrigatória, *"deterministic"* e

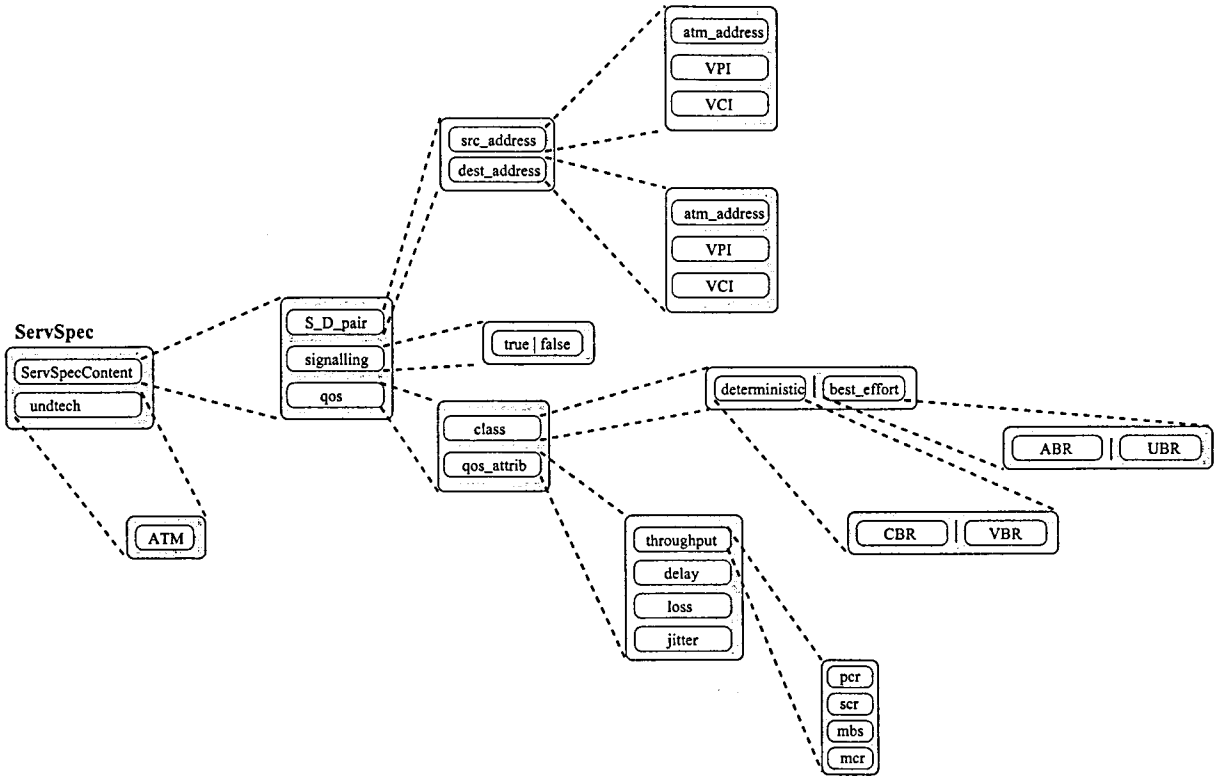



Figura 5.3: Especificação de um serviço

*"best_effort"*, utilizadas para se definir se se pretende utilizar uma ligação determinística, ou a melhor possível.

Como parâmetro único, a expressão *deterministic* pode utilizar uma das expressões que representam o tipo de classe de ligação - **CBR** ou **VBR**. No que diz respeito à expressão *best_effort*, esta admite uma das expressões que representam o tipo de classe de ligação - **ABR** ou **UBR**.

Para a definição dos parâmetros de *throughput*, define-se então a expressão **throughput** que inclui como parâmetros as seguintes expressões: **pcr** - *"Peak Cell Rate"*, **scr** - *"Sustainable Cell Rate"*, **mbs** - *"Maximum Burst Size"* e **mcr** - *"Maximum Cell Rate"*.

Para melhor precepção do exposto, atente-se à tabela 5.16, onde se apresenta a sintaxe geral da expressão **ServSpecContent**.

A figura 5.3 pretende mostrar de uma forma simplificada a constituição da expressão pertencente ao nível do objectivo semântico da linguagem - **ServSpec**, utilizada, como foi referido, para a especificação de um serviço de ligação.

```
ServSpecContent(S_D_pair(src_address(atm_address,
                                     VPI,
                                     VCI),
                         dest_address(atm_address,
                                     VPI,
                                     VCI)),
                signalling(T |
                           F),
                qos(class(deterministic(CBR |
                                        VBR)|
                          best_effort(ABR |
                                      UBR)),
                    qos_attrib(troughput(PCR,
                                         SCR,
                                         MBS,
                                         MCR),
                               delay,
                               loss,
                               jitter)))
```

Tabela 5.16: Especificação do conteúdo de um serviço

## 5.5 Caso de Utilização

Após a especificação da linguagem de conteúdo, e para uma melhor assimilação do que a ela foi referenciado, torna-se evidente a apresentação de um cenário real de funcionamento da arquitectura. Pretende-se com ele, por um lado verificar na prática quais são as expressões da linguagem de conteúdo utilizadas e, por outro, verificar como é utilizada pelos agentes a linguagem de conteúdo, isto é, como é que o KQML, sendo a linguagem de comunicação utilizada, faz o transporte das mensagens trocadas.

Para o efeito, vamos supor que somos os administradores de uma rede privada na qual existe um agente UA. Este agente pretende ainda que o agente facilitator (FA) lhe indique um agente de uma rede pública que esteja em condições de prestar um serviço de ligação. Após a indicação de um agente ASPA, o nosso agente UA entra em contacto com aquele e pede-lhe para enviar informações sobre um determinado serviço (ver figura 5.4). Neste cenário encontramos duas fases, correspondendo a primeira à indicação de um agente de uma rede pública (legendas 1 a 4 da mesma figura) e, a segunda, ao pedido efectuado pelo nosso agente UA ao agente ASPA da rede pública, no sentido de ser informado sobre um serviço de ligação (legendas 5 e 6 da figura). Como estas fases já foram descritas na apresentação da arquitectura no capítulo 4, não se entrará aqui em detalhe na sua explicação. No entanto, convém indicar como são utilizadas as expressões da linguagem

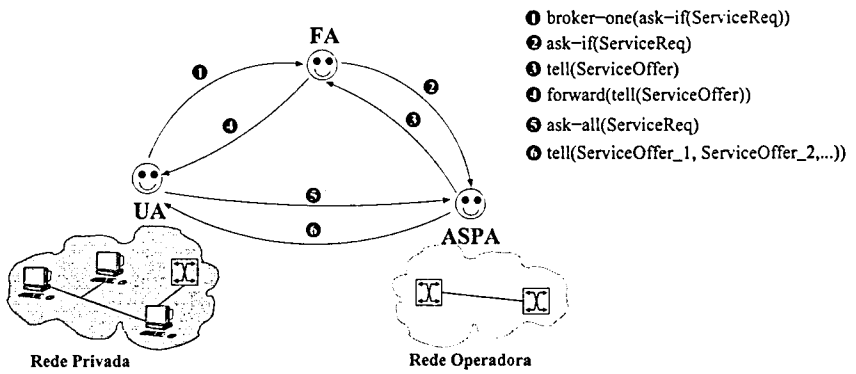



Figura 5.4: Cenário de aplicação real: indicação de uma rede pública e pedido de informações sobre um serviço

no contexto deste cenário real.

O cenário 5.17 abaixo representado pretende indicar as performativas utilizadas pelos agentes neste contexto e as expressões a elas associadas.

TABELA 5.17 : **Pedido de informações sobre serviços - Caso de estudo**

```
(broker-one
   :sender     UA
   :receiver   facilitator
   :reply-with id?
   :language   KQML
   :ontology   kqml-ontology
   :content    (ask-if
                  :sender     UA
                  :reply-with id2
                  :language   MngmtCL
                  :ontology   CAC
                  :content    "ServiceReq"))

(ask-if
   :sender      facilitator
   :receiver    ASPA1
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")
```

*Continua na página seguinte ...*

**Pedido de informações sobre serviços - Caso de estudo** *(continuação)*

```
(tell
   :sender       ASPA1
   :receiver     facilitator
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "ServiceOffer")

(forward
   :from         ASPA1
   :sender       facilitator
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      (tell
                    :receiver    UA
                    :language    MngmtCL
                    :ontology    CAC
                    :content     "ServiceOffer"))
...
(ask-all
   :sender       UA
   :receiver     ASPA
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "ServiceReq")

                    :content     "ServiceOffer"))
...

(tell
   :sender       ASPA
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
```

*Continua na página seguinte ...*

**Pedido de informações sobre serviços - Caso de estudo** *(continuação)*

```
:language    MngmtCL
:ontology    CAC
:content     "ServiceOffer_1, ServiceOffer_2,..., ServiceOffer_N")
```

A expressão utilizada é a expressão **ServiceReq** uma vez que é a que corresponde ao pedido de um serviço, quer para ser negociado e estabelecido, quer para pedir informações. Contudo, a sua constituição vai variar em sequência das fases deste cenário real.

TABELA 5.18 : **Sintaxe da expressão ServiceReq para a fase 1 do cenário real**

```
ServiceReq(ServSpec(ServSpecContent('''',
                                    '''',
                                    '''')),
                    '''',
            '''',
            '''')
```

Como constituição de base, esta expressão é, como já foi indicado na secção das expressões da linguagem e sua sintaxe, constituída por três outras expressões *ServSpec*, *Cost* e *TimeInterval*. Uma vez que esta fase corresponde à fase da indicação de uma rede operadora, não é, portanto, necessária a especificação concreta de um serviço, nem a indicação de custos, assim como nem o tempo desejado da ligação. Assim, a expressão *ServSpec* pode ser constituída como o indicado na tabela (5.18). Nela poderemos verificar que a expressão **Cost** corresponde a uma *string* vazia por não ser necessário aqui defenirem custos. De igaul forma, expressão *TimeInterval* é também uma *string* vazia, por aqui ainda se não definir o tempo da ligação pretendido.

Quanto à especificação do serviço pretendido (formalizado pela expressão *ServSpec*), não se indica a tecnologia pretendia, sendo o parâmetro **undtech** uma *string* vazia e o parâmetro *ServSpecContent* (o que caracteriza o conteúdo do serviço) constituído também por *strings* vazias por não se especificar ainda nesta fase o tipo de serviço. A relembrar que estamos a pedir informações sobre os mesmos.

No que se refere à constituição desta expressão na segunda fase do cenário, torna-se necessário especificar pelo menos alguns parâmetros de serviço, como seja a tecnologia a utilizar, a identificação dos extremos do circuito possível de ser criado (endereços das redes de fonte e de destino), assim como se é pretendido ou não utilizar sinalização. Assim, o agente UA deve utilizar na performativa *ask-all*, (por desejarmos receber informações sobre todos os serviços possíveis de prestar naquele momento), a expressão **ServiceReq** composta da seguinte forma (tabela 5.19): uma expressão *ServSpec* e, possivelmente, a indicação do tempo de ligação pretendido (por exemplo 12 horas); um parâmetro *Cost*, também aqui vazio, por ser o agente da operadora a indicar a forma de cálculo de custos associados ao serviço especificado. Quanto ao serviço sobre que se deseja obter informações, apenas é necessário identificar os seus extremos, isto é, os endereços ATM das respectivas

redes, a tecnologia a utilizar (neste caso ATM), que é pretendido utilizar sinalização e uma ligação determinística CBR para possibilitar, por exemplo, videoconferência.

TABELA 5.19 : **Composição da expressão ServiceReq para o pedido de informação sobre um serviço - fase 2**

```
ServiceReq(ServSpec(ServSpecContent(
                    S_D_pair(src_address(''atm address'',
                                         '''',
                                         '''),
                             dest_address(''atm address'',
                                          '''',
                                          ''')),
                    True,
                    qos(class(deterministic(''CBR'')),
                        qos_attrib(troughput('''',
                                             '''',
                                             '''',
                                             '''),
                                   '''',
                                   '''',
                                   ''')),
                ''ATM'',)
          '''',
          ''12h'')
```

Em resposta à performativa *ask-if* recebida, o agente ASPA deverá recolher todas as informações sobre a possibilidade de prestar este serviço inicial e responder, preenchendo o resto dos campos. Se, por motivo de alocação de recursos existentes, ele não puder fornecer o serviço inicialmente especificado pelo agente UA, ele pode sugerir outros serviços, aproveitando os parâmetros iniciais da expressão **ServiceReq** para utilização na expressão **ServiceOffer**, a qual representa a informação sobre um serviço possível de ser prestado (tabela 5.19). No exemplo demonstrado nesta tabela pode verificar-se que o agente ASPA sugeriu um serviço com duração de 8 horas, e os parâmetros de **througput** - *PCR*, *SCR*, *MBS* e *MCR* - com 5Mb/s cada. Outra particularidade foi a de lhe passar também a forma de cálculo de custos associados a esse serviço, formalizado através da expressão **Cost**.

TABELA 5.20 : **Composição da expressão ServiceOffer para a resposta ao pedido de informação sobre um serviço - fase 2**

```
ServiceOffer(ServSpec(ServSpecContent(
                    S_D_pair(src_address(''atm address'',
                                         '''',
                                         '''),
```

*Continua na página seguinte ...*

**Composição da expressão ServiceOffer para a resposta ao pedido de informação sobre um serviço - fase 2** *(continuação)*

```
                        dest_address(''atm address'',
                                     '''',
                                     ''''))),
                True,
                qos(class(deterministic(''CBR''),
                    qos_attrib(troughput(''5Mb/s'',
                                         ''5Mb/s'',
                                         ''5Mb/s'',
                                         ''5Mb/s''),
                               '''',
                               '''',
                               ''''))),
            ''ATM'',)
        Cost,
        ''8h'')
```

## 5.6 Conclusão

Neste capítulo foi inicialmente apresentada a linguagem de conteúdo especificada para utilização dos agentes na arquitectura multi-agentes apresentada no capítulo 4. Após a visão geral da linguagem, entrou-se mais em detalhe na sua especificação, onde foram descritos pormenorizadamente as suas expressões e a sua localização estrutural. O motivo da classificação estrutural efectuada a essas expressões (níveis de base e níveis tecnológicos) deve-se ao facto de, por um lado existirem expressões de base, transparentes a qualquer tecnologia/protocolo de transporte físico utilizado e, por outro, ao facto de existirem expressões que estão associados as essas tecnologias/protocolos e, portanto, deles dependentes.

Outro factor que pesou nesta definição sintática da linguagem de conteúdo foi a de permitir adaptá-la a qualquer tecnologia, tornando a arquitectura multi-agente mais aberta e transparente.

Sendo o objectivo principal da arquitectura o estabelecimento de circuitos de comunicação, tornou-se necessário descrever como se especifica um serviço, forma esta concretizada por intermédio de uma expressão **ServSpec**.

Finalmente, e para uma melhor compreensão do funcionamento e utilização da linguagem na arquitectura, apresentou-se um cenário de utilização real onde se mostrou, primeiro, como as mensagens entre agentes são passadas ao nível da linguagem de comunicação KQML, depois como se utilizam as expressões da linguagem de conteúdo, nomeadamente as expressões **ServiceReq** e **ServiceOffer** para, respectivamente neste cenário, se pedir e receber informações sobre serviços de ligação.

A especificação da linguagem não deve esgotar aqui, pelo que se torna necessário, para maturidade da própria arquitectura, definir novas expressões que correspondam a novas

tarefas a atribuir aos seus agentes. De momento, a linguagem encontra-se especificada para um funcionamento primitivo da arquitectura, isto é, para a identificação dos agentes, para o pedido de informações sobre determinados serviços de ligação, e custos a eles associados, e para a negociação e estabelecimento de contratos de prestação de serviços de ligação.

# Capítulo 6

# Conclusões e Trabalho Futuro

## 6.1 Conclusões

Neste trabalho é apresentada uma arquitectura multi-agente que se insere no contexto do estabelecimento de circuitos de comunicação entre duas redes privadas, através de redes públicas de área alargada, e pretende, com o recurso à tecnologia tridimensional dos agentes, substituir o factor humano existente no contexto actual das redes de telecomunicações por aplicações autónomas de software.

As linhas seguidas para a resolução do problema proposto, isto é, a definição da arquitectura multi-agente, foram, primeiro o estudo da tecnologia dos agentes inteligentes, dos sistemas existentes, das suas linguagens de comunicação e de conteúdo (capítulos 2 e 3), seguindo-se a apresentação da arquitectura propriamente dita, onde são definidos os papéis dos seus agentes, os aspectos funcionais e os cenários de comunicação previstos (capítulo 4). Finalmente, é especificada uma linguagem de conteúdo que permite aos agentes da arquitectura entenderem-se, reagirem e actuarem de acordo com o propósito dos actos de comunicação por eles efectuados (capítulo 5).

Apesar de a arquitectura ainda se encontrar numa fase inicial de desenvolvimento, sendo ainda um protótipo onde a linguagem de conteúdo possui apenas algumas expressões, e existirem algumas questões que requerem um estudo ainda mais aprofundado (como é o exemplo da segurança), foi possível constatar que as potencialidades oferecidas pelos agentes estão muito adequadas ao contexto das actuais redes de telecomunicações.

Ao nível da linguagem de comunicação KQML, foram previstos os cenários da descoberta de operadoras, os cenários do pedido de informação sobre serviços possíveis de serem prestados, assim como os seus custos. Para além destes, foram bem definidos os cenários de negociação e efectivação de contratos de prestação de serviços (e subprestação, quando existe subcontratação de serviços de uma operadora a outra). Havendo necessidade de informar os agentes dos outros extremos de um circuito de comunicação, que se vai estabelecer um contrato de prestação de serviços, foi previsto um cenário deste género, onde também é pedido para se configurar as interfaces das redes. Quando, por motivos óbvios de redução de custos, um agente UA pretender contratar um serviço a outra operadora,

porém sem interromper o estabelecido previamente, ele deverá informar o outro agente UA de tal facto, proceder à efectivação de um novo *Service Level Agreement*, e pedir a esse agente UA para reconfigurar as interfaces de rede. Só após estes procedimentos poderá dar por concluído o contrato de prestação de serviços com a operadora inicial. Desta forma, não existe quebra no fluxo de dados entre as redes privadas representadas por estes agentes UA.

Quando à linguagem de conteúdo **MngmtCL**, especificada de raíz para utilização na arquitectura multi-agente, foi definido um conjunto de expressões de base por forma a poderem implementar os cenários de comunicação previstos para ela. Estas expressões constituem-se à custa de outras, tendo sido necessário classificá-las quanto aos seus níveis. Assim, definiram-se basicamente dois tipos de níveis que correspondem aos níveis de base, aqueles onde as expressões têm significado de uma forma transparente à utilização das tecnologias/protocolos físicos utilizados na comunicação, e aos níveis tecnológicos que correspondem à utilização de expressões mais específicas por serem muito dependentes dessas mesmas tecnologias e protocolos. Os níveis de base são compostos por quatro subníveis: o primeiro, e o da base da linguagem de conteúdo, é designado de *Nível do objectivo semântico*; o segundo de *Nível de especificação de conteúdo semântico*; o terceiro de *Nível de qualificação de serviços*; e o quarto de *Nível de parametrização de serviços.* Quanto aos níveis dependentes das tecnologias e protocolos, estes são compostos por três subníveis, designados respectivamente por *Nível de caracterização paramétrica*, *Nível de identificação e de QdS* e, finalmente, *Nível de classe de serviço.*

Outro aspecto que foi possível verificar ao longo do trabalho desenvolvido, foi o facto de, apesar de não ser muito recente e não se encontrar normalizada em comparação com o FIPA-ACL, a linguagem de comunicação KQML se adequar perfeitamente às necessidades da arquitectura, permitindo-nos desenvolver de raíz uma linguagem de conteúdo por forma a que os agentes se possam entender e reagir conforme esperado. Não é de mais mencionar que, ao nível das linguagens de comunicação aqui referidas, o KQML foi das primeiras a aparecer, sendo indirectamente responsável pelo aparecimento do FIPA que segue, em traços gerais, a sua filosofia de base.

## 6.2 Trabalho Futuro

O trabalho futuro deverá centrar-se na melhoria da linguagem de conteúdo, adicionando-lhe novas competências à medida que vão surgindo novas redes de telecomunicações (ou amadurecendo as existentes), e no desenvolvimento de outros agentes, atribuindo-lhes diferentes papéis na gestão global do contexto da arquitectura proposta, contribuindo desta forma para a formalização da arquitectura em termos da sua implementação em ambiente real e não académico.

A linguagem de conteúdo **MngmtCL** foi especificada para a tecnologia ATM, tendo, no entanto, sido prevista a utilização de outras tecnologias/protocolos. Assim, torna-se necessário mais algum trabalho para tornar a linguagem neutra e independente das tecnologias e protocolos a utilizar.

Finalmente, o trabalho efectuado na presente tese insere-se no esforço conjunto de um grupo de investigação constituído pelo Professor Doutor Raúl Oliveira que coordenou dois projectos de tese de mestrado (este, e outro realizado pela Licenciada Rita Marques, que estudou a tecnologia ATM, nomeadamente os problemas da sinalização e de selecção de rotas para os circuitos de comunicação, e a configuração do endereçamento IP aos circuitos ATM). Fazendo uma intersecção desses dois trabalhos, torna-se necessário implementar na prática toda a arquitectura multi-agente, desenvolvendo-se esta, por exemplo, a partir das ferramentas de software livre existentes (como é o caso do JAVA e do JKQML), e testar a linguagem de comunicação, não em ambientes simulados como foi o nosso, mas num ambiente real onde encontramos redes privadas ligadas a redes públicas de área alargada.

# Bibliografia


[1] A. Hayzelden and J. Bigham. Agent technology for communications systems: An overview. *Knowledge Engineering Review Journal*, 1999.

[2] Raul Filipe Teixeira de Oliveira. *Managing Awareness in Networks Through Software Agents.* PhD thesis, Ecole Nationale Superieure des Telecomunications, 1998.

[3] Tim Finin, Yannis Labrou, and James Mayfield. Kqml as an agent communication language. In *Software Agents*, chapter 14, pages 291–316. AAAI Press/The MIT Press, 1997.

[4] M Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 1995.

[5] J. P. Müller. Arquitectures and applications of inteligent agents: A survey. *The Knowledge Engineering Review*, 1998.

[6] Yannis Labrou and Tim Finin. A proposal for a new kqml specification. Technical report, Computer Science and Electrical Engineering Department(CSEE), University of Maryland Baltimore County(UMBC), February 1997.

[7] Michael R. Genesereth and Richard E. Fikes. *Knowledge Interchange Format, Version 3.0, Reference Manual.* Logic Group, Computer Science Department., Stanford University, January 1992.

[8] Alex L. G. Hayzelden and John Bigham. Future communication networks using software agents. In Alex L. G. Hayzelden and John Bigha, editors, *Software Agents for Future Communication Systems*, chapter 1, pages 1–57. Springer, March 1999.

[9] Decina M. and Trecordi V. Convergence of telecommunications and computing to networking models for integrated services and applications. In *IEEE*, volume 85, number 12, December 1997.

[10] Goldszmidt G. and Yemini U. Delegated agents for network management. *IEEE Communications Magazine*, 36, number 3, March 1998.

[11] Michael Wooldridge and Nicholas R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[12] Anand S. Rao and Michael P. Georgeff. Bdi agents: From theory to practice. In *First Intl. Conference on Multiagent Systems (ICMAS-95)*, pages 321–319, San Francisco,CA, June 1995.

[13] N. R. Jennings et al. Grate: A general framework for cooperative problem solving. *IEE-BCS Journal of Intelligent Systems Engineering*, (1 (2)):102–114, 1992.

[14] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Sommers, and R. Evans. Software agents: A review. Technical report, Trinity College, University of Dublin, June 1997. Available at http://www.cs.tcd.ie/research_groups/aig/iag.

[15] British Telecom. *Software Agents for ATM performance management.* NOM'98, 1998.

[16] Foundation For Intelligent Physical Agents. *FIPA 97 Specification - Part 7: Network Management and Provisioning.* Geneva, Switzerland, version 1.0 part7 edition, 1997. http://www.fipa.org.

[17] Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Communications of the ACM*, 37(7), July 1994.

[18] Klaus Fischer, Jorg P. Muller, and Markus Pischel. A pragmatic bdi architecture. In Michael Wooldridge, Jorg P. Muller, , and Milind Tambe, editors, *Intelligent Agents II - Agent Theories, Architectures, and Languages*, pages 203–218. Springer, 1995.

[19] R.S. Patil et al. The darpa knowledge sharing effort: Progress report, 1997.

[20] Tim Finin, Jay Weber, Gio Wiederhold, Michael Genesereth, Richard Fritzon, james McGuire Stuart Shapiro, and Chris Beck. *Draft Specification of the KQML Agent-Communication Language.* The DARPA Knowledge Sharing Initiative, External Interfaces Working Group, June 1993.

[21] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 2:199–220, 1993.

[22] Foundation for Intelligent Physical Agents. *Specifications, 1997.* http://www.fipa.org.

[23] Foundation For Intelligent Physical Agents. *FIPA 97 Specification - Part 2: Agent Communication Language.* Geneva, Switzerland, version 1.0 part2 edition, 1997. http://www.fipa.org.

[24] Y. Labrou. *Semantics for an Agent Communication Language.* PhD thesis, Computer Science and Electrical Engineering Dept., Univ. of Maryland, Baltimore County, 1996.

[25] Yanis Labrou, Tim Finin, and Yun Peng. Agent communication languages: The current landscape. *IEEE Intelligent Systems*, pages 45–51, MARCH/APRIL 1999.

[26] Michael R. Genesereth. An agent-based framework for interoperability. In editor Jeffrey M. Bradshaw, editor, *Software Agents*, pages chapter 15, pages 317–345. Aaai press/mit press edition, 1997.

[27] Michael R. Genesereth. *Knowledge Interchange Format, draft proposed American National Standard (dpANS).* NCITS.T2/98-004, Stanford University, 1998.

[28] Raul Filipe Teixeira de Oliveira. *Managing Awareness in Networks Through Software Agents.* PhD thesis, Ecole Nationale Superieure des Telecomunications, 1998.

[29] Morsy M. Cheikhrouhou, Pierre Conti, and Karina Marcus. Software agents or network management: Case studies and experience-gained. Technical report, Institut Eurécom, Corporate Communications Dept., 1999.

[30] ATM Forum. Atm user-network interface (uni) signalling specification version 4.0. Technical report, ATM Forum, 1996.

[31] ATM Forum. Private network-network interface specification version 1.0 (pnni 1.0). Technical report, ATM Forum, 1996.

[32] Uyless Black. *Signalling in broadband networks.* Prentice-Hall, 1998.

[33] David Ginsburg. *ATM solutions for enterprise internetworking.* Addison-Wesley, 1999.

[34] Manfred Huber Handel and Stefan Schroder. *ATM Networks: Concepts, Protocols and Apllications.* Addison-Wesley, 1998.

[35] Hajime Tsuchitani. *JKQML, an Java-based KQML API.* IBM Alpha Works, http://alphaworks.ibm.com, March 1999.

[36] Gerald P. Ryan. Atm traffic management. Technical report, Applied Technologies Group, http://www.techguide.com, 1997.

[37] CANCAN. Contract negotiation and charging in atm networks. Final Report AC041/DCU/P1/1/OJ1/FR-A7, September 1998.

[38] CA$HMAN. Charging and accounting schemes in multi-service networks. Technical report, http://www.infowin.org/ACTS/RUS/PROJECTS/ac039.htm.

# Apêndice A

# Actos de Comunicação

## A.1 Relações entre agentes UA, ASPA e FA

### A.1.1 Determinação de Operadoras

ACTOS DE COMUNICAÇÃO A.1 : **Utilizando performativas** *broker-all*

```
(broker-all
   :sender     UA
   :receiver   facilitator
   :reply-with id?
   :language   KQML
   :ontology   kqml-ontology
   :content    (ask-if
                  :sender     UA
                  :reply-with id2
                  :language   MngmtCL
                  :ontology   CAC
                  :content    "ServiceReq"))

(ask-if
   :sender      facilitator
   :receiver    ASPA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")

(tell
```

*Continua na página seguinte ...*

**Utilizando performativas** *broker-all (continuação)*

```
   :sender       ASPA
   :receiver     facilitator
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "ServiceOffer")

(forward
   :from         ASPA
   :sender       facilitator
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      (tell
                    :receiver    UA
                    :language    MngmtCL
                    :ontology    CAC
                    :content     "ServiceOffer"))

(untell
   :sender       ASPA
   :receiver     facilitator
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "ServiceReq")

(forward
   :from         ASPA
   :sender       facilitator
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      (untell
```

*Continua na página seguinte ...*

**Utilizando performativas** *broker-all (continuação)*

```
                    :receiver    UA
                    :language    MngmtCL
                    :ontology    CAC
                    :content     "ServiceReq"))
```

ACTOS DE COMUNICAÇÃO A.2 : **Utilizando performativas** *broker-one*

```
(broker-one
   :sender     UA
   :receiver   facilitator
   :reply-with id?
   :language   KQML
   :ontology   kqml-ontology
   :content    (ask-if
                  :sender     UA
                  :reply-with id2
                  :language   MngmtCL
                  :ontology   CAC
                  :content    "ServiceReq"))

(ask-if
   :sender      facilitator
   :receiver    ASPA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")

(tell
   :sender      ASPA
   :receiver    facilitator
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")

(forward
   :from        ASPA
   :sender      facilitator
```

*Continua na página seguinte ...*

**Utilizando performativas** *broker-one (continuação)*

```
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (tell
                   :receiver    UA
                   :language    MngmtCL
                   :ontology    CAC
                   :content     "ServiceReq"))

(untell
   :sender      ASPA
   :receiver    facilitator
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")

(forward
   :from        ASPA
   :sender      facilitator
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (deny
                   :receiver    UA
                   :language    MngmtCL
                   :ontology    CAC
                   :content     "ServiceReq"))
```

ACTOS DE COMUNICAÇÃO A.3 : **Utilizando performativas** *recomend-one*

```
(recommend-one
   :sender      UA
   :receiver    facilitator
   :in-reply-to id?
   :reply-with  id?
```

*Continua na página seguinte ...*

**Utilizando performativas** *recomend-one (continuação)*

```
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
                   :sender    UA
                   :language    MngmtCL
                   :ontology    CAC
                   :content     "ServiceReq"))

(forward
   :from        ASPA
   :to          UA
   :sender      facilitator
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (advertise
                   :sender     ASPA
                   :receiver   UA
                   :reply-with id?
                   :language   KQML
                   :ontology   kqml-ontology
                   :content    (ask-if
                                  :sender      UA
                                  :receiver    ASPA
                                  :in-reply-to id?
                                  :language    MngmtCL
                                  :ontology    CAC
                                  :content     "ServiceReq")))
```

ACTOS DE COMUNICAÇÃO A.4 : **Utilizando performativas** *recomend-all*

```
(recommend-all
   :sender      UA
   :receiver    facilitator
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
```

*Continua na página seguinte ...*

**Utilizando performativas** *recomend-all (continuação)*

```
                    :sender     UA
                    :language   MngmtCL
                    :ontology   CAC
                    :content    "ServiceReq"))

(forward
   :from         ASPA1
   :to           UA
   :sender       facilitator
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      (advertise
                    :sender     ASPA1
                    :receiver   UA
                    :reply-with id?
                    :language   KQML
                    :ontology   kqml-ontology
                    :content    (ask-if
                                   :sender      UA
                                   :receiver    ASPA1
                                   :in-reply-to id?
                                   :language    MngmtCL
                                   :ontology    CAC
                                   :content     "ServiceReq")))

...

(forward
   :from         ASPA_n
   :to           UA
   :sender       facilitator
   :receiver     UA
   :in-reply-to  id?
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      (advertise
```

*Continua na página seguinte ...*

**Utilizando performativas** *recomend-all (continuação)*

```
                    :sender     ASPA_n
                    :receiver   UA
                    :reply-with id?
                    :language   KQML
                    :ontology   kqml-ontology
                    :content    (ask-if
                                   :sender      UA
                                   :receiver    ASPA_n
                                   :in-reply-to id?
                                   :language    MngmtCL
                                   :ontology    CAC
                                   :content     "ServiceReq"))
```

ACTOS DE COMUNICAÇÃO A.5 : **Utilizando performativas** *recruit-one*

```
(recruit-one
   :sender      UA
   :receiver    facilitator
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
                   :sender    UA
                   :language  MngmtCL
                   :ontology  CAC
                   :content   "ServiceReq"))

(forward
   :from        UA
   :to          ASPA
   :sender      facilitator
   :receiver    ASPA
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
                   :sender      UA
                   :receiver    ASPA
                   :in-reply-to id?
                   :reply-with  id?
```

*Continua na página seguinte ...*

**Utilizando performativas** *recruit-one (continuação)*

```
                    :language    MngmtCL
                    :ontology    CAC
                    :content     (ServiceReq)))
```

ACTOS DE COMUNICAÇÃO A.6 : **Utilizando performativas** *recruit-all*

```
(recruit-all
   :sender      UA
   :receiver    facilitator
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
                   :sender    UA
                   :language  MngmtCL
                   :ontology  CAC
                   :content   "ServiceReq"))

(forward
   :from        UA
   :to          ASPA1
   :sender      facilitator
   :receiver    ASPA1
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content     (ask-if
                   :sender      UA
                   :receiver    ASPA1
                   :in-reply-to id?
                   :reply-with  id?
                   :language    MngmtCL
                   :ontology    CAC
                   :content     (ServiceReq)))

 ...

(forward
   :from        UA
   :to          ASPA_n
```

*Continua na página seguinte ...*

**Utilizando performativas** *recruit-all (continuação)*

```
:sender      facilitator
:receiver    ASPA_n
:reply-with  id?
:language    KQML
:ontology    kqml-ontology
:content     (ask-if
                :sender      UA
                :receiver    ASPA_n
                :in-reply-to id?
                :reply-with  id?
                :language    MngmtCL
                :ontology    CAC
                :content     (ServiceReq)))
```

### A.1.2 Pedido de Informações Sobre Serviços

ACTOS DE COMUNICAÇÃO A.7 : **Utilizando performativas** *ask-all*

```
(ask-all
   :sender      UA
   :receiver    ASPA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "[ServiceOffer_1,ServiceOffer_2]")
```

ACTOS DE COMUNICAÇÃO A.8 : **Utilizando performativas** *stream-all*

```
(stream-all
   :sender      UA
   :receiver    ASPA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer_1")
```

*Continua na página seguinte ...*

**Utilizando performativas** *stream-all (continuação)*

```
(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer_2")

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer_3")
```

ACTOS DE COMUNICAÇÃO A.9 : **Utilizando performativas** *subscribe*

```
(subscribe
   :sender      UA
   :receiver    ASPA
   :in-reply-to id?
   :reply-with  id?
   :language    KQML
   :ontology    kqml-ontology
   :content
      (ask-all
         :sender      UA
         :receiver    ASPA
         :in-reply-to id?
         :reply-with  id?
         :language    MngmtCL
         :ontology    CAC
         :content     "ServiceReq"))

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
```

*Continua na página seguinte ...*

**Utilizando performativas** *subscribe (continuação)*

```
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "[ServiceOffer_1,ServiceOffer_2]")

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "[ServiceOffer_3]")
```

### A.1.3 Pedido de Estabelecimento de circuitos

ACTOS DE COMUNICAÇÃO A.10 : **Verificação da Possibilidade de Prestação de Serviço**

```
(ask-if
   :sender      UA
   :receiver    ASPA
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")

(tell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer")

(untell
   :sender      ASPA
   :receiver    UA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")
```

ACTOS DE COMUNICAÇÃO A.11 : **Contrato de Prestação de Serviço - Assinatura SLA**

```
(achieve
   :sender      UA
   :receiver    ASPA
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "SLA")
```

*Continua na página seguinte ...*

**Contrato de Prestação de Serviço - Assinatura SLA** *(continuação)*

```
(tell
    :sender       ASPA
    :receiver     UA
    :in-reply-to  id?
    :language     MngmtCL
    :ontology     CAC
    :content      "SLA")
```

ACTOS DE COMUNICAÇÃO A.12 : **Contrato de Prestação de Serviço - Negação SLA**

```
(achieve
   :sender       UA
   :receiver     ASPA
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "DSLA")

(tell
    :sender       ASPA
    :receiver     UA
    :in-reply-to  id?
    :reply-with   id?
    :language     MngmtCL
    :ontology     CAC
    :content      "DSLA")
```

ACTOS DE COMUNICAÇÃO A.13 : **Fim do Contrato de Prestação de Serviço**

```
(unachieve
   :sender       UA
   :receiver     ASPA
   :in-reply-to  id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "SLA")
```

*Continua na página seguinte ...*

**Fim do Contrato de Prestação de Serviço** *(continuação)*

```
(untell
    :sender       ASPA
    :receiver     UA
    :in-reply-to  id?
    :reply-with   id?
    :language     MngmtCL
    :ontology     CAC
    :content     "SLA")
```

### A.1.4 Pedido de Informações Sobre Custos

ACTOS DE COMUNICAÇÃO A.14 : **Pedido de Informações Sobre Custos**

```
(ask-if
   :sender      UA
   :receiver    ASPA
   :reply-with id?
   :language    MngmtCL
   :ontology    CAC
   :content     "CostReq")

(tell
   :sender       ASPA
   :receiver     UA
   :in-reply-to id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "Cost")

(untell
   :sender       ASPA
   :receiver     UA
   :in-reply-to id?
   :reply-with   id?
   :language     MngmtCL
   :ontology     CAC
   :content      "CostReq")
```

## A.2 Relações exclusivas entre agentes UA

### A.2.1 Pretensão de Estabelecimento de Circuito

ACTOS DE COMUNICAÇÃO A.15 : **Pretensão de Estabelecimento de Circuito**

```
(achieve
    :sender      UA1
    :receiver    UA2
    :in-reply-to id?
    :reply-with  id?
    :language    MngmtCL
    :ontology    CAC
    :content     "AServ")

(tell
    :sender      UA2
    :receiver    UA1
    :in-reply-to id?
    :reply-with  id?
    :language    MngmtCL
    :ontology    CAC
    :content     "AServ")

(tell
    :sender      UA2
    :receiver    UA1
    :in-reply-to id?
    :reply-with  id?
    :language    MngmtCL
    :ontology    CAC
    :content     "IntFace")
```

### A.2.2 Indicação de Mudança de Operadora

ACTOS DE COMUNICAÇÃO A.16 : **Indicação de Mudança de Operadora**

```
(achieve
    :sender       UA1
    :receiver     UA2
    :in-reply-to  id?
    :reply-with   id?
    :language     MngmtCL
    :ontology     CAC
    :content      "ChOper")

(tell
    :sender       UA2
    :receiver     UA1
    :in-reply-to  id?
    :reply-with   id?
    :language     MngmtCL
    :ontology     CAC
    :content      "ChOper")
```

### A.2.3 Indicação de Fim de Ligação

ACTOS DE COMUNICAÇÃO A.17 : **Indicação de Fim de Ligação**

```
(achieve
    :sender      UA1
    :receiver    UA2
    :in-reply-to id?
    :reply-with  id?
    :language    MngmtCL
    :ontology    CAC
    :content     "EndCom")

(tell
    :sender      UA2
    :receiver    UA1
    :in-reply-to id?
    :reply-with  id?
    :language    MngmtCL
    :ontology    CAC
    :content     "EndCom")
```

## A.3 Relações entre agentes ASPA e FA

### A.3.1 Informação Sobre Topologia de agentes ASPA

ACTOS DE COMUNICAÇÃO A.18 : **Informação Sobre Topologia de agentes ASPA**

```
(advertise
   :sender       ASPA
   :receiver     facilitator
   :reply-with   id?
   :language     KQML
   :ontology     kqml-ontology
   :content      "Neighbourhood"
```

### A.3.2 Indicação de Operadora Candidata a Prestação de Serviços

ACTOS DE COMUNICAÇÃO A.19 : **Indicação de Operadora Candidata a Prestação de Serviços**

```
(advertise
   :sender     ASPA
   :receiver   facilitator
   :reply-with id?
   :language   KQML
   :ontology   kqml-ontology
   :content    (ask-if
                  :reply-with id?
                  :language   MngmtCL
                  :ontology   CAC
                  :content    "ServiceReq"))
```

### A.3.3 Indicação de Indisponibilidade de Prestação de Serviços

ACTOS DE COMUNICAÇÃO A.20 : **Indicação de Indisponibilidade de Prestação de Serviços**

```
(unadvertise
   :sender     ASPA
   :receiver   facilitator
   :reply-with id?
   :language   KQML
   :ontology   kqml-ontology
   :content    (ask-if
                  :reply-with id?
                  :language   MngmtCL
                  :ontology   CAC
                  :content    "ServiceReq"))
```

### A.3.4 Subcontratação de Serviços

ACTOS DE COMUNICAÇÃO A.21 : **Verificação da Possibilidade de Prestação de Serviço**

```
(ask-if
   :sender      ASPA1
   :receiver    ASPA2
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")

(tell
   :sender      ASPA2
   :receiver    ASPA1
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceOffer")

(untell
   :sender      ASPA2
   :receiver    ASPA1
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "ServiceReq")
```

ACTOS DE COMUNICAÇÃO A.22 : **Efectivação da Subcontratação de Serviço**

```
(achieve
   :sender      ASPA1
   :receiver    ASPA2
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "SLA")
```

*Continua na página seguinte ...*

**Efectivação da Subcontratação de Serviço** *(continuação)*

```
(tell
    :sender      ASPA2
    :receiver    ASPA1
    :in-reply-to id?
    :language    MngmtCL
    :ontology    CAC
    :content     "SLA")
```

ACTOS DE COMUNICAÇÃO A.23 : **Negação da Subcontratação de Serviço**

```
(achieve
   :sender      ASPA1
   :receiver    ASPA2
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "DSLA")

(tell
    :sender      ASPA2
    :receiver    ASPA1
    :in-reply-to id?
    :reply-with  id?
    :language    MngmtCL
    :ontology    CAC
    :content     "DSL")
```

ACTOS DE COMUNICAÇÃO A.24 : **Fim do Contrato de Subcontratação de Serviço**

```
(unachieve
   :sender      ASPA1
   :receiver    ASPA2
   :in-reply-to id?
   :reply-with  id?
   :language    MngmtCL
   :ontology    CAC
   :content     "SLA")

(untell
    :sender      ASPA2
```

*Continua na página seguinte ...*

**Fim do Contrato de Subcontratação de Serviço** *(continuação)*

```
    :receiver     ASPA1
    :in-reply-to  id?
    :reply-with   id?
    :language     MngmtCL
    :ontology     CAC
    :content      "SLA")
```

# Apêndice B

# Especificação da Linguagem de Conteúdo

## B.1 Expressões da Linguagem MngmtCL Segundo o Formato BNF

EXPRESSÃO B.1 : **ServiceReq**

```
<ServiceReq>   ::= (<ServSpec>, <Cost>, <TimeInterval>)
<Cost>         ::= <string>
<TimeInterval> ::= <string>
```

EXPRESSÃO B.2 : **ServiceOffer**

```
<ServiceOffer>   ::= (<ServSpec>, <Cost>, <TimeInterval>)
<Cost>         ::= <string>
<TimeInterval> ::= <string>
```

EXPRESSÃO B.3 : **SLA**

```
<SLA>          ::= (<ServSpec>, Ccost>, <TimeInterval>)
<Cost>         ::= <string>
<TimeInterval> ::= <string>
```

EXPRESSÃO B.4 : **DSLA**

```
<DSLA>         ::= (<ServSpec>, <Cost>, <TimeInterval>)
<Cost>         ::= <string>
<TimeInterval> ::= <string>
```

EXPRESSÃO B.5 : **CostReq**

```
<CostReq> :: = (<ServSpec>)
```

EXPRESSÃO B.6 : **Cost**

```
<Cost>         ::= (<fcost>,<PC>,<PA_Q>, <PCR>, <TimeInterval>)
<fcost>        ::= <string>
<PC>           ::= <string>
<PA_Q>         ::= <string>
<PCR>          ::= <string>
<TimeInterval> ::= <string>
```

EXPRESSÃO B.7 : **Neighbourhood**

```
<Neighbourhood> ::= <AgentNameList>
<AgentNameList> ::= <string*>
```

EXPRESSÃO B.8 : **AServ**

```
<AServ> ::= <ServSpec>
```

EXPRESSÃO B.9 : **IntFace**

```
<IntFace> ::= (<VPI>,<VCI>)
<VPI>     ::= <string>
<VCI>     ::= <string>
```

EXPRESSÃO B.10 : **ChOper**

```
<ChOper> ::= (<VPI>,<VCI>)
<VPI>    ::= <string>
<VCI>    ::= <string>
```

EXPRESSÃO B.11 : **EndCom**

```
<EndCom> ::= (<VPI>,<VCI>)
<VPI>    ::= <string>
<VCI>    ::= <string>
```

## B.2 Especificação Completa

### B.2.1 Especificação de um Serviço para ATM no Formato BNF

SERVSPEC B.12 : **Especificação de Serviço para ATM**

```
<ServSpec>          ::= (<ServSpecContent> , <undtech>)
<undtech>           ::= <ATM>
<ATM>               ::= <string>
<ServSpecContent>   ::= (<S_D_pair>, <signalling>, <qos>)
<S_D_pair>          ::= (<src_address>, <dest_address>)
<src_address>       ::= (<atm_address>, <VPI>, <VCI>)
<dest_address>      ::= (<atm_address>, <VPI>, <VCI>)
<atm_address>       ::= <string>
<VPI>               ::= <string>
<VCI>               ::= <string>
<signalling>        ::= <bool>
<qos>               ::= (<class>, <qos_attrib>)
<class>             ::= <deterministic>
                      | <best_effort>
<deterministic>     ::= <CBR>
                      | <VBR>
<CBR>               ::= <string>
<VBR>               ::= <string>
<best_effort>       ::= <ABR>
                      | <UBR>
<ABR>               ::= <string>
<UBR>               ::= <string>
<qos_attrib>        ::= (<throughput>, <delay>, <loss>, <jitter>)
<throughput>        ::= (<pcr>, <scr>, <mbs>, <mcr>)
<delay>             ::= <string>
<loss>              ::= <string>
<jitter>            ::= <string>
<pcr>               ::= <string>
<scr>               ::= <string>
<mbs>               ::= <string>
<mcr>               ::= <string>
<string>            ::= "<stringchar>*"
                      | #<digit><digit>*"<ascii>*
<stringchar>        ::= \<ascii>
                      | <ascii>-\-<double-quote>
<bool>              ::= <T>
```

*Continua na página seguinte ...*

**Especificação de Serviço para ATM** *(continuação)*

```
                    | <F>
<T>                 ::= 1
<F>                 ::= 0
```

### B.2.2 Especificação de um Serviço para IP no Formato BNF

SERVSPEC B.13 : **Especificação de Serviço para IP**

```
<ServSpec>          ::= (<ServSpecContent> , <undtech>)
<undtech>           ::= <IP>
<IP>                ::= <string>
<ServSpecContent>   ::= (<S_D_pair>, <qos>)
<S_D_pair>          ::= (<src_address>, <dest_address>)
<src_address>       ::= <string>
<dest_address>      ::= <string>
<qos>               ::= (<RSVP>, <qos_attrib>)
<RSVP>              ::= <best_effort>
                      | <rate_sensitive>
                      | <delay_sensitive>
<best_effort>       ::= <string>
<rate_sensitive>    ::= <string>
<delay_sensitive>   ::= <string>
<qos_attrib>        ::= (<throughput>, <delay>, <loss>, <jitter>)
<throughput>        ::= <string>
<delay>             ::= <string>
<loss>              ::= <string>
<jitter>            ::= <string>
<string>            ::= "<stringchar>*"
                      | #<digit><digit>*"<ascii>*
<stringchar>        ::= \<ascii>
                      | <ascii>-\-<double-quote>
```

### B.2.3 Sintaxe Completa da MngmtCL no formato BNF

MNGMTCL B.14 : **Linguagem de conteúdo MngmtCL**

```
<ServiceReq>       ::= (<ServSpec>, <Cost>, <TimeInterval>)
<ServiceOffer>     ::= (<ServSpec>, <Cost>, <TimeInterval>)
<SLA>              ::= (<ServSpec>, <cost>, <TimeInterval>)
<DSLA>             ::= (<ServSpec>, <cost>, <TimeInterval>)
<CostReq>          ::= (<ServSpec>)
<Cost>             ::= (<fcost>,<PC>,<PA_Q>, <PCR>, <TimeInterval>)
<TimeInterval>     ::= <string>
<fcost>            ::= <string>
<PC>               ::= <string>
<PA_Q>             ::= <string>
<PCR>              ::= <string>
<Neighbourhood>    ::= <AgentNameList>
<AgentNameList>    ::= <string*>
<AServ>            ::= <ServSpec>
<IntFace>          ::= (<VPI>,<VCI>)
<ChOper>           ::= (<VPI>,<VCI>)
<EndCom>           ::= (<VPI>,<VCI>)
<ServSpec>         ::= (<ServSpecContent>,<undtech>)
<undtech>          ::= <ATM>
                     | <IP>
<ATM>              ::= <string>
<IP>               ::= <string>
<ServSpecContent>  ::= (<S_D_pair>, <Signalling>, <QoS>)
                     | (<S_D_pair>, <qos>)
<S_D_pair>         ::= (<Src_Address>, <Dest_Address>)
<Src_Address>      ::= (<ATM_Address>, <VPI>, <VCI>)
                     |  <string>
<Dest_Address>     ::= (<ATM_Address>, <VPI>, <VCI>)
                     |  <string>
<ATM_address>      ::= <string>
<VPI>              ::= <string>
<VCI>              ::= <string>
<Signalling>       ::= <bool>
<QoS>              ::= (<Class> | <RSVP>, <qos_attrib>)
<Class>            ::= <deterministic>
                     | <best_effort>
<deterministic>    ::= <CBR>
                     | <VBR>
```

*Continua na página seguinte ...*

**Linguagem de conteúdo MngmtCL** *(continuação)*

```
<CBR>              ::= <string>
<VBR>              ::= <string>
<best_effort>      ::= <ABR>
                     | <UBR>
                     | <string>
<ABR>              ::= <string>
<UBR>              ::= <string>
<RSVP>             ::= <best_effort>
                     | <rate_sensitive>
                     | <delay_sensitive>
<rate_sensitive>   ::= <string>
<delay_sensitive>  ::= <string>
<qos_attrib>       ::= (<throughput>, <delay>, <loss>, <jitter>)
<throughput>       ::= (<pcr>, <scr>, <mbs>, <mcr>)
                     |  <string>
<pcr>              ::= <string>
<scr>              ::= <string>
<mbs>              ::= <string>
<mcr>              ::= <string>
<delay>            ::= <string>
<loss>             ::= <string>
<jitter>           ::= <string>
<undtech>          ::= <IP>
<string>           ::= ``<stringchar>*''
                     | #<digit> <digit>* ''<ascii>''
<stringchar>       ::= \<ascii>
                     | <ascii>-\-<double-quote>
<bool>             ::= <T>
                     | <F>
<T>                ::=  1
<F>                ::=  0
```

## B.3 Expressões da Linguagem de Conteúdo e Performativas KQML Associadas

| Objecto | Performativa | Semântica associada |
|---|---|---|
| | advertise(ask-if) | Enviada por um agente ASPA ao agente FA para informar que aceita pedidos para fornecimento de serviço de ligação. |
| | unadvertise | Enviada por um agente ASPA ao agente FA para informar que já não aceita mais pedidos para fornecimento de serviço de ligação. Por outras palavras, desfaz o efeito da performativa *advertise* anterior. |
| | broker-all | Utilizada por um agente UA para que o agente FA lhe indique todos os agentes ASPA que lhe possam prestar um serviço de ligação. |
| | recomend-one | Utilizada por um agente UA para o agente FA lhe recomendar um ASPA agent que lhe possa prestar um serviço. |
| | recomend-all | Utilizada por um agente UA para o agente FA lhe recomendar todos os agentes ASPA que lhe possam prestar um serviço de ligação. |
| ServiceReq | recruit-one | Utilizada por um agente UA para o agente FA lhe recrutar um agente ASPA que lhe possa prestar um serviço. |
| | recruit-all | Utilizada por um agente UA para que o agente FA lhe recomende todos agente ASPA que lhe possam prestar um serviço de ligação. |
| | ask-if | Utilizada por um agente UA para perguntar a um agente ASPA se este lhe pode estabelecer um determinado circuito de ligação. |
| | ask-all | Utilizada por um agente UA para perguntar a um agente ASPA sobre todos os tipos de serviço ele lhe pode prestar de momento. |

| | | |
|---|---|---|
| | stream-all | Utilizada por um agente UA para perguntar a um agente ASPA sobre todos os tipos de serviço ele lhe pode prestar de momento. Outra forma similar à anterior. |
| | subscribe | Utilizada por um agente UA para perguntar a um agente ASPA sobre todos os tipos de serviço ele lhe pode prestar de momento. Outra forma similar à anterior. |
| | untell | Utilizada por um agente ASPA para informar que não pode prestar o serviço pretendido. |
| | forward(untell) | Utilizada pelo agente FA, para fazer seguir a resposta da performativa *ask-if*(performativa *untell*) do agente ASPA, até ao agente UA. |
| ServiceOffer | tell | Utilizada pelos agentes ASPA para, por um lado submeter aos agentes UA propostas de contratos de prestação de serviços, por outro para submeter propostas de subcontratação de serviços |
| | tell | Utilizada por um agente ASPA para indicar que pode prestar o serviço pretendido, informando as condições do mesmo. |
| | untell | Utilizada por um agente ASPA para indicar o fim da prestação de serviço |
| SLA | forward(tell) | Utilizada pelo agente FA, para fazer seguir a resposta da performativa *ask-if* (performativa *tell*) do agente ASPA, até ao agente UA. |
| | achieve | Enviada por um agente UA a um agente ASPA para iniciar o contrato de prestação de serviços. |
| | unachieve | Enviada por um agente UA a um agente ASPA para terminar o contrato de prestação de serviços. |
| DSLA | achieve | Enviada por um agente UA a um agente ASPA para informar que não concorda com o contrato de prestação de serviços proposto. |
| CostReq | ask-if | Utilizada por um agente UA para perguntar a um agente ASPA sobre custos associados ao estabelecimento de um circuito de ligação. |

| | | |
|---|---|---|
| | untell | Resposta do agente ASPA ao pedido de informação sobre custos associados a um serviço devido à indisponibilidade do respectivo estabelecimento. |
| Cost | tell | Indicação de um agente ASPA a um agente UA sobre a forma de cálculo de custos sobre o serviço pretendido. |
| Neighbourhood | advertise | Indicação de um agente ASPA ao agente FA sobre todos os agentes ASPA vizinhos. |
| AServ | achieve | Pedido de um agente UA a outro para que este aceite o pedido de estabelecimento de um circuito. |
| IntFace | tell | Informação de um agente UA a outro sobre a configuração da sua interface de rede. |
| ChOper | achieve | Pedido de um agente UA a outro para reconfiguração das interfaces do circuito de comunicação, por motivos de contratação de outra operadora, sem pretensão de interromper comunicação já estabelecida. |
| EndCom | achieve | Pedido de um agente UA a outro para libertação da interface de rede por motivos de fim do estabelecimento de um serviço. |

Tabela B.15: Expressões da linguagem de conteúdo e performativas KQML associadas