Process matchmaking on a p2p environment

Process Matchmaking on a P2P Environment

Conference Paper · December 2006

DOI: 10.1109/WI-IATW.2006.106 · Source: DBLP

5 authors, including:

Ibrahim Cereci
Atilim University
8 PUBLICATIONS **53** CITATIONS
SEE PROFILE

Hürevren Kılıç
Gediz University
27 PUBLICATIONS **138** CITATIONS
SEE PROFILE

Process Matchmaking on a P2P Environment

Remzi Çelebi, Hüseyin Ellezer, Cem Baylam, İbrahim Cereci and Hürevren Kılıç
Atılım University Computer Engineering Dept.
Incek, Golbasi, Ankara, TURKEY
e-mail: {rcelebi, hellezer, cbaylam, icereci, hurevren}@atilim.edu.tr

Abstract

A process matchmaking environment based on P2P architecture and Gnutella protocol is established. Java Agent Development Framework (JADE) is used as *middleware. 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 environment can easily be customized to a specific application domain by simple user-interface modifications and through the development of related state ontologies.*

1. Introduction

Peer-to-Peer (P2P) systems have potential to enhance internet-based trading among organizations or individuals. Decentralized nature of internet forces the P2P systems to be of choice not only for content sharing but also process level matchmaking. Local publication of business processes (or individual capabilities) and development of an automated matching mechanism for them may result in cheap contracting and automated trading among such interacting individuals or societies. Such mechanism can be achieved with or without a facilitator [1][2]. There are three basic questions related to the establishment of such business process matchmaking environments: (1) How to represent business goals and capabilities in the form of process description (2) How can we describe the match operation among the described processes? (3) What can be an efficient architecture and protocol that can facilitate such interactions?

Related to the first question, information publication in a raw string format followed by string matching while not considering any state information like UDDI [3] and WSDL[4] based solutions do, are not sufficient to represent process level dynamics. An alternative solution is to develop a representation in the form of Deterministic Finite Automaton (DFA) considering the required state information. For example, in WSCL[5] proposal finite state automata over the alphabet of message types is used to model input output sequences. In [6], annotated DFA (a-DFA) has been used for process description. About the second question; in [6], the match operation among two processes is defined by the existence of their language intersection. In both WSCL and a-DFA proposals, processes are modeled as interacting automata couple changing each other's state through message passing. In the second proposal, the match operation among processes is associated with sharing of common message sequence between processes. If there is no such common message sequence, the processes are said to be incompatible and no match occurs. About the third question; the WSCL is not related with the architectural issues but representation. The proposed architecture in [6] on the other hand, is a centralized client-server approach realized through a matchmaking engine [7]. In fact, to the best of our knowledge there is no business process matchmaking system implementation based on P2P protocols and architecture.

In this paper, we propose an alternative business process representation based on "one-input transition system" model described in [8]. Different from other representation proposals, in our approach, the process descriptions can be incomplete i.e. some states are allowed to be unreachable in given process description. We describe the match operation as a state-level merge of two processes followed by a reachability test for the goal states. In our approach, the reachability of goal states means the existence of match among processes. Our second contribution is the implementation of business process matchmaking environment using well-known P2P protocol, Gnutella 0.4. The P2P protocol implementation is realized on Java Agent Development Framework (JADE) [9]. In the implementation, the JADE agents behave like peers communicating through Gnutella 0.4 protocol using different state describing ontologies.

In section 2, we give formal definitions for the process representation and an algorithm describing the match operation. In section 3, the architecture and details of developed P2P process matchmaking system is introduced. The last section is the conclusion.

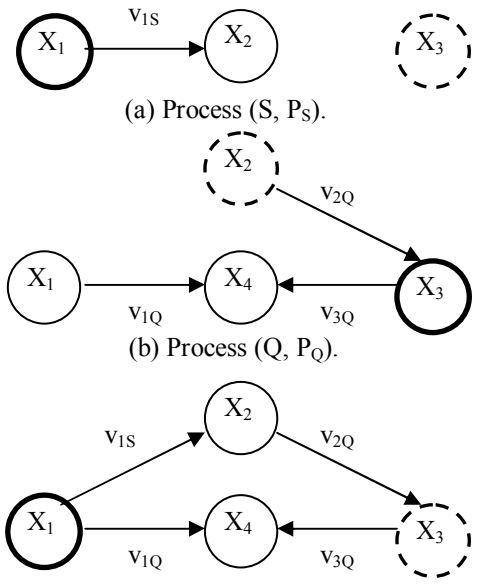
2. Process representation and match

Definition 1: Let Z be a finite set of states. *Process* is a tuple (S, P_S) such that S is a one-input transition system $S=(X, V, \delta, I)$ where $X \subseteq Z$ is a finite set of states and V is a finite set representing peer's capabilities. $\delta: X \times V \rightarrow X$ is the state transition function of the process. $I \in X$ is the initial (or starting) state of the process. $P_S \subseteq X$ is the peer's end (or goal) state set.

Due to the introduced input, one-input transition system is an open system. In its graph-like representation,

Proceedings of the 2006 IEEE/WIC/ACM International Conference on
Web Intelligence and Intelligent Agent Technology (WI-IAT 2006 Workshops)(WI-IATW'06)
0-7695-2749-3/06 $20.00 © 2006 IEEE

see Figure-1, circle nodes define process states ($x_i \in X$). The tick circle shows the initial state and dashed circles show the end states. Goal of the peer executing its process is to reach one of its goal states from its starting state. In Figure-1, $Z = \{x_1, x_2, x_3, x_4\}$, $S = (X, V, \delta, I)$ where $S_X = \{x_1, x_2, x_3\}$, $S_V = \{v_{1S}\}$, $S_\delta = \{((x_1, v_{1S}) \rightarrow x_2)\}$, $S_I = \{x_1\}$ and $P_S = \{x_3\}$. Similarly, $Q = (X, V, \delta, I)$ where $Q_X = Z$, $Q_V = \{v_{1Q}, v_{2Q}, v_{3Q}\}$, $Q_\delta = \{((x_1, v_{1Q}) \rightarrow x_4), ((x_2, v_{2Q}) \rightarrow x_3), ((x_3, v_{3Q}) \rightarrow x_4)\}$, $Q_I = \{x_3\}$ and $P_Q = \{x_2\}$. For both system S and Q their individual induced behaviors do not satisfy the property of "reaching to" their goal-states P_S and P_Q, respectively. In other words, there exists no successfully executing peer either for process (S, P_S) or process (Q, P_Q).



(a) Process (S, P_S).

(b) Process (Q, P_Q).

(c) Process $(T=Merge(S, Q), P_S)$.

Figure 1. Example process descriptions and merge operation.

Definition 2: Given two processes (S, P_S) and (Q, P_Q). S and Q are called c*apability-disjoint* iff $S_V \cap Q_V = \varnothing$.

Definition 3: Processes (S, P_S) and (Q, P_Q) are called *goal-equivalent* iff $P_S = P_Q$.

Systems S and Q in Figure-1 are *capability-disjoint*. Processes (S, P_S) and (Q, P_Q) are not *goal-equivalent*. In general, a one-input transition system is not *capability-disjoint* with itself however any process is *goal-equivalent* to itself.

Definition 4: Given two capability-disjoint one-input transition systems S and Q, *merge*(S, Q) operation returns a one-input transition system T such that $T_X = S_X \cup Q_X$, $T_V = S_V \cup Q_V$, $T_\delta = S_\delta \cup Q_\delta$ and $T_I = S_I$.

The operation is not commutative but associative. In fact, the merge operation implies a graph union whose nodes (i.e. states) take values from the same domain. In the implementation of merge operation, different state values from different domains are handled by different pre-constructed state-ontologies. Figure-1(c) shows system T obtained by the merge of systems S and Q.

Definition 5: (*Behavior Induced by Input*).
Given a system $S = (X, V, \delta, I)$ and an input sequence $\psi \in V^*$, the behavior of S starting from I in the presence of ψ is a sequence

$$\xi(\psi) = \xi[0], \xi[1], \ldots \in X^*$$

such that $\xi[0] = I$ and for every i, $\xi[i+1] = \delta(\xi[i], \psi[i])$.
In the automaton of Figure-1(c), an input starting with v_{1S}, v_{2Q} generates a behavior starting with X_1, X_2, X_3 a fact that can be denoted as:

$$X_1 \xrightarrow{v_{1S}} X_2 \xrightarrow{v_{2Q}} X_3$$

Definition 6: (Reachability for Open Systems).
Given a system $S = (X, V, \delta, I)$ and a set $P \in X$, is there some input sequence $\psi \in V^*$ such that the behavior $\xi(\psi)$ reaches P?

Assume that $\delta(x)$ is the set of all *immediate successors* of x, i.e. $\delta(x) = \{x': \exists v\ \delta(x, v) = x'\}$ and we can extend this notation to sets of states F by letting $\delta(F) = \{(\delta(x) : x \in F\}$ The following algorithm computes all reachable states of a one-input transition system:

Algorithm *Reachables*
Input: S = (X, V, δ, I), a one-input transition system
Output: F, set of all states reachable from I

```
F^0 := I
repeat
    F^{k+1} := F^k ∪ δ(F^k)
until F^{k+1} = F^k
F* := F^k
return F*
```

The algorithm is a simple polynomial-time graph search algorithm searching breadth-first manner in which every F^k consists of the states reachable after at most k transitions.

Note that, for the systems of Figure-1, *Reachables*(S) returns $\{X_1, X_2\}$ and *Reachables*(Q) returns $\{X_3, X_4\}$. Similarly, *Reachables*(T) returns $\{X_1, X_2, X_3, X_4\}$. Having defined merge operation and reachables algorithm we can define our match algorithm executed by every peer. In the following algorithm, it is assumed that the process owned and to be executed by peer *p* is (S, P_S) and process description from another peer *r* is (Q, P_Q). By applying the algorithm a peer may obtain four different match results: 0 – no match; 1 – only the process of peer *p* matches with *r*'s; 2 – only the process of peer *r* matches with *p*'s; 3 – mutual match, both *p* and *r*'s processes match with each other. The match result of the example in Figure-1 is $match((S, P_S), (Q, P_Q)) = 1$.

Proceedings of the 2006 IEEE/WIC/ACM International Conference on
Web Intelligence and Intelligent Agent Technology (WI-IAT 2006 Workshops)(WI-IATW'06)
0-7695-2749-3/06 $20.00 © 2006 IEEE

Algorithm *Match*
Input: Two processes (S, P_S) and (Q, P_Q) in given order whose systems S and Q are *capability-disjoint*.
Output: match-result.

T = *merge*(S, Q);
T' = *merge*(Q, S);
switch match-result
0 : (($P_S \cap$ *reachables*(T)) = ∅) **and**
(($P_Q \cap$ *reachables*(T')) = ∅);
1 : (($P_S \cap$ *reachables*(T)) ≠ ∅) **and**
(($P_Q \cap$ *reachables*(T')) = ∅);
2 : (($P_S \cap$ *reachables*(T)) = ∅) **and**
(($P_Q \cap$ *reachables*(T')) ≠ ∅);
3 : (($P_S \cap$ *reachables*(T)) ≠ ∅) **and**
(($P_Q \cap$ *reachables*(T')) ≠ ∅);
end;
return match-result;

3. Implemented P2P business process matchmaking system

The P2P system is implemented on JADE platform. JADE provides a middle-ware for the development and run-time execution of peer-to-peer applications. The main reason behind using JADE was to use its peer-to-peer facilitating architecture and its rich message-handling capabilities. It enables an interoperable platform for both in wired and wireless environments. Multiagent systems are inherently P2P systems and an agent is a peer in P2P agent systems [10]. From the perspective of multiagent systems the implemented P2P system is not competing but cooperating agents. As it can be seen from Figure-2, the P2P network is an overlay on top of JADE middleware.

The interaction protocol among peers is assumed to be Gnutella 0.4 which supports the unstructured topology of P2P setups. The implemented Gnutella messages namely *ping*, *pong*, *query* and *query-hit* are embedded into the basic syntax of standard FIPA Agent Control Language (ACL) supported by JADE. In ACL syntax any message starts with the performative showing the intended actions to be taken by its receiver. Rest of the message may either contain built-in attributes like sender, receiver, in-reply-to or user-defined attributes.

The process descriptions are entered via a generic user interface as seen in Figure-3. It can easily be customized to a specific user domain. In our implementation, peers are made neighbor-aware through a local look-up table holding neighbors' addresses updated via basic *ping* and *pong* messages of Gnutella protocol. The query type message of the protocol is used to pass business process descriptions to other peers in the network which may have a match potential. Match algorithm is runs on each peer and according to its result the necessary Gnutella 0.4 action like Query or Query-hit is taken by the peer. The following examples show the Gnutella message implementations using the ACL syntax.
Ping implementation:

```
(QUERY-REF
:sender peer2
:receiver peer5
:reply-with ping1154566259531
:X-ttl 5
:X-originator "peer1" )
```

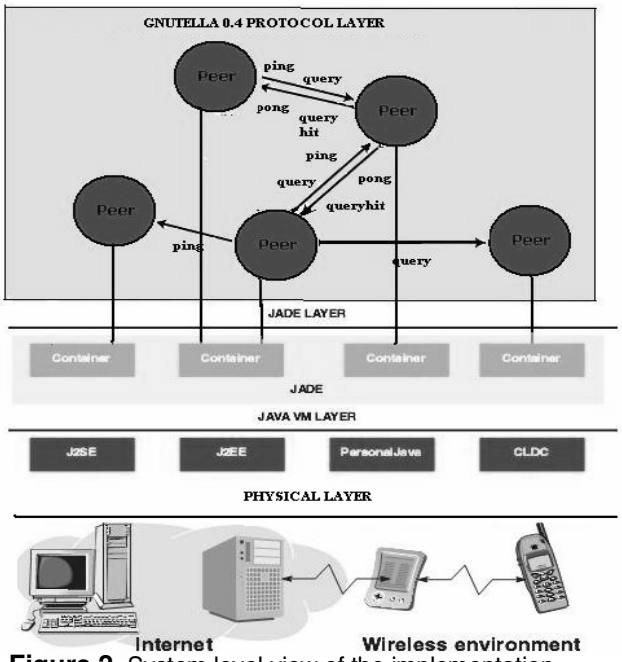


Figure 2. System level view of the implementation

The 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 implementation:

```
(INFORM
:sender  peer5
:receiver peer2
:in-reply-to  ping1154566259531
:X-ttl 5
:X-originator "peer5"
:X-receiver "peer1")
```

The pong message is the answer for the ping message. It is used to inform the message originator about liveness of the *ping*ed 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.

Query implementation:

```
(CFP
:sender peer1
:receiver peer5
:content    "((Reachable    owner:peer1
process:a1))"
:language fipa-sl
:ontology Task-description-ontology
:X-ttl 5
:X-orginator "peer1" )
```

The query message, in our context, is used to initiate the process match operation. The sender of the Call For Proposal (CFP) message looks for a match 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 matches between owner's process a1 and receiver peers' internal processes. The ontology attribute is used to decide on the related state-domain of the process.

Query-Hit implementation:

```
(PROPOSE
:sender peer5
:receiver peer1
:content  "((Propose  :proposer  peer5
:matchresult 2 :process a3))"
:in-reply-to  query1154566257093
:language  fipa-sl
:ontology  Task-description-ontology
:X-ttl 5
:X-orginator "peer5"
:X-receiver "peer1")
```

The query-hit message is the answer for the query message. The content part of the Query-Hit message holds the proposer's (or owner's) id, type of the match and the proposer's corresponding process description, if any match occurs. The ontology attribute shows the process' state domain. In typical JADE installation, there is a main container holding a specialized agent called Directory Facilitator (DF). In our implementation, the DF agent is directly used as the BootStrapServer of P2P setup. The role of the BootStrapServer provides an address list of peers residing in the network to those peers that want to be part of it.

4. Conclusion

A process representation enabling incomplete descriptions and an algorithm facilitating matchmaking operation on them are introduced. Following this, a decentralized P2P matchmaking environment is established. The environment can easily be customized to a specific application domain by simple user-interface modifications and through the development of related state ontologies.

In future, we can enhance process representation through the assignment of utility values to the goals and costs to the capabilites that may facilitate the consideration of possible negotiation mechanisms among peers.

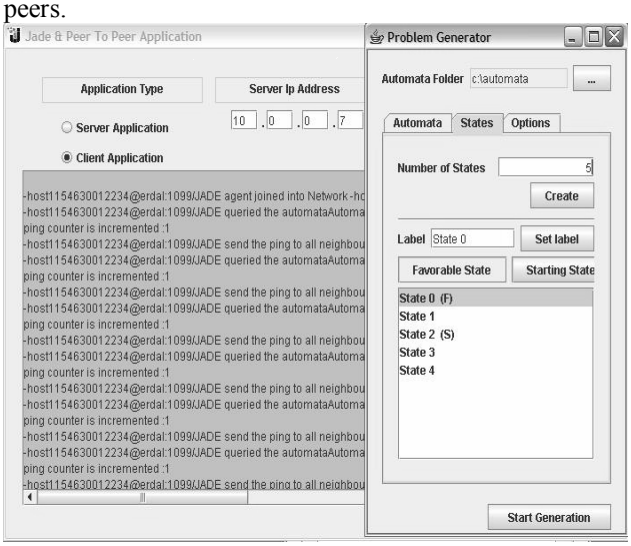


Figure 3. User interface for process description entry.

5. References

[1] K. Sycara, J. Lu, M. Klusch, and S. Widoff, "Matchmaking among heterogeneous agents on the internet", *in Proceedings of the 1999 AAAI Spring Symposium on Intelligent Agents in Cyberspace*, Stanford University, USA 22-24 March 1999.

[2] I. Constantinescu, and B. Faltings "Efficient Matchmaking and Directory Services", *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, October 2003, pp. 75-81.

[3] I. Ariba, I. Corporation, and M. Corporation, "Universal description, discovery and integration", September 2000, http://www.uddi.org/.

[4] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web services description language (WSDL) 1.1", March 2001. http://www.w3.org/TR/wsdl.

[5] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma, and S. Williams. "Web services conversation language (WSCL) 1.0", March 2002. http://www.w3.org/TR/wscl10/.

[6] A. Wombacher, P. Fankhauser, B. Mahleko, and E. Neuhold, "Matchmaking for Business Processes Based on Choreographies", *International Journal of Web Services Research*, 1(4), Oct-Dec 2004, pp. 14-32.

[7] A. Wombacher, B. Mahleko, and E. Neuhold, "IPSI-PF: A business process matchmaking engine based on annotated finite state automata", *Inf. Syst. E-Business Management*, 3(2), 2005, pp. 127-150.

[8] O. Maler, "Control from Computer Science", *Annual Reviews in Control*, 26, 2002, pp. 175-187.

[9] F. Bellifemine, A. Poggi, G. Rimassa, "JADE: A White Paper", 3 September 2003.

[10] O. Shehory, "Robustness challenges in peer-to-peer agent systems", *Agents and Peer-to-Peer Computing, Second Intl.Workshop*, AP2PC 2003, LNAI 2872, pp. 13-22.

Proceedings of the 2006 IEEE/WIC/ACM International Conference on
Web Intelligence and Intelligent Agent Technology (WI-IAT 2006 Workshops)(WI-IATW'06)
0-7695-2749-3/06 $20.00 © 2006 IEEE