DESIGN AND IMPLEMENTATION OF A P2P CONTRACTING OVERLAY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

REMZİ ÇELEBİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JUNE 2009

Approval of the thesis:

DESIGN AND IMPLEMENTATION OF A P2P CONTRACTING OVERLAY

submitted by **REMZİ ÇELEBİ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Müslüm Bozyiğit _____
Head of Department, **Computer Engineering**

Prof. Dr. Faruk Polat _____
Supervisor, **Computer Engineering Department, METU**

Asst. Prof. Dr. Hürevren Kılıç _____
Co-supervisor, **Computer Department, Atılım University**

Examining Committee Members:

Assoc. Prof. Dr. Ahmet Coşar
Computer Engineering, METU _____

Prof. Dr. Faruk Polat
Computer Engineering, METU _____

Asst. Prof. Dr. Hürevren Kılıç
Computer Engineering, Atılım University _____

Asst. Prof. Dr. Tansel Özyer
Computer Engineering, TOBB ETU _____

Asst. Prof. Dr. Pınar Şenkul
Computer Engineering, METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: REMZİ ÇELEBİ

Signature :

ABSTRACT

DESIGN AND IMPLEMENTATION OF A P2P CONTRACTING OVERLAY

Çelebi, Remzi
M.S., Department of Computer Engineering
Supervisor : Prof. Dr. Faruk Polat
Co-Supervisor : Asst. Prof. Dr. Hürevren Kılıç

June 2009, 50 pages

Today, with widespread use of Internet in many areas, the common procedures frequently encountered in business life such as contracting and negotiation need to be automated. The distributed structure of the Internet and the difficulty of resources dispersed on one center makes such a system to have a distributed architecture . In this study, for first time, automatization of a contracting form through business processes was proposed and was carried out.

A peer to peer process contracting overlay what we call Peer-Con is developed. The system is an extension of Java Agent Development Framework (JADE) and uses IEEE Foundation for Intelligent Physical Agents (FIPA) Agent Communication Language (ACL) standard. Cost aware flexible representation of process capabilities; description of an operator to decide on whether given capabilities turnout to an agreement or not and self organization of peer connectivity for better contracting performance are distinguishing features of the system. The system can easily be adapted to different domains while the core functionality remains the same. Practical use of Peer-Con is shown by two applications from different domains; Driving Route Calculation on Web Maps and Digital Signal Processing Module (DSPM) product planning domain.

Keywords: Business Process Contracting, Peer-to-Peer Systems, Cooperating Agents, Process Representation, Multi-agent Systems

ÖZ

DENK- UÇ YERPAYLAŞAN SİSTEMLERDE BİR KONTRATLAMA TASARIMI VE GERÇEKLEŞTİRİMİ

Çelebi, Remzi
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Faruk Polat
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Hürevren Kılıç

Haziran 2009, 50 sayfa

Günümüzde internetin yaygınlaşması ile birçok alanda olduğu gibi sözleşme ve kontratlama gibi iş hayatında sık karşılaşılan yordamların otomatikleştirilmesinin ihtiyacı doğmuştur. İnternet ortamının dağınık yapısı ve kaynakların bir merkezde bulunma güçlüğü gibi nedenlerden dolayı böyle bir sistem için dağınık bir yapının zorunluluğu ortaya çıkmaktadır. Bu çalışmada ilk defa iş süreçleri(process) üzerinden otomatikleştirilmiş bir iş akdi biçimi sunulmuş ve gerçekleştirilmiştir. Önerilen sistem denk uç sistemlerin üzerinde yerpaylaşan bir yapı olup, Per-Con olarak adlandırılmaktadır. Bahsedilen sistem Java Agent Development Framework (JADE)' unun uzantısı olup, IEEE Foundation for Intelligent Physical Agents (IEEE FIPA) Agent Communication Language (ACL) standardı kullanılarak gerçekleştirildi. Sistem şu ayrıt edici özellikleri içinde barındırmaktadır; İş süreçlerini ve bunun yeteneklerinin fiyatlandırmasını sağlayan bir gösterim şekli ve bunu izleyen iş süreçlerinin beraber çalıştırılabilmesi için eşler arası anlaşmaya imkan verebileceğini veya verilemeyeceğini belirleyen bir mekanizma ve en son olarak daha iyi bir iş akdi için de dinamik komşuluk bağlantısını kendi kendine düzenleme. Sistem, çekirdek fonksiyonellik aynı kalınarak başka etki alanına kolayca adapte edilebilir. Peer-con pratik kullanımı, Digital Signal Process-

ing Module (DSPM) cihazlarının üretim tasarım planında ve optimal rota hesaplamasında gösterilmiştir.

Anahtar Kelimeler: İş süreci Kontratlama, Denk-Uç Sistemleri, Kooperatif Erkler, İş süreç gösterimi, Çoklu-Ajan Sistemleri

ACKNOWLEDGMENTS

I want to give my thanks to my thesis advisor, Prof. Dr. Faruk Polat, for his guidance, support throughout my research. In particularly, I wish to thank my co-advisor Hürevren Kılıç for not just this work but for all things he helped me with during my university study. He has been full of ideas, enthusiasm and always motivating and I enjoyed working with him a lot. I also very much like to thank my family and my friends for their support and encouragement. Thanks also go to all others who have contributed in this thesis directly or indirectly. Lastly, I would like to thank Turkish Scientific and Technical Council (TUBITAK) for their financial support during my graduate study.

TABLE OF CONTENTS

ABSTRACT iv

ÖZ vi

ACKNOWLEDGMENTS viii

TABLE OF CONTENTS ix

LIST OF TABLES xi

LIST OF FIGURES xii

CHAPTERS

1 INTRODUCTION 1

2 BACKGROUND and RELATED WORKS 4

2.1 Background 4

2.1.1 Peer-to-Peer Systems 4

2.1.1.1 Gnutella Protocol 6

2.1.2 Multi-agent Systems 8

2.1.2.1 JADE 9

2.2 Related Works 10

2.2.1 Business Process Representation 10

2.2.1.1 BPEL 12

2.2.1.2 RosettaNet 12

2.2.1.3 ebXML 13

2.2.1.4 Finite State Automaton 13

2.2.1.5 Workflow Nets 14

2.2.2 Process based Inter-organizational Agreement 14

2.2.3 Matchmaking/Contracting Environment 15

3 The OVERLAY 17
3.1 System Architecture 17
3.1.1 Peer Interactions 19
3.2 Single Peer Implementation 22
4 OPERATORS AND DESCRIPTIONS USED IN THE OVERLAY 31
4.1 Description of Business Process 31
4.2 Description of Match Operator 32
4.3 Illustrative Example of Definitions 33
5 APPLICATIONS 35
5.1 Driving Route Calculation on Web Maps 35
5.2 Digital Signal Processing Module Product Planning Application . . . 43
6 CONCLUSION 45
6.1 Future Directions 46
REFERENCES 47

LIST OF TABLES

TABLES

Table 3.1 An example search cycle instance defined by cfc and rfc messages. 21

Table 5.1 Peer-Con concepts and their interpretations in web mapping context. 36

LIST OF FIGURES

FIGURES

Figure 2.1 Peer-to-Peer Overlay 6

Figure 3.1 Peer-Con layer and its environment. 18

Figure 3.2 User, originator peer and contracting network interactions. 20

Figure 3.3 An example search-for-contracting cycle with cfc and rfc messages. 21

Figure 3.4 Evaluation and Handling Unit. 23

Figure 3.5 Pseudocode for CFC Handler. 24

Figure 3.6 CFC Message Sample. 25

Figure 3.7 Pseudocode for RFC Handler. 27

Figure 3.8 RFC Message Sample 28

Figure 3.9 Pseudocode for Neighborhood Organizer and ping/pong examples. 30

Figure 4.1 Example process descriptions and merge operation result. 34

Figure 5.1 Road map of eastern part of Turkey and example hypothetical transportation costs between cities. 37

Figure 5.2 Peer-Con Process Description tab. 38

Figure 5.3 Another example of hypothetical transportation costs between cities 39

Figure 5.4 Merge of two processes. 40

Figure 5.5 Peer-Con search and query result tab. 41

Figure 5.6 Peer-Con system parameters settings tab. 42

Figure 5.7 Example DSPM processes and their merge 44

CHAPTER 1

INTRODUCTION

In parallel to the development of the Internet, most information-centric industrial automation systems require inter organizational knowledge sharing and its automated cooperative processing to improve business success. Thus, the need of automation of common procedures in business life such as contracts and negotiations become apparent. To realize such a contracting and further negotiation between software agents, automated mechanism for search and selection of relevant services and agents is required [42]. It makes such an environment very challenging due to the distributed nature of Internet and the availability of services and resources in demand, mostly, on different agents.

In the thesis, the idea of matchmaking and contracting based on business processes is exploited. We define a process matchmaking as the agreement of a two contractors on basis process match found which satisfies a peer's process goal(s) in a collaborative way. In our context, process contracting is defined as finding a team of agents, where their cooperative joint process execution is realizable in order to satisfy a peer's goal. The basic difference between process matchmaking and process contracting is the consideration of (or utility) and handling of many-to-one engagements.

A typical matchmaking/contracting scenario can be considered as following steps; 1) publishing provider capabilities 2) requesting desired capabilities 3) finding appropriate match/-matches.

Peer-to-Peer (P2P) systems have potential to enhance the performance and reliability of interactions especially among individuals (or small scale enterprises). P2P architectures seems to be good candidate for the realization of decentralized information processing at process description sharing and consequently at automated process contracting[20]. However, there are

some challenges to build a P2P setups for process contracting like realistic process representation, automated identification of inter-organizational agreements through process elaboration and establishment of efficient peer connectivity management for better cooperative contracting.

Thus, establishment of such process matchmaking/contracting environments over decentralized interconnected computing environments should enable i) Local publication (or advertisement) of industrial (design, production, marketing etc.) capabilities in the form of process descriptions ii) Cheap contracting and automated capability exchange among industrial entities without support of any single facilitator.

We proposed and implemented a P2P process contracting system called Peer-Con, realized on overlay level P2P concepts and on agent-oriented development framework JADE. Due to its flexible and generic process description, it can be used for cooperative industrial design, production and marketing planning purposes among locally trusted organizations having frequently changing needs and capabilities. Peer-Con is developed to satisfy challenges mentioned above. Process description using a simple graph-based representation with edge consideration; provision to merge of multiple process descriptions; development of a protocol that allows a peer to contract with a team of other peers (i.e. multi-peer contracting) and self-organization of peer neighborhood topology via feedback from previous contract results are four basic distinguished features of the proposal. Besides from these, inherent P2P features like fault-tolerance and load-balancing provide an efficient and robust peer contracting.

Practical use of Peer-Con infrastructure is shown by two different applications: driving route calculation on web maps and the establishment of an industrial process contracting environment for Digital Signal Processing Module (DSPM) product planning. The problem of driving route calculation is simply to find out cheap contracts among peers for calculating least cost driving routes. The other problem is to find out cheap planning contracts among peers which represent producers. Process descriptions are representing the technological capabilities of producers together with their implementation costs. In [44], it has been pointed out that development of explicit process ontologies is one of the potentially viable approaches to provide a unified view for process networks. For our purpose, Peer-Con is also supported by a problem specific state-ontology and a random initial peer neighborhood settings. In our implementation, we developed a Contract-Net like [16] custom protocol by which potential

peer contracting space is searched efficiently.

In Peer-Con, we developed a custom peer interaction protocol facilitating robust decentralized contracting still by using the same primitives based on Gnutella 0.4 protocol that supports the unstructured topology of P2P setup. Ping, pong, query and query-hit are basic Gnutella messages interpreted into the basic syntax of standard IEEE FIPA-Agent Control Language (ACL) supported by JADE [8].

In Chapter 2, we give some background knowledge about peer-to-peer systems, multi-agent System and its relation to current work and related works on this study. In Chapter 4, formal definitions for the process representation and cost calculation operations are given. Peer-Con architecture and details of the peer interaction protocol has been outlined in Chapter 3. The case studies in Chapter 5 defines the use of Peer-Con for driving route calculation on web maps and process contracting for efficient DSPM (Digital Signal Processing Module) product planning. Chapter 6 contains conclusion and future research directions.

CHAPTER 2

BACKGROUND and RELATED WORKS

2.1 Background

Peer-con is both a peer-to-peer system and a multi-agent system. In this section, we give some background knowledge about these two systems and discuss how and why we facilitate our design and development using these two concepts.

2.1.1 Peer-to-Peer Systems

P2P systems consist of a group of distributed peers or computers that can communicate, collaborate and share resources between the other peers directly on an equal basis. In other words, all of the peers in the network have rights to initiate a query or connection, and they can respond to requests. This is in contrast to a client/server system in which a central server provides services to many of clients and there is no interaction between clients. However, peer-to-peer architecture proposes that each peer (a node in the P2P system) is both server and client (called servent). Any peer may both provide services to other peers and consume services from other peers. In client-server model, the network is highly dependent on the single server and if there is any failure at server, it causes the whole system to get down.

P2P systems have emerged as a promising way to share files (Napster, Gnutella, and FreeNet), computing resource (SETI@home). There exist some P2P systems based on distributed hashing algorithms where given an object , the algorithms guarantee to locate a peer that has that object such as CAN [10], Chord [14], and Pastry [1].

Functionally, P2P technology comes in three different forms, namely: pure-P2P, where each

participant has equal role; hybrid-P2P, where some of the nodes act as a central server and provide search facilities; and server based-P2P, where a dedicated server indexes control information but the data flow is between participants [37].

Pure P2P: The best known examples are Gnutella and FreeNet. Both of them have a pure distributed architecture, where there is no centralized database. All the peers in the systems establish a connection with others through request propagation.

Hybrid P2P: Hybrid P2P has recently emerged, for example, FastTrack. FastTrack has some supernodes, which are used for indexing the contents of part of the system and play a major role in the organization of the systems. A peer can become a supernode only if it has adequate resources, e.g., bandwidth and computing power.

Server Based P2P: The best example is Napster. Napster uses a centralized database to index the files each peer has in the system. To look for a file, a peer first sends a request to the database, and then gets a list of other peers who may have the files.

A peer in P2P system interacts with other peers through "neighbours" thus it avoids central control and the single point of failure. This interaction is defined through the algorithms that specify the roles of the participants and topology. Any shared participant' data can be stored/accessed by other participants, if needed, with or without knowing its origin.

The information resources, peers and connection links between them, in the P2P system could be quite dynamic where any peer or resource may appear and disappear at any time. If a participant fails or disappears, other participants can take over rebuilding the system.

Peer nodes in the network have specific functionalities although the physical capabilities may differ from one node to another. The general functionalities of a peer node in P2P Network can be described as; the ability to search the resources and serve the resources to participant use of specific protocol to provide communication and resources sharing.

Peer-to-Peer Networks are virtual networks and they overlay on the internet infrastructure. A peer could be a participant of P2P Network even it is located far from the other peers of the system as shown in Figure 2.1.

Morever, Peer-to-Peer (P2P) systems with their decentralized nature seems to be a good candidate for the establishment of locally trusted, cooperative and adaptive architectures for in-

dustrial information systems [25][18][17]. In practice, most of the state of art P2P-based interaction systems are focused on cooperative content sharing realized by efficient symmetric lightweight protocols [31][22][19][40][7][28]. Peer-to-Peer (P2P) system with their its advantages, user autonomy, ease and speed of growth, has potential to enhance the performance and reliability of interactions individuals especially among small-medium scale enterprises (SMEs). They have also potential for the realization of decentralized information processing at not only simple query-based content sharing but also at process description sharing and consequently at automated process contracting[20].

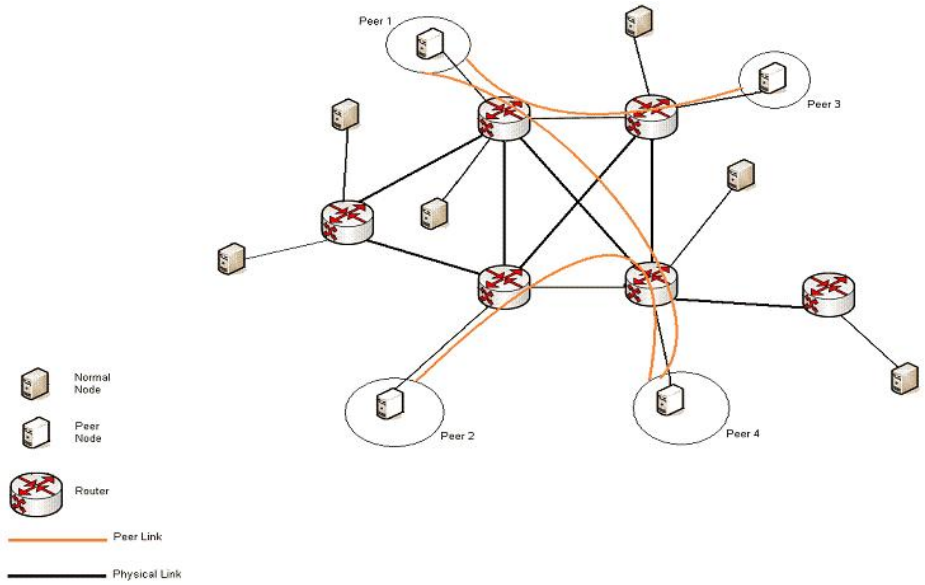


Figure 2.1: Peer-to-Peer Overlay

Gnutella is a popular pure P2P protocol and used by most of P2P applications. It is open-source protocol for distributed search and has fully decentralized architecture. We developed a custom peer interaction protocol facilitating robust decentralized contracting still by using Gnutella Protocol.

2.1.1.1 Gnutella Protocol

The Gnutella protocol consists of a set of message types representing the ways in which servents communicate over the network and a set of rules governing the inter-servent exchange of messages [28].

In Gnutella protocol, there are mainly four message types; Ping, Pong, Query, QueryHit, and one additional message type, Push. A ping message is used to actively discover hosts on the network. A servent receiving a Ping descriptor is expected to respond with one or more Pong descriptors. A pong is the response to a Ping message and it carries the same ID with the corresponding Ping message. It contains the address of a connected Gnutella servent and total size and number of shared files of this servent. The Query message is used for searching the distributed networks and holds the queried file name as string and the minimum download speed. If any peer that has already taken Query message has the file queried, the peer is expected to respond with QueryHit message. QueryHit is the response to a Query. This descriptor provides the recipient with enough information to acquire the data matching the corresponding Query [14]. Push is additional message type that is a mechanism for downloading from the firewalled servent.

The node joins the network by connecting to one Gnutella node, which can be any node on the network making it generally easy to join in a decentralized fashion. However, the acquisition of a peer IP address is not defined in the protocol. To find any peer in the P2P network for first time, some public web sites such as gnutellahosts.com were developed in a centralized manner. Once it has joined the node discovers other nodes through the first node by issuing ping and receiving pong descriptors from peers accepting connections [43].

Gnutella nodes typically connect to three nodes and then search by broadcasting their search request to all connected neighbours. Each neighbour repeats this search request to his/her neighbours and so on, which is known as flooding the network [43].

In order to prevent the query messages from being passed around the network indefinitely, Gnutella packets include a time-to-live (TTL) parameter - usually with a default value of 7. As the message is received by a node the TTL is decremented. If the value for the TTL is positive, then the message is forwarded to all of the connected nodes except for the node that it was received from. In addition, each message also contains a unique ID which is used to prevent messages from being sent to nodes which have already received them.

The range of the search, as dictated by the TTL, is referred to as the search horizon, and although it is an artificial limit, it is a necessary one. Without it a query would pass throughout the network until the whole network was processing the query. While in theory, flooding the network with requests to find as much information as possible is a good idea, the network

becomes swamped very quickly.

2.1.2 Multi-agent Systems

Over the past few years, the software agents deployed in today's computing world, Internet, has been exponentially increasing. Software agents are good candidates for reducing people's efforts especially where tasks become too complex to analyze for humans and they are dynamically changeable. In essence, a software agent could be defined as a software assistant that takes care of specific tasks on behalf of its user with little or no human intervention.

Multiagent Systems have always been thought of as a set of intelligent agents interacting to coordinate their knowledge, goals, skills and plans jointly to take action or to solve problems. The Multiagent paradigm can lay over on the P2P architecture, where agents embody the description of the task environments, the decision-support capabilities, the collective behavior, and the interaction protocols of each peer [42]. Each agent in multiagent system behaves according to some coordination protocols. The coordination or cooperation among the agents of a multiagent system may be achieved in various ways according to the model employed. The models that has been proposed for cooperation in this direction of the research are such as the blackboard [26], the contract net [39], and the actor [2] models. In this study, we employed a ContractNet-like custom protocol by which peer contracting space is searched efficiently.

Contract Net Interaction Protocol could enable the agents to work together, solving common goals, to fulfill ones' needs and desires. Fully automated competitive negotiation between agents through the use of contracts can be established by the interaction protocol which Contract Net specifies. In essence, an agent distribute its task among a group of agents to find a partner for its task to be done. Originally perceived by Smith in 1980, it was first applied to a simulated distributed acoustic sensor network. One of the most promising uses for Contract Net is to create an electronic marketplace for buying and selling goods. Each agent is assumed to be self-interested in the protocol, meaning that the final solution maybe be the best for the agents involved, but not for the group as a whole.

Although both agents and P2P system are currently gaining popularity, there are not more systems that combine the two [24]. Many MASs are considered to to be P2P systems due to

the fact that the agents communicate on an equal basis such as RETSINA [41] and AgentCities [48].

Internet is heterogeneous place and the agents mostly are incapable of understanding each other. Autonomous agents cooperate by sending messages and using concepts from a domain ontology where, in this work, we assume that all agents share the domain ontology and have common language to communicate between each others.

Usefulness of agent-oriented frameworks for business process development and control has been discussed in [3]. Agents are assumed to be peers in P2P agent systems [38]. From the multiagent systems perspective, Peer-Con peers are not competing but cooperating agents. Peer-Con is implemented on Java Agent Development Framework (JADE) which is for developing a multi-agent systems in compliance with the Foundation for Intelligent Physical Agents (FIPA) specifications. FIPA specifications represent a collection of standards which are intended to promote the interoperation of heterogeneous agents and the services that they can represent. By means of its peer-to-peer facilitating architecture and its rich message-handling capabilities, JADE provided us an interoperable and distributed platform.

2.1.2.1 JADE

JADE is a software framework fully implemented in Java. It allows reducing the time-to-market for developing distributed multi-agent applications by providing a set of ready and easy-to-use functionalities that comply with the standard FIPA specifications and a set of tools that supports the debugging and monitoring phases [8]. The architecture of JADE consists of AMS, DF, and Containers likely FIPA-OS. Each Container in the JADE is a running instance of its runtime environment in which several agents can reside. The set of active containers is called a Platform. The Main container, a single special container, is always active in a platform and all other containers register with it after they become active. The AMS and DF must reside in Main container and treat a service to several operations as follow 'registration', 'deletion', 'modification', and 'searching' through the DF.

In JADE, messages adhere strictly to the FIPA ACL (Agent Communication Language) standard which allows several possibilities for the encoding of the actual content. Agent Communication Language (ACL) is a standard language for agent communications recommended

and proposed by FIPA. FIPA ACL is based on the speech-act theory. Speech acts are expressed by means of standard keywords, also known as 'performatives' (for example 'ask', 'request' or 'tell'). The agent's message can then include several parameters such as 'sender' and 'receiver' of the message, the 'language' and 'ontology' (vocabulary) of the embedded content, and the actual 'content'.

2.2 Related Works

In this section, an overview of the literature on the related works is given. It includes related studies which have been trying to answer the following questions; i) How to represent business goals and capabilities in the form of process description? (Representation) ii)How to define the agreement operation among processes? (Business agreement operator) iii) What can be an efficient architecture and protocol that can facilitate such interactions? (Environment).

2.2.1 Business Process Representation

Business process is defined as a collection of interrelated tasks, which solve a particular issue. There have been many proposals to model business processes. It is challenging issue and many researches have been working on better representation of business processes in terms of different benefits.

Related to the process representation challenge, there are solutions ranging from stateless Universal Description, Discovery and Integration (UDDI) [6] and Web Services Description Language (WSDL) [13]; annotated Finite State Automaton (FSA) enabling to identify and differentiate communication partners and state transitions [49]; Petri-Net based Workflow Nets [46] ; Open Workflow Nets (oWFN) [33] to one-input transition system model allowing incomplete (or partially-defined) process descriptions [11].

Web Services are self-contained, modular business process applications that are based on the industry standard technologies of WSDL (to describe), UDDI (to advertise and syndicate), and SOAP (to communicate) where information is represented in a stateless raw string format. Web services does not support full potential interaction of processes because of its underling static binding of services. In [49], they state this as follows; "While stateless services are not

sufficient for implementing business processes, static binding of services does not use the full potential of loosely coupled systems also known as service oriented architectures.“

A business process should specify the potential execution order of operations from a collection of Web services, the data shared between these Web services, which partners are involved and how they are involved in the business process, joint exception handling for collections of Web services, and other issues involving how multiple services and organizations participate. There are some frameworks enabling collaboration among partners based on their business processes execution.

RosettaNet[15] targeted IT supply chain and developed XML-based standard electronic commerce interfaces to align the processes between IT supply chain partners on a global basis. There does not exist any pre-negotiated and uniquely named frame contracts published by RosettaNet which is not sufficient for business processes [49]. ebXML framework business enables partners to express their business capabilities (including their business processes) using trading partner profiles (CPPs) without providing any means to match these. It faces the same problem with RosettaNet.

One solution is Finite State Automaton (FSA). Finite state automaton constitute a suitable starting point to model business processes for the purpose of matchmaking [27]. Its extension by annotations (i.e. annotated FSA) is uesd to identify and differentiate communication partners enabling state transitions [49].

Other alternative is to use Petri-Nets as handled in Workflow Nets (WFNet) and Open Workflow Nets (oWFN) proposals [46][33]. oWFN is an unrestricted and enriched version of WF-Net by communication places supporting asynchronous interaction. On the other hand, a restriction on oWFN is the acyclicity assumption over Petri-Net representation. In [50], soundness and equivalence of Petri-Nets and annotated Finite State Automata has been shown in the context of Service Oriented Architectures(SOA).

The authors of [11] have proposed another business process representation based on “one-input transition system“ model. A distinguishing property of this approach is to allow incomplete (or partially-defined) process descriptions (i.e. some states are allowed to be unreachable). Also, cyclic state transitions are not needed to be treated in a special way since reachability to goal state is the only concern to decide on existence of a match instance among

processes.

In the following subsections, brief information about representations and technologies for alternative business process models such as BPEL, RosettaNet, ebXML, FSA and Workflow-Nets are provided.

2.2.1.1 BPEL

Business Process Execution Language For Web Services (BPEL4WS or BPEL, for short) allows specifying business processes and how they relate to Web services [21]. This includes specifying how a business process makes use of Web services to achieve its goal, as well as specifying Web services that are provided by a business process. Business processes specified in BPEL are fully executable and portable between BPEL-conformant environments. A BPEL business process interoperates with the Web services of its partners, whether or not these Web services are implemented based on BPEL. Finally, BPEL supports the specification of business protocols between partners and views on complex internal business processes.

2.2.1.2 RosettaNet

The RosettaNet framework enables supply chain business partners to execute interoperable electronic business (e-business) processes by developing and maintaining PIP implementation guidelines. RosettaNet distributes PIPs to the trading partners, who use these guidelines as a road map to develop their own software applications. PIPs include all business logic, message flow, and message contents to enable alignment of two processes. In order to do electronic business within the RosettaNet framework, there are a number of steps the partners have to go through. First, the supply chain partners come together and analyze their common inter-company business scenarios (i.e., public processes), that is, how they interact to do business with each other, which documents they exchange and in what sequence. These inter-company processes are in fact, the “as-is“ scenarios of their way of doing business with each other. Then they re-engineer these processes to define the electronic processes to be implemented within the scope of the RosettaNet Framework.

2.2.1.3 ebXML

ebXML is a set of specifications that together enable a modular electronic business framework. The vision of ebXML is to enable a global electronic marketplace where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML-based messages[36].

An ebXML Registry provides a set of services that manage the repository and enable the sharing of information between Trading Partners. Note that business Processes, CPPs, business document desciptions and core components are published and retrieved via ebXML Registry Services. A trading partner may discover other trading partners by searching for the CPPs in the registry. The ebXML Messaging Service is used as the transport mechanism for all communication into and out of the Registry.

2.2.1.4 Finite State Automaton

A finite-state automaton is a device having a finite number of states and can be in one of its states. It can switch to another state which is called a transition in certain conditions . When the automaton starts working, it can be in one of its initial states. There is also another important subset of states of the automaton: the final states. If the automaton is in a final state when it stops working, it is said to accept its input. The input is a sequence of symbols. The interpretation of the symbols depends on the application; they usually represent events, or can be interpreted as "the event that a particular data became available". The symbols must come from a finite set of symbols, called the alphabet . If a particular symbol in a particular state triggers a transition from that state to another one, that transition is labeled with that symbol. The labels of transitions can contain one particular symbol that is not in the alphabet. A transition is labeled with E (not present in the alphabet) if it can be traversed with no input symbol.

It is convenient to present automata as directed graphs. The vertices denote states. They are portrayed as small circles. The transitions form the edges - arcs with arrows pointing from the source state (the state where the transition originates) to the target state. They are labeled with symbols. Unless it is clear from the context, the initial states have short arrows that point to them from "nowhere". The final states are represented as two concentric circles.

2.2.1.5 Workflow Nets

Workflows are case-based, i.e., every piece of work is executed for a specific case. Examples of cases are a mortgage, an insurance claim, a tax declaration, an order, or a request for information. Cases are often generated by an external customer. However, it is also possible that a case is generated by another department within the same organization (internal customer). The goal of workflow management is to handle cases as efficient and effective as possible. A workflow process is designed to handle similar cases. Cases are handled by executing tasks in a specific order. The workflow process definition specifies which tasks need to be executed and in what order. Alternative terms for workflow process definition are: 'procedure', 'flow diagram' and 'routing definition'.

Following quotes from the author of [46] indicates that the workflow can be modeled by Petri nets.

" In the workflow process definition, building blocks such as the AND-split, AND-join, OR-split and OR-join are used to model sequential, conditional, parallel and iterative routing." "Clearly, a Petri net can be used to specify the routing of cases. Tasks are modeled by transitions and causal dependencies are modeled by places. In fact, a place corresponds to a condition which can be used as pre- and/or post-conditions for tasks. An AND-split corresponds to a transition with two or more output places, and an AND-join corresponds to a transition with two or more input places. OR-splits/OR-joins correspond to places with multiple outgoing/ ingoing arcs."

2.2.2 Process based Inter-organizational Agreement

Automated contracting systems require a mechanism for identification of inter-organizational agreements through process elaboration. To decide on whether given capabilities (or processes) turnout to a business agreement or not, one needs to describe a generic agreement operator that considers peer capabilities together with their execution costs.

Related to the identification of process based inter-organizational agreement challenge, in [49], non-emptiness of the intersection of languages defined by annotated FSAs representing individual processes has been defined for the existence check of process level agreements.

In their study, matchmaking is defined in terms of being able to be backup of a process. The approach seems to be restrictive, since companies usually look for a partner firm having different services rather than doing the same business.

In [11], agreement operation is defined formally as a state-level merge of two processes followed by a reachability test for the goals. It is understood that via "one-input transition system" representation and merge/synthesis operator descriptions supported by domain specific ontology, it is possible to describe a core process matchmaking mechanism. Also, centralized matchmaking among multiple processes can be achieved in polynomial-time. On the other hand, concurrent merge of more than two processes and costing of transitions are two missing properties that are required for establishment of a proper contracting system.

Other matchmaking approaches are based on semantic information [9], [34]. In [42], a language describing the functional aspects as well as the messages and their parameters based on a domain specific ontology is introduced. DAML-S [5] uses workflow aspects as well as the functional semantic description of the service within the matchmaking. The web service offering language (WSOL) [45] provides additional semantic meta information to increase precision of the service discovery and the classes of services are modeled by specifying functional constraints, QoS, simple access rights, price, and other constraints in addition to a WSDL description.

The main draw back of semantic annotation is the necessity of a common ontology used for annotating and querying services. Unfortunately, no such ontology currently is in place[49].

Note that, different from matchmaking environment developed in [11], a process contracting environment is defined in [12]. We have to define the difference between process matchmaking and contracting in our context. In process matchmaking, costs of peer capabilities are not considered and an interaction result between two processes is either match or no-match. However, in process contracting, capabilities have associated cost values and many-toone engagement of peers for cooperative joint business process execution is possible.

2.2.3 Matchmaking/Contracting Environment

The proposed architecture in [49] is a centralized client-server approach realized through a matchmaking engine [7]. In fact, to the best of our knowledge there is only the work

of [11] used for business process matchmaking implemented based on P2P protocols and architecture. A decentralized process matchmaking environment based on P2P architecture and Gnutella protocol is proposed in their work. The processes are modeled as one-input transition systems augmented by goal state descriptions. A polynomial-time algorithm for handling matchmaking of peer process encounters is developed.

The question of whether there is any single environment model being more efficient in terms of process matchmaking performance among different unstructured and decentralized digital environments (e.g. P2P, small-world, grid etc.) is investigated in the study of [10]. By the end of this macroscopic investigation, they conclude that there is no domination of one model over another.

CHAPTER 3

The OVERLAY

General system descriptions including architecture, technologies used, peer behavior and interaction protocol are explained in the following subsections.

3.1 System Architecture

The Peer-Con is realized on an agent-based development environment JADE which constitutes a middleware for our P2P setup. The system has four layers (see Figure 3.1). Physical layer contains the internet and wireless environments. The virtual machine layer supports different scales of Java-based solutions ranging from standard and enterprise editions of Java-2 to its micro edition supporting Connected Limited Device Configuration (CLDC) for low power and memory, mobile devices and Personal Java. The third layer is JADE on which we create the peers. Containers are basic logical units of residence of this layer. They provided us platform independent distributed peer deployment and execution. In a typical JADE installation, there is a main container holding a specialized agent called Directory Facilitator (DF). In the implementation, the DF agent is directly used as the BootStrapServer of the P2P setup. The top layer, Peer-Con is an overlay on JADE. This layer supports two basic peer capabilities: i) Process representation and operator (merge and cost calculation) ii) Peer interaction protocol. The process representation and the merge operator are implemented in Java. We used basic syntax of standard IEEE FIPA-ACL in protocol implementation.

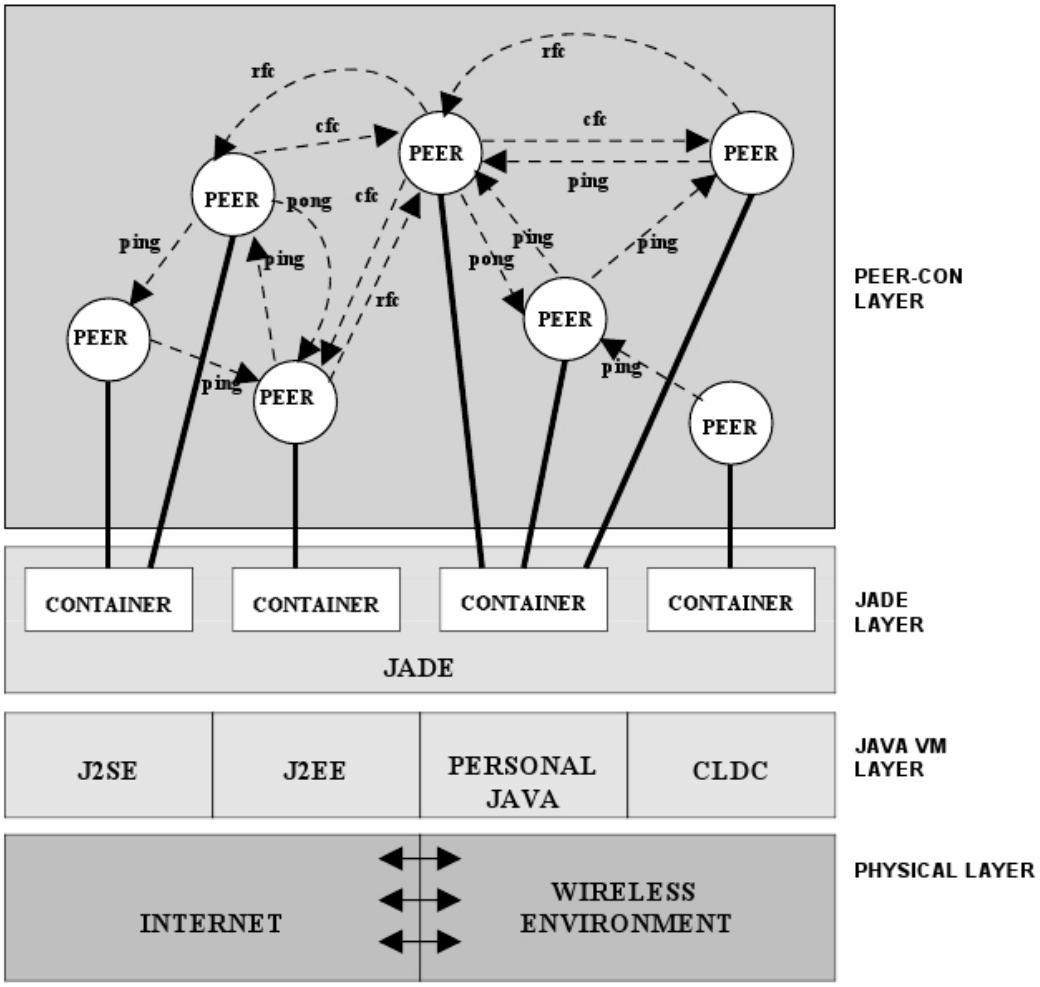


Figure 3.1: Peer-Con layer and its environment.

3.1.1 Peer Interactions

Peers are implemented as cooperative agents that may reside in containers of different distributed machines. Each peer is owned by and interacts with one user and one or more peers constituting its immediate neighbors, respectively. Communication is achieved via four basic message types: ping, pong, cfc (call for contracting) and rfc (response for contracting). Ping and pong messages are used for network connectivity awareness of peer; cfc and rfc are developed to realize low cost process contracting among peers. The main responsibility of a peer owner is to assign a process description (P) to its peer (called originator). Besides from P, the owner (or user) feeds two other inputs to his/her peer: i) An acceptable upper limit for process contracting cost (u) ii) Time-To-Live (TTL) value showing the upper limit for the number of hops that P2P system will be searched for contracting, see Figure 3.2. Simply, TTL defines the radius of search for contracting. Goal of the originator peer is to find out one or more peers residing in the search radius TTL that can jointly execute process P with total cost less than u. Other peers (called cooperating peers) handle incoming requests in a cooperative way. Any peer in the system may play the role of originator or cooperating peer at any time.

After the broadcast of the originator peer including P, u and TTL values to the cooperating peers in its neighborhood, they handle the received query in forward mode. In this mode, on each hop to the next peer, the current TTL value of the process description is decremented by 1. Also the set L holding the peer IDs along the path is expanded by the ID of current peer. Initially, the set L only includes the originator peer itself. Once the total cost less than u value is attained, the cooperating peers in forward mode turns into backward mode during which the set L is pruned into minimal set $L_{min} \subseteq L$ with total process cost being less than u, cooperatively. In order to avoid from long waiting for responses, the originator peer disregards the rfcs received later than a constant timeout period (v). Among those received timeout-filtered rfcs, the originator peer selects the rfc with minimum-cost merged description and returns it to its owner. Note that in forward mode, once TTL value becomes zero the the process description simply is not passed any further among peer and becomes obsolete. In Figure 3.3, an example cfc sequence followed by an rfc sequence generated along a search cycle is depicted. Formally, a sequence generated along a search cycle is defined by $(cfc)^n(rfc)^n$ where $TTL \geq n \geq 0$. Cfc and rfc sequences describe the forward and backward modes of the search cycle, respectively. An instance $n = 0$ means the originator peer already finds an acceptable

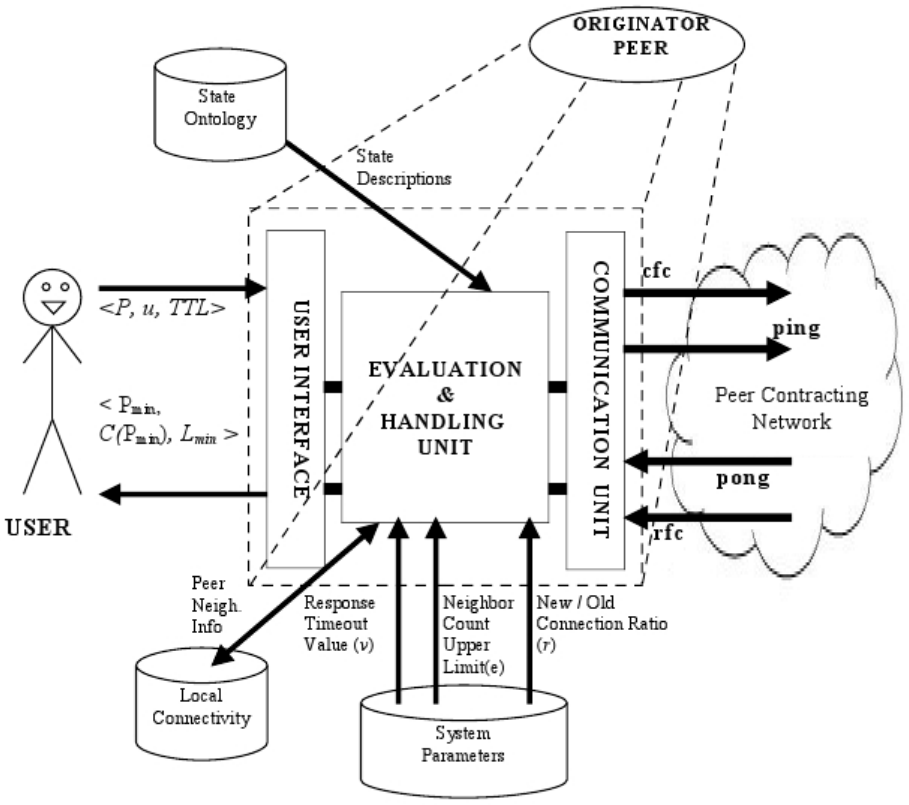


Figure 3.2: User, originator peer and contracting network interactions.

solution below the cost u. In forward mode, on each hop to the next peer, the current TTL value is decremented by 1 and the set L holding the peer IDs along the path is expanded by the ID of current peer. In backward mode, L is pruned into minimal set L . L min with total process cost being less than u. Our aim for pruning is to reduce peer neighborhood that results in lower traffic load on the system. After taking user inputs, the originator peer broadcasts the user input to every peer in its neighborhood as cfc request represented by $\langle \Pi, u, TTL, L \rangle$. Initially L=P_{self}. The cfc requests are processed by other peers in the system and one or more rfc results are returned back to the originator peer. Like cfc, an rfc is constituted by three components: i) A merged process description $(\Pi)^*$ ii) Cost of the description $(C(\Pi^*))$ iii) The IDs of peers contributing to the description (L). An rfc is represented by triplet $\langle (\Pi)^*, C((\Pi)^*), L \rangle$. Table 3.1 holds an example instance of search cycle defined by cfc and rfc messages. Type of the user generated message in the first row is assumed to be cfc with L value initialized to O. By the end of search cycle, user receives a contracting peers set P1, P3 whose total cost of execution $C(\Pi_{1..3})$ is less than or equal to 5.0. Note that the fifth row

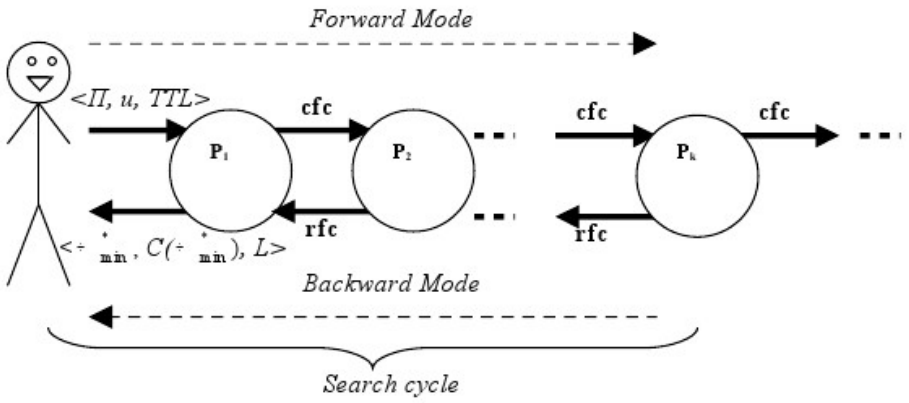


Figure 3.3: An example search-for-contracting cycle with cfc and rfc messages.

of the table has a cfc entry implying the continuation of the search for further cycles. As a consequence of multiple search cycles due to the same cfc, the originator peer may receive more than one rfc results. In the example, multiple rfcs are not considered. On the other hand, depending on TTL value, the search time can be too long. In order to avoid from long waiting for responses, the originator peer disregards the rfcs received later than a constant timeout period (v). Among those timeout-filtered rfcs, the originator peer selects the rfc with minimum-cost merged description (i.e. $(\Pi_{min})^*$) and returns it to its owner.

Table 3.1: An example search cycle instance defined by cfc and rfc messages.

Sender (P_i)	**Receiver** (P_j)	**Message Type** (*cfc* or *rfc*)	**Message** < *Π, u, TTL, L*> or < Π^*, $C(\Pi^*)$, *L*>
User	P_1	cfc	< Π_1, 5.0, 5, Ø >
P_1	P_2	cfc	< Π_1, 5.0, 5, {P_1} >
P_2	P_3	cfc	< $\Pi_{1.2}$, 5.0, 4, {P_1,P_2} >
P_3	P_2	rfc	< $\Pi_{1.3}$, $C(\Pi_{1.3})$, {P_1,P_2,P_3} >
P_3	P_4	cfc	< $\Pi_{1.3}$, 5.0, 3, { P_1,P_2,P_3} >
P_2	P_1	rfc	< $\Pi_{1.3}$, $C(\Pi_{1.3})$, {P_1,P_3} >
P_1	User	rfc	< $\Pi_{1.3}$, $C(\Pi_{1.3})$, {P_1,P_3} >

An important problem with our search cycle mechanism is the possibility of message explosion due to redundant message looping. A message instance may bounce back to peer itself if it is not under control. Note that, looping is not a problem with rfc type of messages since peers memorize and decide on using their message return path. For a cfc message, loops are identified by checking whether the peer itself is in the current contracting set L or not. Loop

identification results in dropping the received cfc message. Other two message types used in communication are ping and pong. These messages provide network awareness and are initiated by peer itself. Ping message is used to check aliveness of peer's current neighbors. It can also discover new peers in the contracting network outside its neighborhood. The extent of discovery is specified by a system parameter neighbor count upper limit (e), see Figure 3.2. Each new ping message ID is added to a look-up table. Pong message is response of a ping message and it contains the information about aliveness of the sender peer. Loops that may occur during ping pong messaging are identified by using message ID look-up table. The identified loops are eliminated by simply killing the message.

3.2 Single Peer Implementation

Evaluation and Handling Unit is the core functional unit of peer. It contains three basic components: i) CFC Handler ii) RFC Handler and iii) Neighborhood Organizer (see Figure 3.4). CFC Handler either receives cfc instance from a neighbor peer or from its owner. Merge of process descriptions (see Definition-3), process cost calculation (i.e. shortest path calculation), rfc result preparation, cfc update and send operations, recording of cfc info, loop elimination and TTL and cost upper limit u checks are basic functions of CFC Handler.

Peer neighborhood info is used to send updated cfcs to neighbors. Pseudocode for CFC Handler is given in Figure 3.5. In Figure 3.6, an example of cfc message implementation using IEEE FIPA-ACL syntax is is shown.

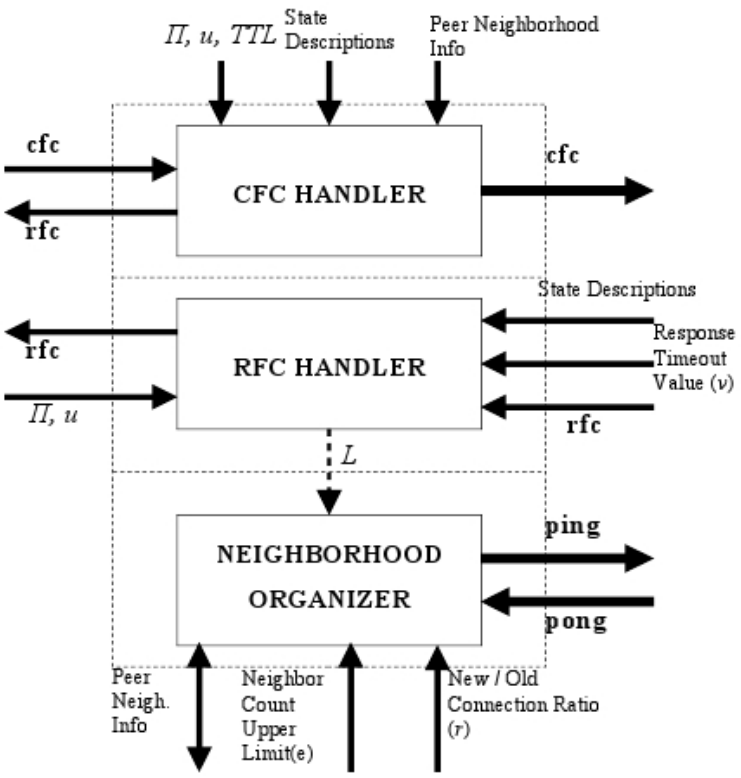


Figure 3.4: Evaluation and Handling Unit.

```
-Initialize CFC () /* by user */

 L = P_self;
 If C(Π_self) <= u  then send RFC message to the user;
 cfc = Construct_CFC(Π_self, u, TTL, L);
 Forward cfc to neighbors except its sender;
 Keep cfc message in the lookup table;
 Initialize timeout for cfc;


-Receive CFC (cfc) /* from a neighbor peer */

 If  cfc message is seen before then Drop the message;
 else If P_self  is in L then Drop the message;
 else{
    Π_new= Merge(Π_cfc, Π_self);
    L_new = L ∪ P_self;
    If C(Π_new) <= u  then send RFC message to its sender;
    Decrease TTL;
    If TTL > 0 then
    {
       cfc = Construct_CFC(Π_new, u, TTL, L_new);
       Forward cfc to neighbors except its sender;
       Keep cfc message in the lookup table;
    }
 }
```

Figure 3.5: Pseudocode for CFC Handler.

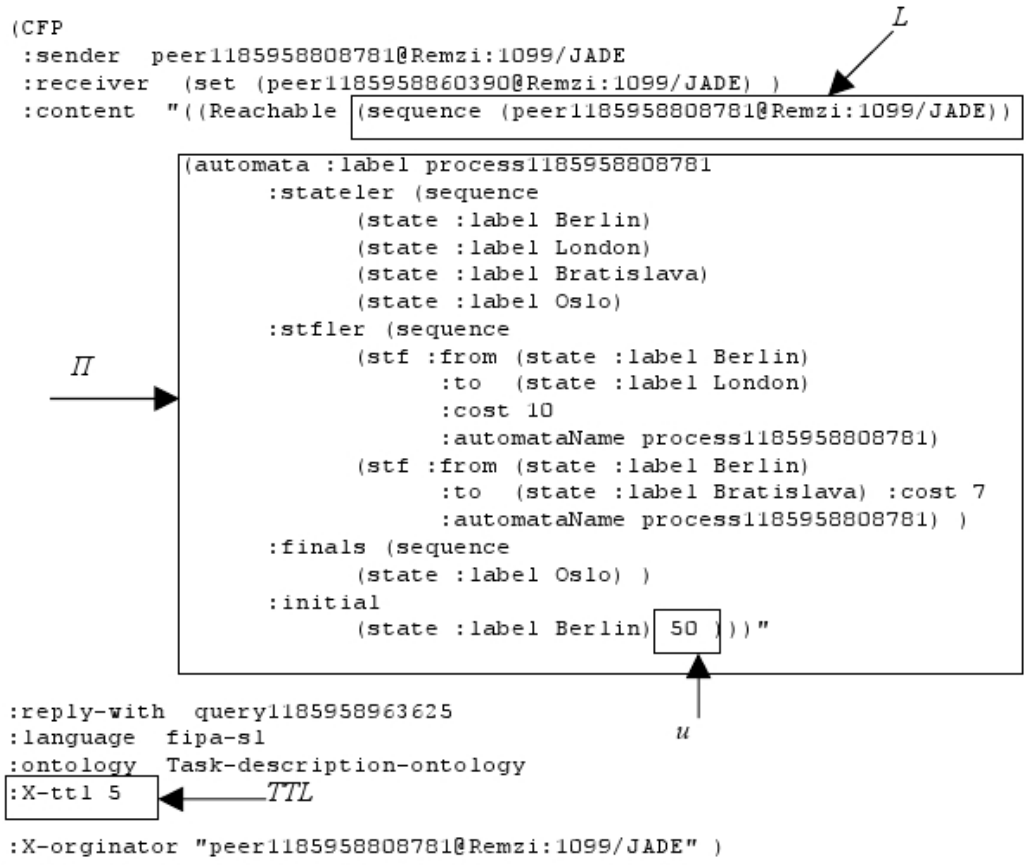


Figure 3.6: CFC Message Sample.

The cfc message, in our context, is used to initiate the process contracting operation. Sender of the Call For Proposal (CFP) message looks for a contract for its process description. The content part of the message is written in standard fipa-sl language. The content includes the query for checking possible contracts among peers. The ontology attribute is used to decide on the related state-domain of the process. Goal of the RFC Handler is to decide on whether the peer itself is a member of minimum cost process executing team along the search cycle or not. This is done by checking the cost of including and excluding the peer process. Team list membership check, extraction of transition system description from the merged description, rfc update and send operations, process originator check and cost calculation, cost upper limit u checks, and rfc collection and user display are basic functions of RFC Handler. Pseudocode for RFC Handler is given in Figure 3.7.

The next example shows rfc message implementation using the IEEE FIPA-ACL syntax. A sample of rfc message is depicted in Figure 3.8.

```
-Receive RFC (rfc) /* from peer */

  If rfc message is seen before then Drop the message;
  Else {
     If  P_self  is in L then Drop the message;
     Else If Is_OrginatorOf(rfc) is true
     {
       Π_ex = Extract(Π_rfc, Π_self);
       If C(Π_ex) <= C(Π_rfc) then {
          L_ex = L \ P_self;
          rfc_new = Construct_RFC(Π_ex, C(Π_ex), L_ex);
          rfc = rfc_new;
       }
       Show rfc to the user;
     }
     Else{
       Π_ex = Extract(Π_rfc, Π_self);
        If C(Π_ex) <= C(Π_rfc) then {
          L_ex = L \ P_self;
          rfc_new = Construct_RFC(Π_ex, C(Π_ex), L_ex);
          pass rfc_new to next peer
        }
        Else pass received rfc to next peer;
     }

  }
```

Figure 3.7: Pseudocode for RFC Handler.

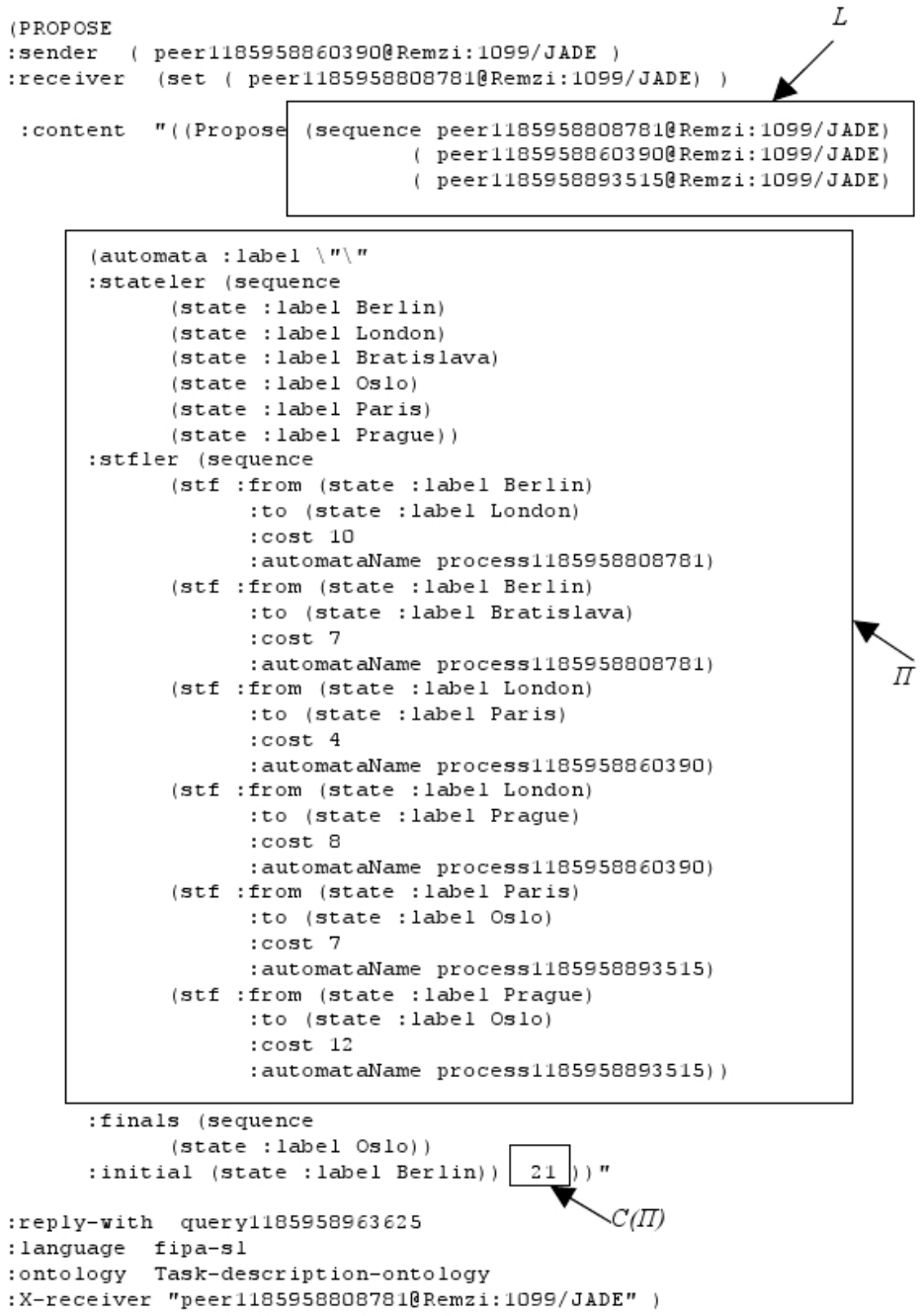


Figure 3.8: RFC Message Sample

The rfc message is the answer for cfc message. The content part of the rfc message holds the contracting team set, the merged process description and its cost. The ontology attribute shows the process' state domain.

Neighborhood Organizer is responsible from generating ping, pong messages, tracking response timeout, finding minimum cost rfc in the collected rfc set and managing peer local connectivity. Neighbor addresses are kept in a local look-up table. There are three events that trigger neighbor look-up table update: i) Pong message receipt ii) Formation of a new contracting team and iii) Non-replied successive ping messages. Once a pong message is received and its sender is not a neighbor peer and also current Neighbor Count Upper Limit (e) is not reached, sender peer ID is added to the table. We made a simple assumption that members of new contracting team are also good candidates for later contracting. Therefore, once a new contracting team is formed, they are combined with the current neighbor set. The combination is done based on New/Old Connection ratio (r). r = 0 means current neighbor population is kept as it is. However, r = 1 means current neighbor population is totaly replaced by the new one. The third event happens when three successive ping messages are sent to a neighbor peer but they are not replied by any pong message. In such cases, the peer is removed from the look-up table. Pseudocode for Neighborhood Organizer is given below.

The example messages depicted in Figure 3.9 are ping pong message implementations in IEEE FIPA-ACL syntax.

Ping message is used to be aware of peer neighborhood structure. It contains a built-in sender and receiver part showing the current and destination peer id's. The ping message is replied with a message hold in the reply-with part of the Query-Ref performative. The user-defined attribute X-originator shows the original source of the ping message. The user-defined attribute X-ttl holds the value of time-to-live in hop unit.

Pong message is the answer for the ping message. It is used to inform the message originator about aliveness of the pinged peer. Pong message also holds the original ping id. The difference between receiver and x-receiver attributes is the former describes the neighbor peer which will receive the current pong message and the latter is the target (or final) receiver of the message.

```
-Send Ping
    Send ping to its all neighbors
    Keep ping message in the lookup table;

-Receive Pong (pong)
    If pong message is seen before then Drop the message
    Else update connectivity;

-Timeout control
    If timeout is reached
       Find the one with minimum cost from collected RFCs;
       update connectivity according to set L_rfc_min;
```

(a) Pseudocode for Neighborhood Organizer

```
(QUERY-REF
:sender  ( peer1185958860390@Remzi:1099/JADE )
:receiver  (set ( peer1185958808781@Remzi:1099/JADE ) )
:reply-with  ping1185958867906
:X-ttl 5
:X-orginator "-host1185958860390@Remzi:1099/JADE" )
```

(b) An example ping message.

```
(INFORM
:sender  ( peer1859588808781@Remzi:1099/JADE )
:receiver  (set (peer1785958867789@Remzi:1099/JADE ) )
:in-reply-to  ping1185958867906
:X-ttl 5
:X-originator "peer1859588808781@Remzi:1099/JADE"
:X-receiver "peer1185958860390@Remzi:1099/JADE")
```

(c) An example pong message.

Figure 3.9: Pseudocode for Neighborhood Organizer and ping/pong examples.

CHAPTER 4

OPERATORS AND DESCRIPTIONS USED IN THE OVERLAY

In the following, the *process* and *merge* operator descriptions that constitute the basis for the system are presented. The formal definition of business process which a peer uses to describe its internal capabilities is given. Initially, each peer in the system is assigned a process description representing the organization's possibly integrated capabilities together with their costing. It could be understood that a peer can achieve its goals without the need of any peer cooperation if the minimum execution cost of its process, calculated as the shortest path of the given graph from the starting node to any one of the final nodes, is smaller than ∞. If a peer could not capable to satisfy its goal(s), it searches through the network for a cooperating peer set. The cooperating set can contract for a peer's process to be executed partially or completely. If a peer has a process that could contribute the originator peer's process description, it means that this peer can be in the cooperating peer set for the originator peer. The test whether any given two processes turnout to an agreement or not is checked by merging two processes followed by a reachability test for goals states.

4.1 Description of Business Process

Definition 1: A *process* is a triplet $\langle G, I, F \rangle$ where G is an *arc labeled multidigraph* defined as a 6-tuple G=(Σ_E, V, E, s, t, w) where V is a set of nodes representing process states, E is a multiset of arcs representing alternative interstate transitions, Σ_E is the domain that arcs take value, in our case it is R^+, s: E $\longrightarrow$ V is the mapping that defines the source state of a transition, t: E $\longrightarrow$ V is the mapping that defines the target state of a transition and w: E $\longrightarrow$ Σ_E is the function defining the cost of state transitions. I $\in$ V is a distinguished node representing starting state of the process and F $\subseteq$ V - $\emptyset$ is the set describing the nonempty final

states set of the process.

The nodes represent a state (or a stage) of the process at which for example some part of the production is realized or even outsourced by the organization. Possible switches that are allowed among production stages are defined by the binary relation E together with its costing defined by the function w. The step from which, say production, starts is defined by a single distinguished state s. Note that in practice, this state can be a dummy initial state without any cost and assumed to be common to all peers in the system. Similarly, one or more final states showing the end of a production sequence is represented by the set F.

4.2 Description of Match Operator

Following a formal description of process, the first task of an organization is nothing but to calculate the minimal cost of the whole process by considering the switching costs. Mathematically, it is nothing but to calculate the shortest path of the given graph from the starting node to any one of the final nodes. Clearly, for a given graph G, we may have more than one minimum cost path reaching any of the final states. In fact, shortest path finding problem is known to be a single peer solvable simple problem that can be solved in polynomial-time [23] and does not require any peer cooperation. On the other hand, what makes more than one peer setup to be a realistic is the inherent decentralized, dynamically changing, partially observable and possibly untrusted nature of the digital environment. We have to point out that the proposed system performs well especially when efficient off-line planning is not possible due to frequent changes in process descriptions. In the industrial process contracting context, it corresponds to frequent changes in usable technology and emergence of alternative outsourcing possibilities for different task in design, production, marketing etc. Another important factor that makes our process representation a realistic one is the permission to incomplete process descriptions. For example, when none of the final stages of production is reachable (i.e. the total process cost is ∞ and self production is not possible), a company can still find a partner(s) from the system to realize its production.

Definition 2: Two processes P and Q are called *capability disjoint* iff $P_{G(E)} \cap Q_{G(E)} = \emptyset$.

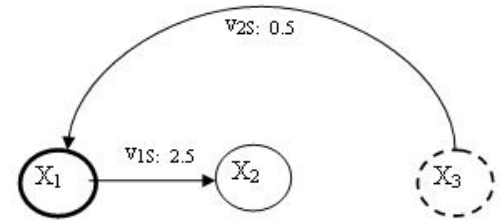
Definition 3: *Merge* of a given two capability disjoint processes P and Q defines a new process T=Merge(P, Q) such that $T_G = P_G \cup Q_G$ defines node level union of given arc labeled multidigraphs P_G and Q_G; $T_{G(\Sigma_E)} = R^+$; $T_{G(s)}$ and $T_{G(t)}$ mappings are defined by union of

corresponding source and target node description mappings $P_{G(s)}$, $Q_{G(s)}$ and $P_{G(t)}$, $Q_{G(t)}$, respectively; similarly the edge cost function $T_{G(w)}$ is defined by corresponding edge functions $P_{G(w)}$ and $Q_{G(w)}$; $T_I = P_I$ and $T_F = P_F$.

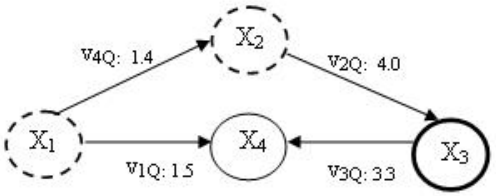
According to Definition-3, Merge(P, Q) ≠ Merge(Q, P). In practice, this asymmetry can be interpreted as: in Merge(P, Q) owner peer of process P which is looking for cooperating peer(s) for its say production planning is the main source of the demand and the role of the peer being owner of process Q defining another production plan is the potential supplier of such demand. In Merge(Q, P), however, the demander/supplier roles of the peers are just opposite. In the same definition, graph nodes are defined from the same domain but the edges are guaranteed to be from different domains by the capability disjoint processes assumption. The semantics of this is: the costs of switching among production stages or outsourcing are defined by peers and related capabilities are owned by them. Furthermore, possible contracting between the demander and supplier peers with associated capability disjoint processes say P and Q occurs only if the shortest path cost for P is greater than the shortest path cost for Merge(P, Q).

4.3 Illustrative Example of Definitions

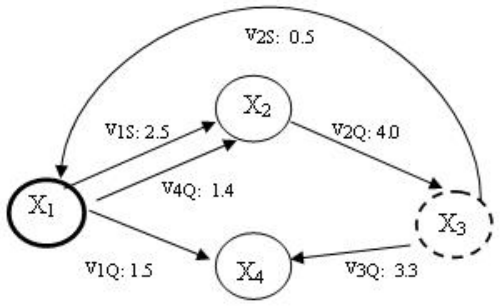
In Figure 4.1, circles depict process states ($x_i \in X$) and arrows represent the capabilities and their costs. For example, label v_{1S}:2.5 represents the capability v_{1S} of peer whose process description is constituted by system S and execution cost of the capability is 2.5. The tick circles and dashed circles show the initial and end states, respectively. Goal of the peer is to reach one of its goal states from its starting state. In Figure 4.1-a, we have S=(G_S, I_S, F_S) and $G_S = (\Sigma_E, V_S, E_S, s_S, t_S, t_S)$ where $V_S = \{x_1, x_2, x_3\}$, $E_S = \{v_{1S}, v_{2S}\}$, $s_S = \{((v_{1S}) \rightarrow x_1)$, $((v_{2S}) \rightarrow x_3)\}$, $t_S = \{((v_{1S}) \rightarrow x_1), (v_{2S}) \rightarrow x_1)\}$, $w_S = \{(v_{1S} \rightarrow 2.5), (v_{2S} \rightarrow 0.5)\}$, $I_S = \{x_1\}$, and $F_S = \{x_3\}$. Similarly in In Figure 4.1-b, we have Q=(G_Q, I_Q, F_Q) and $G_Q = (\Sigma_E, V_Q, E_Q$, s_Q, t_Q, t_Q) where $V_Q = \{x_1, x_2, x_3, x_4\}$ $E_S = \{v_{1Q}, v_{2Q}, v_{3Q}, v_{4Q}\}$ $s_Q = \{((v_{1Q}) \rightarrow x_1), ((v_{2Q})$ $\rightarrow x_2), ((v_{3Q}) \rightarrow x_3), ((v_{4Q}) \rightarrow x_1)\}$ $t_Q = \{((v_{1Q}) \rightarrow x_4), ((v_{2Q}) \rightarrow x_3), ((v_{3Q}) \rightarrow x_4), ((v_{4Q}) \rightarrow$ $x_2)\}$, $w_Q = \{((v_{1Q}) \rightarrow 1.5), ((v_{2Q}) \rightarrow 4.0), ((v_{3Q}) \rightarrow 3.3), ((v_{4Q}) \rightarrow 1.4)\}$, $I_Q = \{x_3\}$, $F_S = \{x_1$, $x_2\}$. The costs of the processes in Figure 4.1 (a) and (b) are ∞ since none of their goal states are reachable. However, the process of merged systems in Figure 4.1 (c) has the shortest path is {v4Q, v2Q} with 1.4 + 4.0 = 5.4.



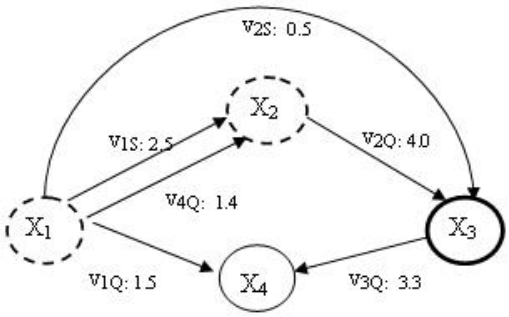
(a) Process S(G_S, I_S, F_S).



(b) Process Q(G_Q, I_Q, F_Q).



(c) Process (T=*Merge*(S, Q)).



(d) Process (R=*Merge*(Q, S)).

Figure 4.1: Example process descriptions and merge operation result.

CHAPTER 5

APPLICATIONS

5.1 Driving Route Calculation on Web Maps

Most of today' s business problems include significant spatial components supported by Geographical Information System (GIS) solutions and the role of GIS in business intelligence is gaining more importance [1]. Specifically, when we consider enterprise resource planning and supply chain management requirements of factories and manufacturing systems, applications like vehicle tracking and dispatch, route analysis, warehouse operations and routing and scheduling facilitated by GIS allow decision-makers to perform at higher level of efficiency. Maps are core components of GIS tools [30] and provide different type of spatial information to the above GIS-based business processes. Among many other alternative definitions in literature, according to International Cartographic Association (ICA) Working Group on Cartographic Definitions, "Map is a holistic representation and intellectual abstraction of geographic reality intended to be communicated for a purpose or purposes, transforming relevant geographical data into an end-product which is visual, digital or tactile. " [4]. Web mapping on the other hand, is the process of development and delivery of maps on World Wide Web. The use of web as a medium enabled new opportunities like real-time maps, cheaper update of data and software, sharing of geographic information and distributed data sources. Different from early static web maps, today' s interactive and dynamic web maps support better functionality and usability. For example, a special type of web maps, called mobile maps, can both be visualized and interacted via mobile computing devices, like PDA, mobile phone, GPS etc. [47]. There are many alternative commercially available web mapping services including MapQuest, MapsOnUs, MapBlast, Yahoo Maps, Map.com, Lycos Maps, Expedia Maps etc. They mainly use client-server technologies like web server, spatial databases, CGI,

web application servers and Web Mapping Servers (WMS) on server side and web browsers, Document Object Model (DOM) support, Scalable Vector Graphics (SVG) support, Java support and web browser plugins on client side. Among the services, MapQuest service is the most widely used one providing a centralized route calculating module for extracting crossing information on a map. In this section, our aim is to show the use of Peer-Con in realizing driving route calculation on web maps supporting GIS-based logistics operations. Routing in logistics context is the operation which aims at minimizing all kind of costs of travel including time, distance, effort, money or combinations of them involved in transportation of goods (or people) from one location to another [32]. Different from centralized, client-server solution offered by MapQuest driving route service [35], our decentralized P2P routing solution has a fault-tolerated architecture and user definable costing for better distributed route cost calculation. Note that the efficiency of our approach emerges especially when local map information of peers is frequently changing and is updated dynamically by users through spatial exploration at runtime.

Table 5.1: Peer-Con concepts and their interpretations in web mapping context.

Peer-Con Terminology Web Mapping Terminology	***Web Mapping Terminology***
State	Spatial Location
Capability	Connection between two locations
State transition function	Location connectivity map
Initial state	Source location
Goal state(s)	Destination location(s)
Capability cost	Cost (e.g. distance or transportation) of connection between two locations
Process	Local map with destination location(s) info One input transition system Local map
Capability disjoint one input transition systems	Any two local maps
Goal equivalent processes	Two local maps with destination location(s) being identical
Merge operation	Unification of two local maps
Behavior induced by a capability sequence	A route starting from initial location ending at any location
Cost of a process	The minimum cost route starting from initial location to any of the destination location

At this point, we need to establish analogies between our former business process related definitions and the concepts of the web mapping domain (see Table 5.1).

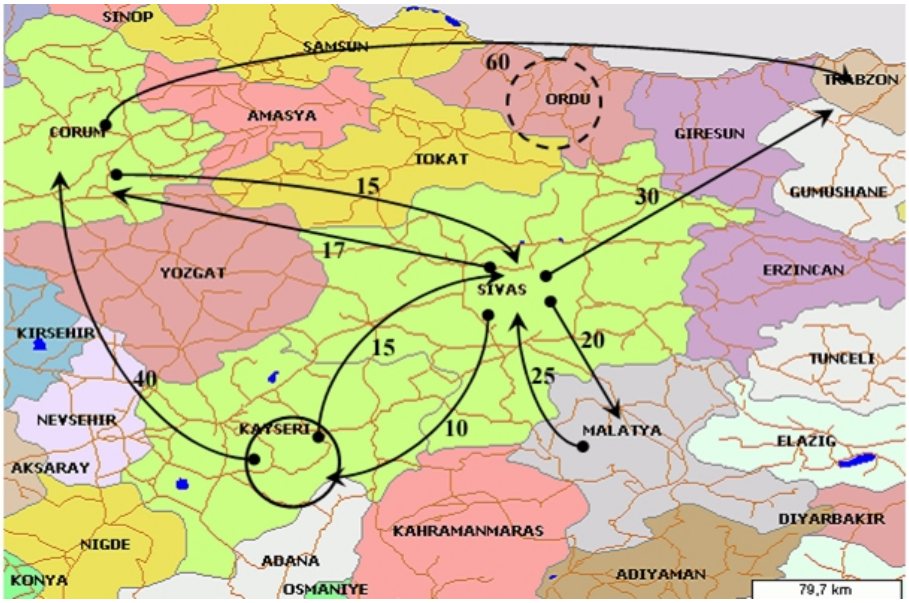


Figure 5.1: Road map of eastern part of Turkey and example hypothetical transportation costs between cities.

Based on Table 5.1, the driving road map of eastern part of Turkey [29] in Figure 5.1 can be defined in the form of a process definition S=(G_S, I_S, F_S) and G_S = (Σ_E, V_S, E_S, s_S, t_S, t_S) where V_S = {Corum, Kayseri, Malatya, Sivas, Trabzon}, E_S = {$v_{Sivas,Corum}$, $v_{Sivas,Trabzon}$, $v_{Sivas,Kayseri}$, $v_{Sivas,Malatya}$, $v_{Corum,Trabzon}$, $v_{Corum,Sivas}$, $v_{Kayseri,Sivas}$, $v_{Kayseri,Corum}$, $v_{Malatya,Sivas}$},

s_S = {(($v_{Sivas,Corum}$) → Sivas), (($v_{Sivas,Trabzon}$) → Sivas), (($v_{Sivas,Kayseri}$) → Sivas), (($v_{Sivas,Malatya}$) → Sivas), (($v_{Corum,Trabzon}$) → Corum), (($v_{Corum,Sivas}$) → Corum), (($v_{Kayseri,Sivas}$) → Kayseri), (($v_{Kayseri,Corum}$) → Kayseri), (($v_{Malatya,Sivas}$) → Malatya) }, t_S = { (($v_{Sivas,Corum}$) → Corum), (($v_{Sivas,Trabzon}$) → Trabzon), (($v_{Sivas,Kayseri}$) → Kayseri), (($v_{Sivas,Malatya}$) → Malatya), (($v_{Corum,Trabzon}$) → Trabzon), (($v_{Corum,Sivas}$) → Sivas), (($v_{Kayseri,Sivas}$) → Sivas), (($v_{Kayseri,Corum}$) → Corum), (($v_{Malatya,Sivas}$) → Sivas) }, w_S = { ($v_{Sivas,Corum}$ → 17), ($v_{Sivas,Trabzon}$ → 30), ($v_{Sivas,Kayseri}$ → 10), ($v_{Sivas,Malatya}$ → 20), ($v_{Corum,Trabzon}$ → 60), ($v_{Corum,Sivas}$ → 15), ($v_{Kayseri,Sivas}$ → 15), ($v_{Kayseri,Corum}$ → 40), ($v_{Malatya,Sivas}$ → 25) }, I_S = {Kayseri}, and F_S = {Ordu}

Note that in Figure 5.1, capability cost values of the system are assumed to be (not the physical distance but) transportation costs between locations. Therefore, in our example cost of the route from Kayseri to Ordu (or process (S)) is ∞. In other words, the user has no idea about the transportation cost of any road reaching to city Ordu.

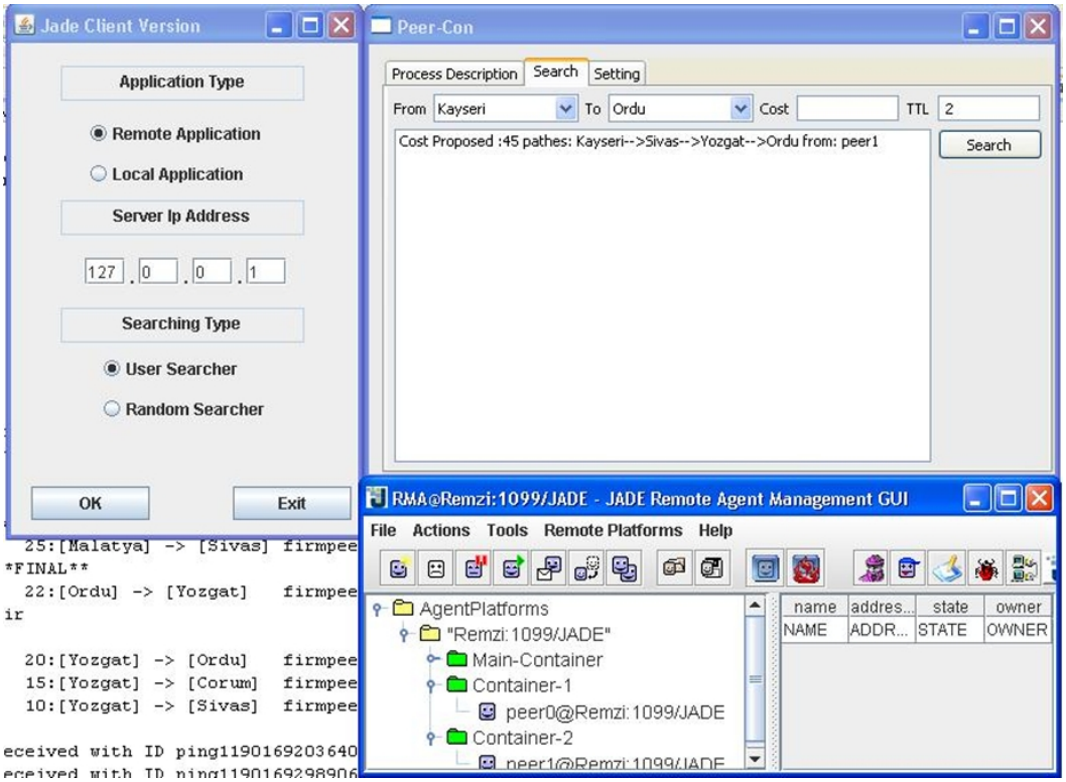


Figure 5.2: Peer-Con Process Description tab.

Customized Peer-Con application user interface holds three tabs. Process description tab is used for entering user local map information. Figure 5.2 shows the entered values for the example process.

Search tab is used to define source and destination locations, cost upper bound and TTL values. For example, in Figure 5.5 user enters a query asking for a route starting from Kayseri to Ordu whose cost is required to be at most 60. Also, the search radius is required to be 2 TTL. Assume there is another peer having process Q defined in Figure 5.3 in the same network in the range of 2 TTL.

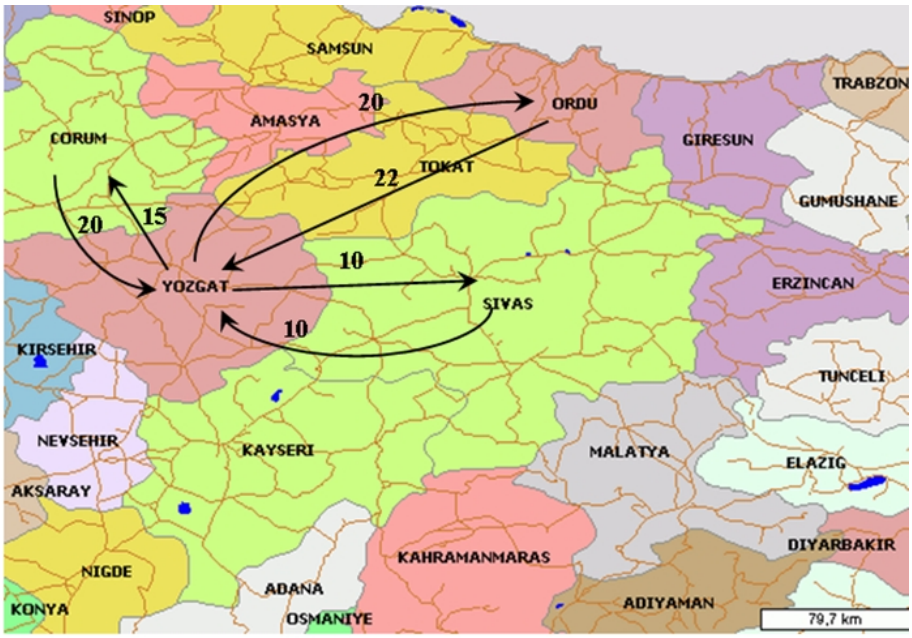


Figure 5.3: Another example of hypothetical transportation costs between cities .

The merge of two process shown in Figure 5.4 constitute a match according to our definition. The result(s) returned by the system are also displayed on the same tab. The query result "Cost proposed: 45 path: Kayseri → Sivas → Yozgat → Ordu from: peer1" in the figure means that there is path from Kayseri to Sivas to Yozgat to Ordu offered by peer1 with cost 45.

The third tab in Figure 5.6 is for system parameter settings. Parameters include peer neighbor upper limit (e), connectivity update ratio (r), timeout value (v) (in seconds) for query processing and ping frequency (in 1/seconds).

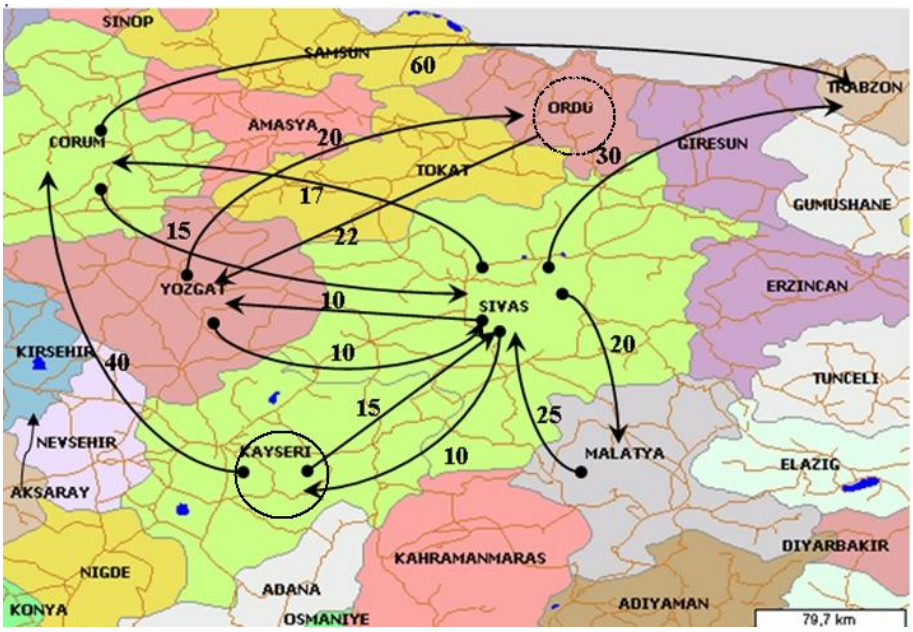


Figure 5.4: Merge of two processes.

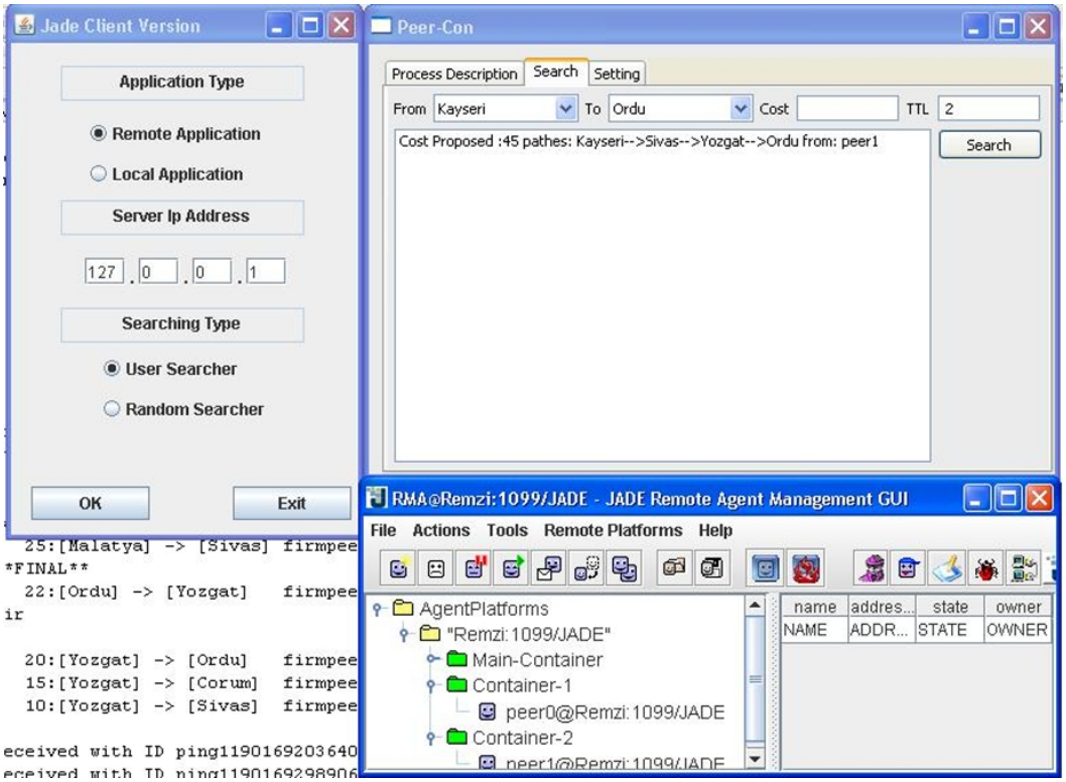


Figure 5.5: Peer-Con search and query result tab.

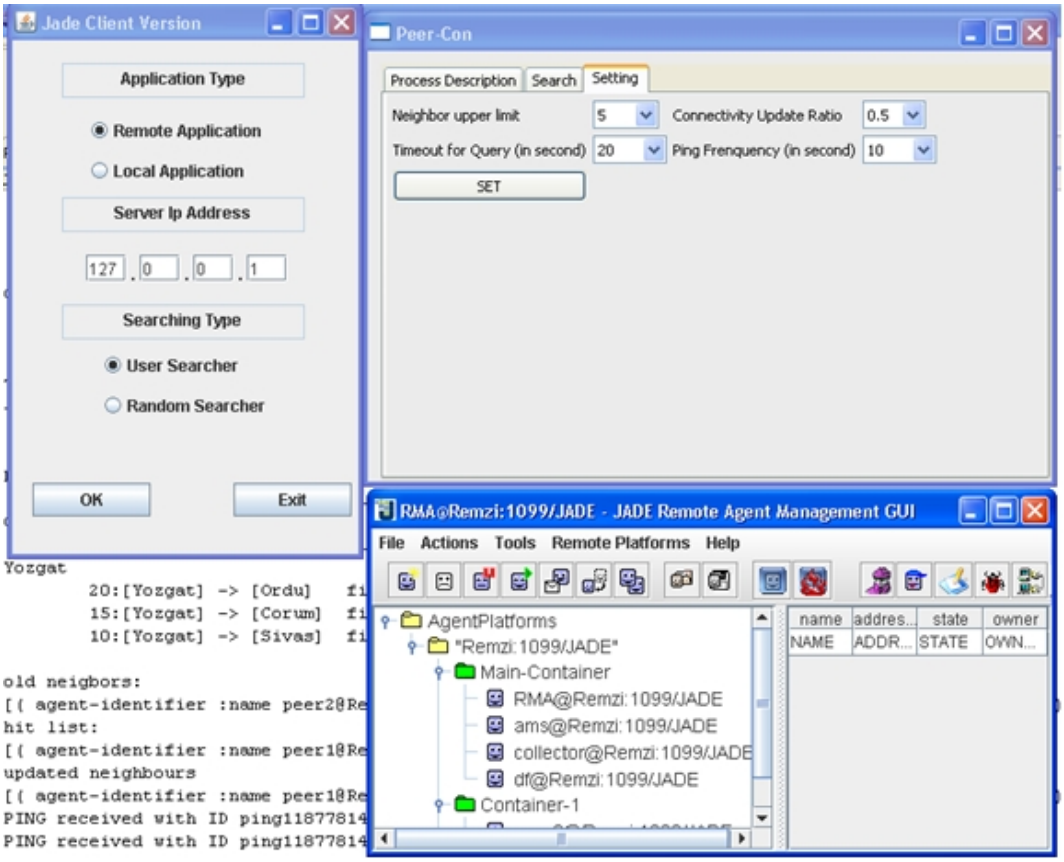


Figure 5.6: Peer-Con system parameters settings tab.

5.2 Digital Signal Processing Module Product Planning Application

The usage of DSPM in electronics industry is quite common since it can be an important component of some higher level products. As a consequence of this, most of the DSPM product planning stages are almost standardized for most companies. There are typical, possibly concurrent stages in DSPM product planning each showing some form of subacquisition for the whole product. They include design, production or outsourcing of submodules for example Printed Circuit Board (PCB) composed of a mainboard and electrical components; DSP card having a DSP software and PCB; and mechanical components integrated with the DSP card. While the above stages are common for most organizations, they may have different production process options depending on their time-varying capabilities. Two example hypothetical process descriptions for our eight-stage DSPM product planning problem are depicted in Figure 5.7.

The descriptions P and Q are assumed to be owned by two different organizations say D_1 and D_2 having standardized DSPM product planning. In the figure, the dashed nodes represent initial states of the process (i.e. P_I=v_1) while the bold ones are for final state descriptions (i.e. P_F=v_8). The stage transition costs are represented by labeled arcs I_p, O_p and C_p showing In-house, Outsourced and Commercial-Off-The-Shelf(COTS) realizations of the current stages. The minimum cost of the process P to organization D_1 is clearly 16+9=25. Similarly, it is 4+15+8=27 to D_2. On the other hand, cooperative execution of the process Merge(P, Q) can be as low as 4+3+1+5+5=18, as long as mainboard design and production is done by D_2 in-house, PCB realization and its software integration (leading to DSP Card availability) are partly done in-house and partly outsourced by D_1 and final realization of the DSP module is outsourced under the control of D_2. Note that a cooperation may not always be formed among peer organizations especially when they have no common view of stages described in related ontologies. Finally, in our formal process description, possible node costs are not taken into consideration. However, this can easily be added to the process model based on domain semantics.

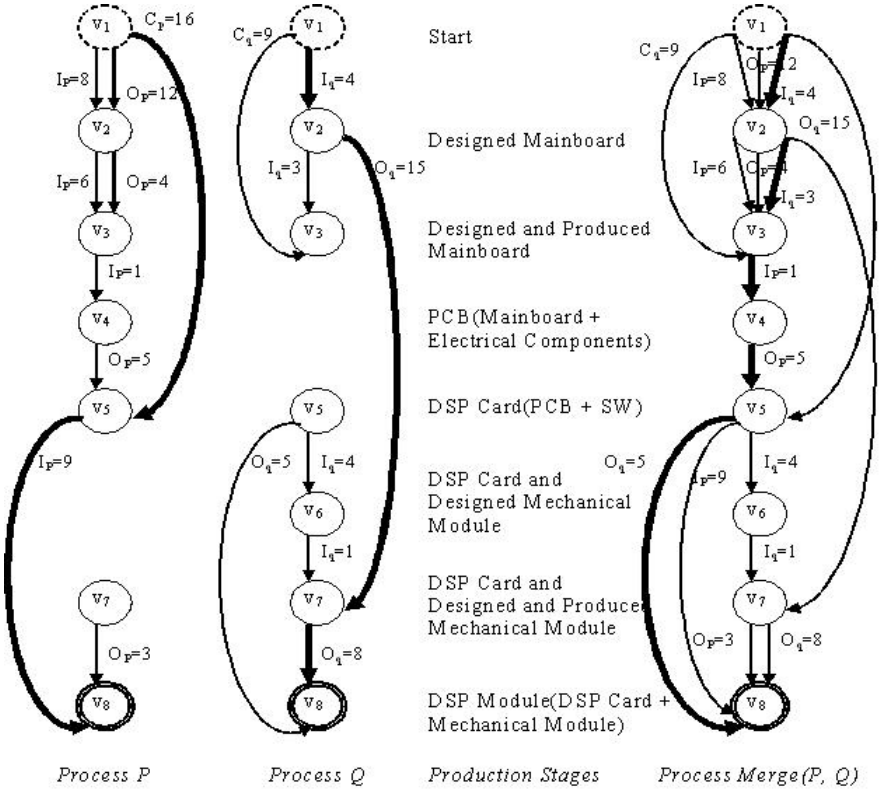


Figure 5.7: Example DSPM processes and their merge

CHAPTER 6

CONCLUSION

In this work, we described and implemented a process contracting system, what we call Peer-Con, based on P2P agent architecture. For this purpose, we developed contract-net like protocol supporting cooperative and decentralized nature of the system. Because of its underlying P2P architecture, Peer-Con facilitates local decision-making and information handling. The system satisfies three basic functional requirements: i) Cost aware flexible representation of capabilities ii) Description of an operator to decide on whether given capabilities turnout to an agreement or not iii) Self organization of peer connectivity for better contracting performance. Furthermore, it has the following general characteristics:

- There is no overlay level global control over the system.
- Interactions occur in asynchronous way via peer process broadcasting. Peer communication is restricted to the local neighborhood whose size is defined by Time-To-Live (TTL) value in *hop* unit.
- Data needed for problem solving are acquired through peer communications and there is no central shared repository of data. Peers become more informed about the contracting environment as they communicate with each other.
- Process description exchange occurs in cooperative way. There is no bargaining and/or negotiation for process information exchange.

Because of its generic process representation and contracting ability, Peer-Con can be customized to different problem domains. As an example, we showed that Peer-Con infrastructure can effectively be used in product planning for digital signal processing module.

6.1 Future Directions

In future, Peer-Con could be extended to support competitive and negotiation-based peer interaction features. For this purpose, we will need to develop a negotiation protocol that can facilitate possible encounters of competitive intelligent peers.

The process of searching for contracts do not have any computational efficiency since each peer sends messages to all its neighbors, so everything grows exponentially. We think some intelligence could be provided in the algorithms so that the search is much more efficient e.g. heuristics depending on past problems, in a way similar to the RTA* algorithm.

One important issue is the privacy of business processes. Our representation for the business process allows contracting among multiple parties however the privacy of the business process description is not considered in the study. Although, a privacy mechanism can be provided by the public-private key mechanism, queried processes are still shared among the peers in the network by the given TTL range. This issue still remains to be challenging and open.

The common ontology for the states of the different process is needed because of the assumption of that the companies run business in the same domain. Whereas, heterogeneous agents from different domains can reside in the environment and contracting and negotiation between them should be possible. To provide such contracting among heterogeneous agents, a state of the ontology could be mapped to a possible identical state of different ontology by assigning semantics or using the ontology mapping techniques.

REFERENCES

[1] M. Foca A. Ionita and M. Ienculecu-Popovici. Gis in business processes. www.gisdevelopment.net, June 2009.

[2] G. Agha and C. Hewitt. Concurrent programming using actors. *Computer Systems Series*, pages 37–53, 1987.

[3] S. Ali, B. Soh, and T. Torabi. A novel approach toward integration of rules into business processes using an agent-oriented framework. *IEEE Transactions on Industrial Informatics*, 2(3):145–154, 2006.

[4] J. H. Andrews. Definitions of the word 'map'. http://www.usm.maine.edu/ maps/essays/andrews.htm, June 2009.

[5] 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. Submitted to the First International Semantic Web Conference (ISWC01), 2002.

[6] I. Ariba, I. Corporation, and M. Corporation. Universal description, discovery and integration. http://www.uddi.org/, June 2009.

[7] X. Bai, S. Liu, P. Zhang, and R. Kantola. Icn: Interest-based clustering network. In *International Conference on P2P Computing*, August 2004.

[8] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. Jade: A white paper. *EXP in search of innovation*, 3:6–19, 2003.

[9] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[10] B. Çakır and H. Kılıç. An investigation about process matchmaking performances of unstructured and decentralized digital environment topologies. In *IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST-2007)*, pages 81–87, 2007.

[11] R. Çelebi, H. Ellezer, C. Baylam, I.Cereci, and H. Kılıç. Process matchmaking on a p2p environment. In *IEEE/WIC/ACM International Workshop on P2P Computing and Autonomous Agents (IEEE/WIC/ACM P2PAA 2006)*, pages 463–466, December 2006.

[12] R. Çelebi and H. Kılıç. Peer-con: A process contracting overlay. In *IEEE International Symposium on Industrial Electronics(IEEE INDEL 2008)*, pages 1808–1813, 2008.

[13] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1. W3c note, World Wide Web Consortium, March 2001.

[14] Limewire Community. Gnutella 0.4 protocol specification. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf, June 2009.

[15] RosettaNet Community. Rossettanet home. http://www.rosettanet.org, June 2009.

[16] R. Davis and R.G. Smith. Negotiation as a metaphor for distributed problem solving. *Artiicial Intelligence*, 20:63–109, 1 1983.

[17] I. M. Delamer and J. L. M. Lastra. A peer-to-peer discovery protocol for semantic web services in industrial embedded controllers. In *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, pages 2661–2667, November 2005.

[18] D. Drinjakovic and U. Epple. Search methods in p2p-networks of process control systems. In *2nd IEEE International Conference on Industrial Informatics (INDIN'04)*, pages 101–107, 2004.

[19] J. Eberspacher, R. Schollmeier, S. Zöls, and G. Kunzmann. Structured p2p networks in mobile and fixed environments. In *Proceedings of HET-NETs' 04 International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks*, July 2004.

[20] E.Ogston and S.Vassiliadis. A peer-to-peer agent auction. In *First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 151-159, 2002.

[21] D. Roller F. Leymann. Business processes in a web services world. http://www.ibm.com/developerworks/library/wsbpelwp/, June 2009.

[22] G.H.L Fletcher, H. A. Sheth, and K. Börner. Unstructured peer-to-peer networks: Topological properties and search performance. In *3rd Int. Workshop on Agents and Peer-to-Peer Computing (AP2PC), at the 3rd Int. Joint Conf. on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 14–27, 2004.

[23] M.R. Garey and D.S. Johnson. *Computers and Intractability: A guide to the theory of NP-Completeness*. W.H. Freeman Co., 1979.

[24] M. L. Gill. Combining mas and p2p systems: The agent trees multi-agent system (atmas). P.hD. thesis in Computing Science, University of Stirling, 2005.

[25] E. Grandgirard, C. Gertosio, and J. M. Seigneur. Trust engines to optimize semi-automated industrial production planning. In *16th IEEE International Symposium on Industrial Electronics (2007)*, pages 1814–1819, 2007.

[26] Barbara Hayes-Roth. A blackboard architecture for control. *Artif. Intell.*, 26(3):251–321, 1985.

[27] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2nd edition, November 2000.

[28] T. Klingberg and R. Manfredi. Gnutella Protocol Development. http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html, June 2009.

[29] Harita Genel Komutanlığı. Road map of eastern part of turkey. http://www.hgk.mil.tr, March 2008.

[30] Kraak and Menno-Jan. Why maps matter in giscience. *Cartographic Journal, The*, 43(1):82–89, March 2006.

[31] P. Maheshwari, S. S. Kanhere, and N. Parameswaran. Service-oriented middleware for peer-to-peer computing. In *2005 3rd IEEE Intemational Conference on Industrial Informatics (INDIN05)*, pages 98–103, 2005.

[32] S. Marmar. Gis in logistics and vehicle routing applications. In *Proceedings of National Workshop on Corporate GIS*, 1999.

[33] P. Massuthe, W. Reisig, and K. Schmidt. An operating guideline approach to the soa. *Annals of Mathematics, Computing Teleinformatics*, 1(3):35–43, 2005.

[34] S. A. Mcilraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16:46–53, 2001.

[35] Mick O'Leary. Mapquest and maps on us: top web map services. *Online*, 21(5):56–58, 1997.

[36] OASIS Open. ebxml - enabling a global electronic market. http://www.ebXML.org/, June 2009.

[37] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to peer architecture and applications. In *In Proceedings of the First International Conference on Peer-to-Peer Computing (P2P 01)*, 2001.

[38] O. Shehory. Robustness challenges in peer-to-peer agent systems. In *Agents and Peer-to-Peer Computing, Second Intl.Workshop, AP2PC 2003*, pages 13–22. Springer-Verlag LNAI 2872, 2003.

[39] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.*, 29(12):1104–1113, 1980.

[40] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *Proceedings of INFOCOM*, 2003.

[41] K. Sycara, J. Giampapa, B. Langley, and M. Paolucci. The retsina mas, a case study. In *in SELMAS*, pages 232–250. Springer-Verlag, 2003.

[42] Katia Sycara, Seth Widoff, Matthias Klusch, and Jianguo Lu. LARKS: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5:173–203, 2002.

[43] Ian J. Taylor and Andrew Harrison. *From P2P and Grids to Services on the Web: Evolving Distributed Communities*. Springer Publishing Company, Incorporated, 2009.

[44] M. Therani. Ontology development for designing and managing dynamic business process networks. *IEEE Transactions on Industrial Informatics*, 3(2):173–185, 2007.

[45] V. Tosic, K. Patel, and B. Pagurek. Wsol - web service offerings language. In *CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*, pages 57–67, London, UK, 2002. Springer-Verlag.

[46] W.M.P. van der Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Analysis - Modelling - Simulation*, pages 335–367, 1999.

[47] Wikipedia. Web mapping. http://en.wikipedia.org/wiki/Web_mapping, June 2009.

[48] Steven Willmott, Matteo Somacher, Ion Constantinescu, Jonathan Dale, Stefan Poslad, David Bonnefoy, Jerome Picault, and Juan Jim Tan. The agentcities network architecture. In *in Proceedings of the first International Workshop on Challenges in Open Agent Systems*, 2002.

[49] A. Wombacher, P. Fankhauser, B. Mahleko, and E. Neuhold. Matchmaking for business processes based on choreographies. *International Journal of Web Services Research*, pages 14–32, 2004.

[50] A. Wombacher and A. Martens. Soundness and Equivalence of Petri Nets and Annotated Finite State Automate: A Comparison in the SOA Context. In *IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, page 6. IEEE, 2007.