Process matchmaking on a p2p environment

Citation for published version (APA):

Document status and date:
Published: 01/01/2006

DOI:
10.1109/WI-IATW.2006.106

Document Version:
Publisher’s PDF, also known as Version of record

Document license:
Taverne

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain.
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.umlib.nl/taverne-license

Take down policy
If you believe that this document breaches copyright please contact us at:
repository@maastrichtuniversity.nl
 providing details and we will investigate your claim.
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

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. \(\delta: XXV \rightarrow X\) 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,
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 \( S_X = \{x_1, x_2, x_3\} \), \( S_V = \{v_1S\} \), \( S_I = \{((x_1, v_1S) \rightarrow x_2)\} \) and \( P_S = \{x_3\} \). Similarly, \( Q = (X, V, \delta, I) \) where \( Q_X = Z \), \( Q_V = \{v_1Q, v_2Q, v_3Q\} \), \( Q_I = \{((x_1, v_1Q) \rightarrow x_2), ((x_3, v_2Q) \rightarrow x_3), ((x_5, v_3Q) \rightarrow x_4)\} \), \( Q_1 = \{x_1\} \) and \( P_Q = \{x_2\} \). For both systems 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 \( (Q, P_Q) \).

**Definition 2:** Given two processes \( (S, P_S) \) and \( (Q, P_Q) \). \( S \) and \( Q \) are called capability-disjoint iff \( S_X \cap Q_X = \emptyset \).

**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 \( T_X = S_X \cup Q_X \), \( T_V = S_V \cup Q_V \), \( T_I = S_I \cup Q_I \), 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 \( \psi \) 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 \( P \subseteq X \), is there some input sequence \( \psi \in V^* \) such that the behavior \( \xi(\psi) \) reaches \( P \)?

Assume that \( \delta(\cdot) \) is the set of all immediate successors of \( x \), i.e. \( \delta(x) = \{x': \exists v \in V \cup \delta(x, v) = x'\} \) 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:** \( S = (X, V, \delta, I) \), a one-input transition system

**Output:** \( F \), set of all states reachable from \( I \)

- \( F_0 := I \)
- repeat \( F_0 := F_k \)
- until \( F_k = F \)
- \( F := F_k \)
- 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 reaches 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 \( \text{match}((S, P_S), (Q, P_Q)) = 0 \).
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.

\[
T = \text{merge}(S, Q);
\]
\[
T' = \text{merge}(Q, S);
\]

switch match-result

\[0 : ((P_S \cap \text{reachables}(T)) = \emptyset) \quad \text{and} \quad ((P_Q \cap \text{reachables}(T')) = \emptyset);
\]
\[1 : ((P_S \cap \text{reachables}(T)) \neq \emptyset) \quad \text{and} \quad ((P_Q \cap \text{reachables}(T')) = \emptyset);
\]
\[2 : ((P_S \cap \text{reachables}(T)) = \emptyset) \quad \text{and} \quad ((P_Q \cap \text{reachables}(T')) \neq \emptyset);
\]
\[3 : ((P_S \cap \text{reachables}(T)) \neq \emptyset) \quad \text{and} \quad ((P_Q \cap \text{reachables}(T')) \neq \emptyset);
\]
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:

\[
\text{(QUERY-REF :sender peer2 :receiver peer5 :reply-with ping1154566259531 :X-ttl 5 :X-originator "peer1")}
\]

Pong implementation:

\[
\text{(INFORM :sender peer5 :receiver peer2 :in-reply-to ping1154566259531 :X-ttl 5 :X-originator "peer5" :X-receiver "peer1")}
\]

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.

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

**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 capabilities that may facilitate the consideration of possible negotiation mechanisms among peers.

**5. References**


