Processing Schedules using Distributed Ontologies on the Semantic Web

Terry R. Payne, Rahul Singh, and Katia Sycara

The Robotics Institute, Carnegie Mellon
University, 5000 Forbes Avenue, Pittsburgh PA 15232, USA.

Abstract. The Semantic Web uses formal distributed ontologies for representing relationships among concepts in the real world. A structured framework such as this allows agents to peruse and reason about published knowledge without the need for scrapers, information agents, and centralized ontologies. However, in order to process any information, an agent must be familiar with the underlying ontology used to markup that information. However, no single agent can be expected to be familiar with all possible ontologies that may be available on the Semantic Web. Therefore, translation services that transform concepts defined within previously unknown ontologies into known concepts allow agents to understand the available information and hence achieve their goal. This transformation may be achieved by invoking specific markup translation services or by logical reasoning through other shared ontologies. The RETSINA Calendar Agent (RCal) is a Distributed Meeting Scheduling Agent that processes schedules marked up on the Semantic Web, and imports them into the user's Personal Information Manager. Translation services, which are used to translate unknown concepts into known concepts, which are located using a DAML-S based service discovery mechanisms. In this paper, we present RCal, and demonstrate how it extracts and uses meaningful knowledge from the semantic markup. In addition, we describe how web-service discovery mechanisms are used when new concepts are encountered.

1 Introduction

The World Wide Web was originally designed as a distributed information space that seamlessly supported human navigation through related, linked documents. Although this medium was designed to do more than simply support human-to-human communication [3], machine or agent mediated assistance has been hindered by the type of markup used within the documents. An emphasis by content providers on presentation and physical design has resulted in a lack of structure, both at the layout and content levels, and rendered most documents opaque to machine comprehension. In addition, the web has evolved from a distributed repository of documents, to a linked network of services offering not only the delivery of dynamic information, but offering functional services (e.g. currency conversions), e-commerce services (e.g. airline reservation or retail

services), and home-management services (e.g. paying bills or managing bank accounts). However, access to these services is still human oriented, and not amenable to automation.

The Semantic Web [3] goes beyond the World Wide Web by encoding knowledge using a structured, logically connected representation, and providing sets of inference rules that can be used to conduct automated reasoning. Since the Semantic Web does not require content providers to use a single centralized ontology; but instead supports the use of many different ontologies, it is unrealistic to assume that agents and services will understand all possible markup. Hence many different agents providing specific services will translate/convert information from one ontology to another as required, thus encapsulating and providing specific functionality to the user. Also, given that the Semantic Web is dynamic and evolving, not only will there be several ontologies for describing a domain, but these will be extended over time, and new ontologies will be created. Although an agent may be designed to understand one, or even several ontologies for a given domain, it will encounter new markup using previously unknown ontologies. For this reason, an agent should be able to adapt its model or reasoning, and make use of third-party services to achieve interoperation between the new markup and known markup.

An essential capability of any agent within an open framework such as the Semantic Web is the ability to locate and interoperate with other services [13, 14, 10]. To locate these, an agent has to be able to describe the desired capabilities of some service and interact with a discovery infrastructure. Such infrastructures typically consist of *broker* or *yellow-pages* services that help agents (i.e. *service requesters*) find *service providers* [14]. *Brokers* typically have knowledge of both the capabilities and preferences of different agents and services [16], and are often found within market-based systems. *Matchmakers*, yellow pages or directory agent systems [14, 15] would only possess knowledge about the capabilities of service providers through capability advertisements. Thus, if an agent has some preferences, it can query a *Matchmaker*, which then returns a list of services that provide the desired capabilities specified by the preference query.

Existing Web-Service languages, such as WSDL, ebXML, UDDI etc facilitate the automation of services on the web by providing semi-structured descriptions of services in terms of their providers, their functionality or workflow and their interfaces. This structure-representation may facilitate some level of automation for tasks such as human-oriented database search and the formation of messages understood by a service's interface. However, automatic composition of meta-services and interoperation between these services can only be achieved through the use of semantic-based frameworks such as DAML-S [1, 2].

DAML-S is a DAML+OIL [4] ontology for describing the properties and capabilities of web services. It consists of descriptive semantics for describing what the service does (though an advertised *Service-Profile*), how the service works (though a *Service-Model*) and how it can be invoked (though a *Service-Grounding*). The Service-Grounding builds upon the WSDL and SOAP standards for describing the service interface and a communication framework. A

DAML-based Discovery Service [9] stores the advertised service descriptions (i.e. Service-Profiles), and uses a semantic matching engine to compare these with service requests. Upon receiving the results of a request-for-service, an agent can then contact the service, by following the description in the Service-Model, and generating the SOAP messages via the Service-Grounding.

In this paper, we present a domain specific agent - the RETSINA Calendar Agent (*RCal*), which understands semantic markup for schedules and events, and show how service discovery using DAML-S can be employed to find translation services when new markup is encountered. The organization of the paper is as follows: in Section 2, we describe RCal, and describe how it utilizes and reasons about Semantic Web schedules. Section 3 discusses the role of service discovery and translation services. We then contrast this work with related systems and conclude the paper in Section 4.

2 RCAL - RETSINA Calendar Agent

The *RETSINA Calendar Agent (RCAL)*[1] is a distributed meeting scheduling agent that reasons about events within schedules such as conference programs and class schedules published on the Semantic Web. It maintains an up-to-date model of the user's current and upcoming events and uses this information to schedule meetings on behalf of the user. The agent's effectiveness in scheduling meetings autonomously without conflicts depends upon the accuracy of this model. Although many Calendar Agents have previously been developed that manage meeting requests on behalf of the user [5, 7], they are only effective if they have an up-to-date, high fidelity model of the user's preferences and current activities. Machine-learning techniques can be used to induce the user model, which can then be used to guide the agent when scheduling regular appointments. However, this approach cannot anticipate new, infrequent or unexpected events, and hence the user generally has to manually enter in those activities and events to notify the agent that these times are busy. Thus, to provide assistance and reduce the load on their user, calendar agents paradoxically require significant effort from the user to maintain an accurate, valid model of the user's current schedule. RCal addresses this problem by simplifying the acquisition of schedule markup from the Semantic Web, and importing this directly into the user's calendar. The gain lies in the fact that users no longer need to accurately type in each event that they would like to attend but rather only need to direct the agent to a schedule that the agent can reason about and automatically add into the calendar. In addition to this, other services can also be dynamically offered to the user, depending upon the content of the schedules browsed.

RCal works synergistically with a commercial Personal Information Manager (PIM)[2]. It retrieves appointments and contact details from the PIM, and uses these to locate potential meeting slots. Current and upcoming appointments, and contact details of people known to the user are retrieved from the PIM.

[1] RCal is available for download from *http://www.daml.ri.cmu.edu/Cal*

[2] The Personal Information Manager used here by RCal is Microsoft Outlook 2000

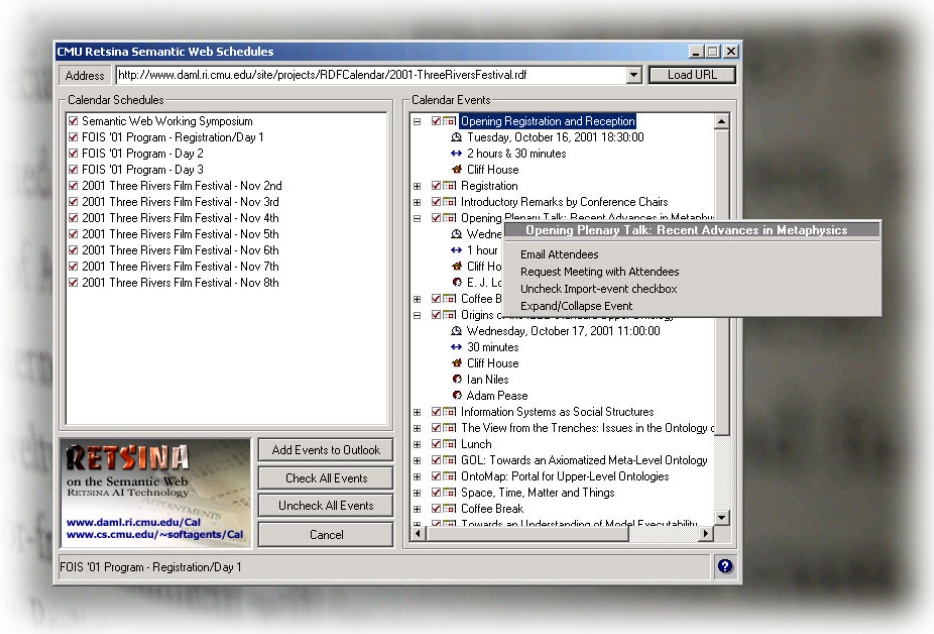

Fig. 1. The RCAL Semantic Web Schedule Browser.

This information is then used while negotiating and reasoning about available meeting times with other RCal agents. A user can specify a desire for a meeting with several other individuals over a given epoch (for example, "*arrange a meeting sometime next Tuesday with the members of my team*"). The user's RCal agent will then negotiate with other RCal agents (associated with the other individuals) to determine mutually available time slots for a meeting. The actual negotiation of possible meeting times is based on the Contract Net Protocol [12], whereby contracts are sent out to each of the agents involved to solicit possible meeting times. Bids returned by the agents are evaluated to determine a suitable meeting time. Once a mutually agreeable meeting time is found (which may require several negotiation cycles with changing time constraints), it is announced using awards sent out in response to the bids. The agents involved in the negotiation then confirm the selected meeting time, and update the calendars of their respective users.

As well as using RCal to schedule meetings with other RCal users, the agent can be used to navigate and import schedules marked up on the Semantic Web. This might include one-off schedules such as conference programs, or new, recurring appointments such as class schedules. The user can either type in the URI of the schedule markup, or select and submit a schedule via a button on a

```
<foaf:Person ID="terrypayne">
  <foaf:name>Terry Payne</foaf:name>
  <foaf:mbox resource="mailto:terryp@cs.cmu.edu"/>
  <foaf:workplaceHomepage resource="http://www.cs.cmu.edu/~terryp"/>
  <foaa:RCalendarAgentName>terry_acm.org-CalAgent</foaa:RCalendarAgentName>
</foaf:Person>

<ical:VCALENDAR ID="TAC01">
  <dc:title>Trading Agent Competition 2001 Workshop</dc:title>
  <dc:contributor resource="#terrypayne"/>
  <dc:date>2001-10-03</dc:date>

  <ical:VEVENT-PROP resource="http://www.tac.org/2001event.rdf#PainInNEC"/>

  <ical:VEVENT-PROP>
    <ical:VEVENT ID="RetsinaTrading">
      <ical:DTSTART>
        <ical:DATE-TIME><value>20011014T134500</value></ical:DATE-TIME>
      </ical:DTSTART>
      <ical:DTEND>
        <ical:DATE-TIME><value>20011014T140000</value></ical:DATE-TIME>
      </ical:DTEND>
      <ical:LOCATION resource="#HRTampa" />
      <ical:ATTENDEE resource="http://www.daml.ri.cmu.edu/people.rdf#ks" />
      <ical:ATTENDEE resource="http://www.daml.ri.cmu.edu/people.rdf#yn" />
      <ical:DESCRIPTION>Presentation: Retsina</ical:DESCRIPTION>
    </ical:VEVENT>
  </ical:VEVENT-PROP>
</ical:VCALENDAR>
```

Fig. 2. A Schedule containing two events.

web page[3]. Schedules can be browsed using the agent's Semantic Web Schedule Browser (Fig. 1), and compared with existing appointments to identify conflicting events. These schedules and the contact details of people attending the meetings can be imported into the user's PIM. For example, a user may browse the talks being given at a conference, and then import only those talks of interest. Notifications can also be associated with each talk and sent to a mobile device (such as a PDA or mobile-phone) to remind the user when each talk is about to start.

The Semantic Web expresses concepts and their properties using the Resource Description Framework (RDF) [6], which is in turn expressed in XML. RDF encodes concepts and their relationships as sets of triples, where each triple represents a subject, predicate and an object. Both subject and object can, in turn, be related to other concepts using additional properties (or predicates), forming a directed graph. RDF extends this natural structure by allowing con-

[3] Currently, this technique relies on the use of cookies to share the RCal agent's name with the web page. When the submit button is pressed, a browse request containing the relevant URI is automatically sent to the agent, which then displays this schedule in the schedule browser (Fig. 1).

cepts to be represented by URIs to other concepts. Thus, concepts are no longer terms bound to a single node, but unique definitions that can be shared by multiple documents. For example, one could refer to the author of a paper by their name, but as that name is unlikely to be unique, it becomes problematic to reason about what other papers have been written by the same author. However, the author could be represented by a URI referring to the concept that uniquely defines that author, and thus avoid ambiguities when reasoning. In addition, the author concept may contain properties to other concepts, thus providing a richer environment for reasoning.

Calendar schedules are marked up using several different ontologies. The schedules and events themselves are described using the *Hybrid RDF Calendar Ontology (iCal)*[4]. This ontology is based upon several Calendar specifications, including those from the Calendaring and Scheduling (calsch) Working Group, Distributed Scheduling Protocol (CHRONOS) IETF Working Group, and the *iCalendar* specification. <`ical:ATTENDEE`> resources are currently described using the Friend-of-a-Friend ontology[5], and the Dublin Core ontology[6] is used to provide meta-data about the schedule. An example of a schedule is presented in Fig. 2.

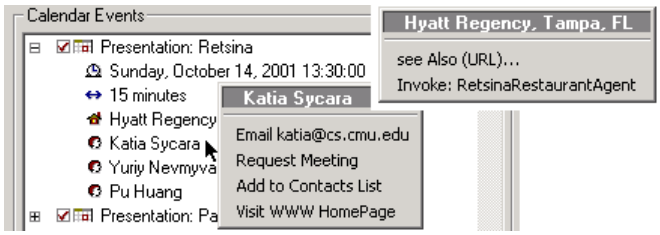

Fig. 3. Browsing schedules and invoking context-based services/agents.

A document may contain one or more calendars or schedules, represented by instances of the <`ical:VCALENDAR`> class. Each instance may include several properties, such as <`ical:VEVENT-PROP`>, which connect a <`ical:VCALENDAR`> to an event class, or a date-time stamp (<`ical:DTSTAMP`>). However, this markup is not limited to the elements described in the iCal ontology; properties defined by other ontologies can be reused to enhance the document. In this case, properties defined by the Dublin Core ontology are used to provide additional information about a schedule, such as its title, description, author etc. The calendar may contain several events, represented by an event class (such as a single event or meeting, <`ical:VEVENT`>, or recurring events, <`ical:REC-VEVENT`>).

[4] For the iCal ontology, see *http://ilrt.org/discovery/2001/06/schemas/ical-full/hybrid.rdf*

[5] For the Friend of a Friend ontology, see *http://xmlns.com/foaf/0.1/*

[6] For the Dublin Core ontology, see *http://dublincore.org*

These may be defined inline within the document (as in the "RetsinaTrading" event in Fig. 2), or referenced by a URI to a resource in the same or an external document (e.g. the "`http://www.tac.org/2001event.rdf#PainInNEC`" resource). Another example of resource reuse is the markup of an individual's contact details. Such details may be encoded using the Friend-of-a-Friend or DAML Markup Agenda (DMA)[7] ontologies, and used by many agents, services and other Semantic Web tools. Currently, RCal will recognize an individual's resource description marked up using the Friend-of-a-Friend ontology.

```
<dma:Speaker rdf:ID="payne">
  <dma:name>Terry Payne</dma:name>
  <dma:email>terryp@cs.cmu.edu</dma:email>
  <dma:homePage>
    http://www.cs.cmu.edu/~terryp
  </dma:homePage>
</dma:Speaker>
```

```
<foaf:Person rdf:ID="payne">
  <foaf:name>Terry Payne</foaf:name>
  <foaf:mbox
    rdf:resource="mailto:terryp@cs.cmu.edu"/>
  <foaf:workplaceHomepage>
    rdf:resource="http://www.cs.cmu.edu/~terryp"/>
</foaf:Person>
```

Fig. 4. Markup for contact details using the DAML Markup Agenda (DMA) ontology and the Friend of a Friend (foaf) ontology.

The ability to refer to resources defined in different ontologies facilitates the navigation of information not directly related to a schedule. For example, RCal locates the name property of an <`ical:ATTENDEE`> (i.e. the <`foaf:Person`> concept illustrated in Fig. 2) when listing the events within the semantic web schedule browser. Other information, such as email or webpage properties may also be defined. These can be used to facilitate additional browsing or offer additional services. For example, a browser could be invoked to display the web page, or a mail client to send an email, if this information is available. If the <`ical:ATTENDEE`> concepts in Fig. 2 contain more than just a first and last name, then additional services are offered to the user when the user selects a concept (e.g. the user right-clicks the <`ical:ATTENDEE`> concept "`Katia Sycara`" in Fig. 3). These properties can also be used to query service providers (i.e. other agents) via a discovery infrastructure (such as a DAML-S Matchmaker [9]). This form of serendipitous service discovery (as opposed to goal-directed service discovery) attempts to find any service that might be of use to the user.

3 Discovery Services and Translation Agents

Whilst RCal can provide browsing and download functionality for schedules marked up using known ontologies, it is unable to understand markup using

[7] See *http://www.daml.org/2001/10/agenda/*

unknown ontologies such as the DAML Markup Agenda (DMA) Ontology or that used by *ITTalks*[8][11]. RCal overcomes this limitation by seeking translation services that convert the new concepts into known concepts. An example of this is the *DMA2FOAF* translation service[9] that translates markup for <dma:Speaker> concepts and converts these into <foaf:Person> concepts (see Fig 4).

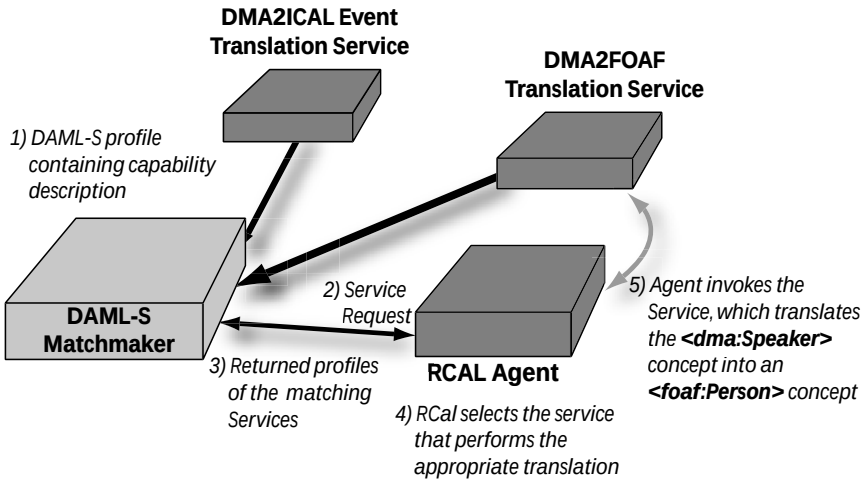

Fig. 5. Advertising and requesting services from the DAML-S Matchmaker

The *DMA2FOAF* translation service describes its capabilities using the DAML-S [1, 2] service description. This includes information about the *Process Model*, i.e. what the service does; the *Service Profile* i.e. the capability advertisement; and the *Service Grounding*, which describes how the service is contacted. When this service is made available on the Semantic Web, it advertises its capabilities with a DAML-S Matchmaker (Fig. 5, Step 1). The RCal agent then detects an unknown concept within the markup, and constructs a request for service based on this concept. This request consists of the properties found within the unknown concept, and the properties desired from the FOAF ontology (Fig 6.). This is then submitted to the DAML-S Matchmaker (Fig. 5 Step 2), which performs a semantic capability match between the request and the stored *Service Profiles*, before returning a list of possible matching services (Fig. 5 Step 3). RCal then selects and invokes one of the returned services (in this case - *DMA2FOAF* in Fig. 5 Step 4), by sending it a query containing the values of the properties

[8] For the ITTalks ontologies, see *http://daml.umbc.edu/ontologies/calendar-ont*
[9] *http://www.daml.ri.cmu.edu/site/projects/DMATranslation/ont/DMA2FOAF.daml*

found within the unknown concept. The service then returns the properties of the known <`foaf:Person`> concept and RCal utilizes this concept accordingly.

```
<profile:NeededService rdf:ID="my_request">
  <profile:serviceCategory rdf:ID="&category;#OntologyTranslation" />
  <profile:input>
    <profile:ParameterDescription rdf:ID="inSpeaker">
      <profile:parameterName>inSpeaker</profile:parameterName>
      <profile:restrictedTo rdf:resource="&dma;#Speaker" />
    </profile:ParameterDescription>
  </profile:input>
  <profile:output>
    <profile:ParameterDescription rdf:resource="outPerson">
      <profile:parameterName>outPerson</profile:parameterName>
      <profile:restrictedTo rdf:resource="&foaf;#Person" />
    </profile:ParameterDescription>
  </profile:output>
</profile:NeededService>
```

Fig. 6. Constructing a Request for a translation service.

The use of concept-to-concept translation services may provide a suitable mapping between the markup using two ontologies, but not a more generic solution that can be more easily applied to markup using other ontologies. Various schemes (such as Onion [8]) identify the correspondence between different ontologies, and can be used to define domain dependent rules for markup translation. However, such rules would also need locating if not linked to the unknown ontology, and hence could be advertised as static markup.

4 Conclusions and Related Work

This paper demonstrates how service discovery and information sharing can allow agent communities to locate and present relevant services to a user, based on the information that is being browsed. Although several different ontologies may be used to markup content, translation services can transform unknown markup into that which can be understood by the agent. RCal makes use of a serendipitous search to look for services that may be of use to the user, based on selected resources. However, in a service rich environment, many irrelevant services may be presented to the user. Thus, work is currently underway to develop profiles of the user's interest, and to infer context (such as locating restaurants in favor of hardware stores when examining the location of a conference site). The *ITTalks Agent* system [11] is an existing web-based system that provides automated, intelligent notification of information technology seminars. Profiles

of user preferences, annotated in DAML+OIL [4], are used to suggest those seminars that might be of interest to the user. These examples demonstrate how, by combining the semantics now available through the Semantic Web, communities of agents can interoperate and work synergistically to provide better access to information and functionality than was previously available on the World Wide Web.

Acknowledgements

The research was funded by the Defense Advanced Research Projects Agency as part of the DARPA Agent Markup Language (DAML) program under Air Force Research Laboratory contract F30601-00-2-0592 to Carnegie Mellon University. Special thanks goes to L. Miller and S. Decker for proposing the original Semantic Web Calendar Challenge, and to O. Lassila for some illuminating discussions about the Calendar Agent Assistance Paradox.

References

1. A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. DAML-S: Semantic markup for web services. In *International Semantic Web Working Symposium*, pages 411–430, 2001.
2. A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S: Web service description for the semantic web. In *Proceedings of the 1st International Semantic Web Conference (ISWC)*, 2002.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.
4. J. Hendler and D. L. McGuinness. DARPA Agent Markup Language. *IEEE Intelligent Systems*, 15(6):72–73, 2001.
5. R. Kozierok and P. Maes. A Learning Interface Agent for Scheduling Meetings. In *Proceedings of the ACM-SIGCHI International Workshop on Intelligent User Interfaces*, pages 81–88. New York, New York:ACM Press, 1993.
6. O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. http://www.w3.org/TR/REC-rdf-syntax/, 1999.
7. T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a Learning Personal Assistant. *Communications of the ACM*, 37(7):81–91, 1994.
8. P. Mitra, G. Wiederhold, and S. Decker. A scalable framework for the interoperation of information sources. In *Semantic Web Working Symposium*, pages 317–329, 2001.
9. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *First International Semantic Web Conference*, 2002.
10. T. Payne, K. Sycara, M. Lewis, T. L. Lenox, and S. K. Hahn. Varying the user interaction within multi-agent systems. In *Autonomous Agents 2000*, 2000.
11. R.Scott Cost et. al. ITTalks: A case student in how the semantic web helps. In *Int. Semantic Web Working Symposium*, pages 477–494, 2001.

12. R. G. Smith. The Contract Net Protocol: High-Level Communications and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, C29(12), 1980.
13. K. Sycara, K. Decker, A. S. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert*, 11(6):36–46, December 1996.
14. K. Sycara, K. Decker, and M. Williamson. Middle-agents for the internet. In *Proceedings of IJCAI-97*, January 1997.
15. K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Record (ACM Special Interests Group on Management of Data)*, 28(1):47–53, 1999.
16. M. Wellman. A Market-Oriented Programming Environment and its Application to Distributed Multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.