UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

LEVERAGING WEB-SERVICES AND PEER-TO-PEER NETWORKS

Mike P. Papazoglou, Jian Yang and Bend J. Kramer

October 2002

Technical Report # DIT-02-0088

Leveraging Web-Services and Peer-to-Peer Networks

Mike P. Papazoglou[1], Jian Yang[1], Bend J. Krämer [2]
1. INFOLAB — Tilburg University, PO Box 90153,
NL-5000 LE Tilburg, The Netherlands
email: {mikep,jian}@kub.nl,

2. FernUniversität Hagen,
D-58084 Hagen, Germany
email: bernd.kraemer@fernuni-hagen.de

July 8, 2002

Abstract

Two of the most recent and popular topics in computing are service-oriented computing (exemplified by web-services) and peer-to-peer computing. Peer-oriented computing is an attempt to weave inter-connected machines into the fabric of the Internet. Web-services, on the other hand, are a more formal technological challenge, an attempt to apply a service-oriented computing model to web resources to provide a loosely coupled paradigm for distributed processing.

Despite the fact that these two concepts have some significant amount of overlap, e.g., each seeks to become a common means for publishing and discovery across networks, their current manifestations still remain quite diverse. In this paper we highlight key intersect points that enable possibilities for using these two technologies together. Moreover, we present an architectural approach and formal framework towards unifying them to provide essential functions required for automating e-business applications such as e-marketplaces and service exchanges.

Keywors: e-business, web-services, service-oriented computing, e-marketplaces, peer-to-peer computing, XML, WSDL, UDDI.

1 Introduction

E-business is shifting attention from component based to web-service based applications. The increasing use of e-business interactions has led to great interest in the deployment of web-services carrying a wide variety of XML encoded business data as part of the service functionality. A large number of enterprises is implementing a SOAP/WSDL/UDDI layer on top of existing applications or components and is assembling applications by consuming web-services. This means that instead of exposing proprietary APIs from enterprise applications, such as for instance SAP, one provides an open abstraction layer that facilitates the translation of requests between systems.

From a conceptual point of view, web-services are an example of a Service-Oriented Architecture (SOA). With this architecture the service requester, service provider and service registrar (broker) form a process triptych that provides publishing, binding and interaction semantics. These three entities work in tandem to provide a loosely coupled computing paradigm. Interactions between web-services occur as series of request-reply message sequences, in which various additional service elements may be added in a value chain. The manifestation of this paradigm is through widely accepted industry standards such as XML, SOAP, WSDL (Web-Services Definition Language) and UDDI (Universal Description, Discovery and Integration protocol). These standards offer the three parties (requester, provider and registrar) a common language to communicate. Interactions of web-services occur as SOAP calls carrying XML data content and the service definitions of the web-services are expressed using WSDL as the common (XML-based) standard. The UDDI standard is a directory service that enables web-service clients to locate candidate services and discover their details. It is the role of the directory to carry out queries and return service descriptors to the requesting application, which then binds them automatically to the service implementation.

The characterization of the web-service operation is the classic client/server model. The service provider (server) will register with the UDDI registry and the requester (client) will contact the registry to discover the server location so that it can interact with it. This is a straightforward approach to distributed computing that provides the advantage that clients are coupled to the servers only via a contract mechanism. Since this contract is fully described by using WSDL, developers can construct clients using the contract information. All providers must make their services available by publishing their contract and advertising their service. However, the use of a centralized directory can lead to performance bottlenecks if a large number of clients visit the directory. Adding more servers or implementing load-balancing strategies do not constitute practical solutions as they may prove to be costly and disruptive.

At its core, Peer-to-Peer (P2P) computing is the sharing of computer resources and services through direct communication between systems. Computers that traditionally acted as clients now incorporate server capabilities that enable them to share processing power, bandwidth, and storage. Peer-to-peer is often described as "collaborative networking" technology, where each node in the system, called a peer, may store data relevant to that peer and potentially useful to other peers in the network. Each functional unit in the network is behaviourally similar. When a peer decides that data hosted on another peer is useful, it visits directly this peer in order to obtain that data. The fundamental characteristic of a P2P network is that any machine in the network is logically capable of both providing and consuming information.

P2P networks such as Gnutella and Freenet [8, 5] display a unique quality that sets them apart from other peer networks: they are truly distributed. Unlike, Napster [12], which is a brokered peer network relying a centralized directory for indexing purposes, P2P networks like Gnutella and Freenet have no need for central index or server. Rather, they operate collectively across all nodes in the network. Searching this type of ”true” P2P network results in a request that searches across the network nodes until most or all nodes are covered. The key element in a P2P network is a complex search algorithm. The search algorithm does not require that the requester and provider have a priori knowledge of one another, but rather enforces a logical network topology by defining the concept of ”neighbouring” peers. The P2P network is usually fluctuating and dynamic with peer neighbour relationships breaking and reforming as the load or infrastructure stability changes.

When comparing P2P networks with web-services functionality, we observe that peer-to-peer systems also leverage a service-oriented architecture but have their own idiosyncrasies. Unlike web-services, the determination of who is a provider, a requester or a registrar (of a resource) is much looser. Typically, a peer is all three of the aforementioned roles. However, like web-services, peers must also publish a resource (allowing it to be found and accessed by other peers) with efficient precision for the other peers to be able to broadcast their needs and receive meaningful responses. Publication and discovery are paramount for both paradigms, however, the two approaches diverge on lookup services. Peers use decentralized discovery while web-services use larger centralized directories such as UDDI. Lastly, another similarity is that both web-services and P2P have heavy emphasis on distributed computing and on using XML as a means to describe information.

Fortunately, the standards and frameworks used to create web-services can also be utilized to develop P2P applications. This is because both sets of architectures fundamentally coordinate interactions between loosely coupled systems. Utilizing a common framework based on current web-services technologies would enable P2P developers with elementary building blocks for building applications. In fact, JXTA, the P2P framework initiated by Sun Microsystems [9], is making adjustments to its core platform to make peers interoperate with web-services using protocols like SOAP and WSDL. It is thus highly probable that in the future both peer services will in the future rely on WSDL and SOAP for service descriptions and invocations.

This paper examines key intersect points that enable web-services and P2P networks to work together and in, particular, looks at ways in which web-services discovery can benefit from P2P decentralization. Our contribution concentrates on an architectural approach and formal framework towards unifying web-services and P2P networks to provide essential functions required for automating e-business applications such as e-marketplaces and service exchanges.

2 Problems with web-service directories

One of the main reasons for enterprises engaging in electronic business is to open new markets and find new sources of supply more easily than with conventional means. To achieve this desired state, enterprises use a common service registry (UDDI) for identifying potential trading partners and for cataloguing their business functions and characteristics. UDDI presents a standard way for enterprises to build a registry to describe and identify e-business services, query other service providers and enterprises, and enable the registered enterprises to share business and technical information globally in a distributed manner. UDDI specification provides two main types of interfaces (APIs): one for describing services and registering service entries in the directory and one for enquiring about service entries and provider characteristics. This allows the services to be dynamically discovered and composed into more complex (value-added) services.

It is unreasonable to expect that there will be a relatively small number of global UDDI registries to provide discovery for all possible services in a similar fashion that search engines index terms and documents. Rather, relatively large numbers of specialised UDDI directories will emerge across the Internet. More specifically, it is expected that vertical sectors will have a variety of registries that serve their community as a whole. For example, some industry-based (or *vertical*) e-marketplaces, such as semiconductors, petro-chemicals, travel industry, aerospace, financial services, etc, will provide to their members a unified view of sets of UDDI-based products and services to enable them to transact business using diverse mechanisms, such as web-services. And this is already happening to a large extend. Applications and services can be published and hosted throughout the e-marketplace network and used on demand. The goal of web-services when used within the context of e-marketplaces is to enable business solutions by assembling and programming web-services offering business functionality on the Web. This allows companies to conduct electronic business, by invoking web-services, with all partners in a marketplace rather than with just the ones with whom they have collaborative business agreements. Service offers are described in such a way, e.g., WSDL over UDDI, that they allow automated discovery to take place and offer request matching on functional and non-functional service capabilities.

Currently, P2P networks focus more on the discovery of content rather than on common files (or documents) published with layer of meta-data over them. The content served by a UDDI repository is in essence some form of meta-data, i.e., WSDL, albeit low-level, describing web-services. However, where P2P networks serve their content in a highly distributed manner, UDDI provides a central discovery mechanism for service provisioning [10]. In P2P networks content is normally described and indexed locally to each peer and search queries are propagated across the network. In this model no central index is required to span the network. One of the major points of intersection between P2P and web-service technologies involves bringing the decentralization aspect of P2P networks to the central service discovery mechanisms of web-services provided by UDDI. It is not difficult to envision a P2P network architecture that promotes a logically decentralized arrangement of registered-service descriptions and that also provides web-service descriptions much in the same way that UDDI does. Using this form of decentralized approach, service providers can become peers in an e-marketplace network of peer services (or web-service providers).

3 A federated architecture for P2P web-services

Rather than requiring each peer to publish their own service descriptors locally or centrally (on the UDDI), we can employ a federation of UDDI-enabled, peer registries that operate in a decentralized fashion. Such federations may represent common interest groups of peers that band together to ensure that they provide added-value syndicated-services to their customers. A *peer (service)-syndication* seeks to promote in-demand services by offering sets of related services throughout the federation rather than on a single location, see Figure 1.

Two of the key concepts in a peer service-syndication are the notions of *advertisement* and *subscription*. Advertisements are simple XML documents that name, describe and publish the existence of peers that act as service providers, while subscriptions also name, describe and publish the service requirements of peers that act as service requesters within a service syndication. Discovery within a service-syndication becomes an issue of matching service subscriptions against service advertisements.

Service providers first publish their services on UDDI and then they may join a service-syndication. A peer-syndication is formed for specific areas of interest in an e-marketplace, such as e-travel, finances, marketing and so on. When joining the P2P web-service network, a peer first registers itself by advertising its services. Secondly, it may subscribe to services that it is interested in from other peers in the syndication. For each syndication a specific peer acts as *super-peer* by providing directory services to the peer-syndication, see Figure 1.

The super-peer acts as an event-notification server that receives and stores the advertisements and subscriptions of the entire peer-syndication. Event-notification is used for asynchronous coordination of distributed systems. With this scheme publisher peers notify subscribers of interesting events, i.e., the publication of services. The subscribers may in turn perform actions in response to these events. Periodically, each super-peer sends a summary update to its syndication to notify all potential subscribers about new advertisements (registrations) and deletions of registrations it receives. This component is similar to that found in a hybrid P2P architecture where indexing is centralized and file exchange is distributed [16]. The super-peer manages a select set of meta-

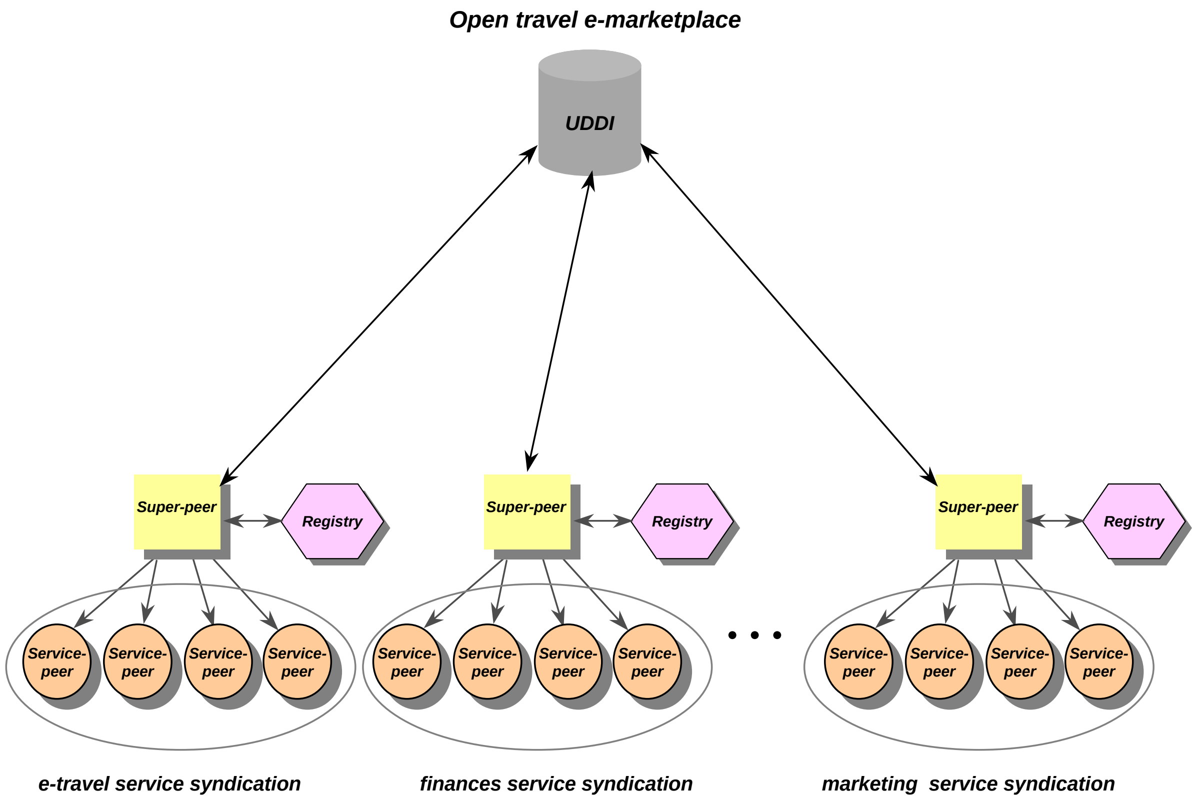

Open travel e-marketplace
UDDI
Super-peer
Registry
Super-peer
Registry
Super-peer
Registry
Service-peer
e-travel service syndication
finances service syndication
marketing service syndication

Figure 1: Conceptual architecture of the P2P service network.

operations for peers, such as joining/leaving the network, publishing service advertisements, and service subscriptions. The super-peer also performs subscription/publication matching, so that relevant notifications can be sent to interested subscribers (peers).

When a super-peer receives a new publish/subscribe event from a peer that wishes to join the network, it performs two kinds of matching operations. The first matching operation matches the new peer's advertisement against all subscriptions that are relevant to it. In this way a notification (regarding the newly advertised peer) is sent to all peers in the network that have subscribed to services relating to services offered by this new peer. The second matching operation matches all previous advertisements against the service subscriptions of this new peer. Whenever a match is detected, the super-peer forwards the peer publication information regarding the services offered by the matching peers and their addresses to all the peers that have subscribed to their advertised services. As a consequence, each peer contains high-level service descriptions as well as the addresses of the peers that have published information (services) that this peer has subscribed to. In this

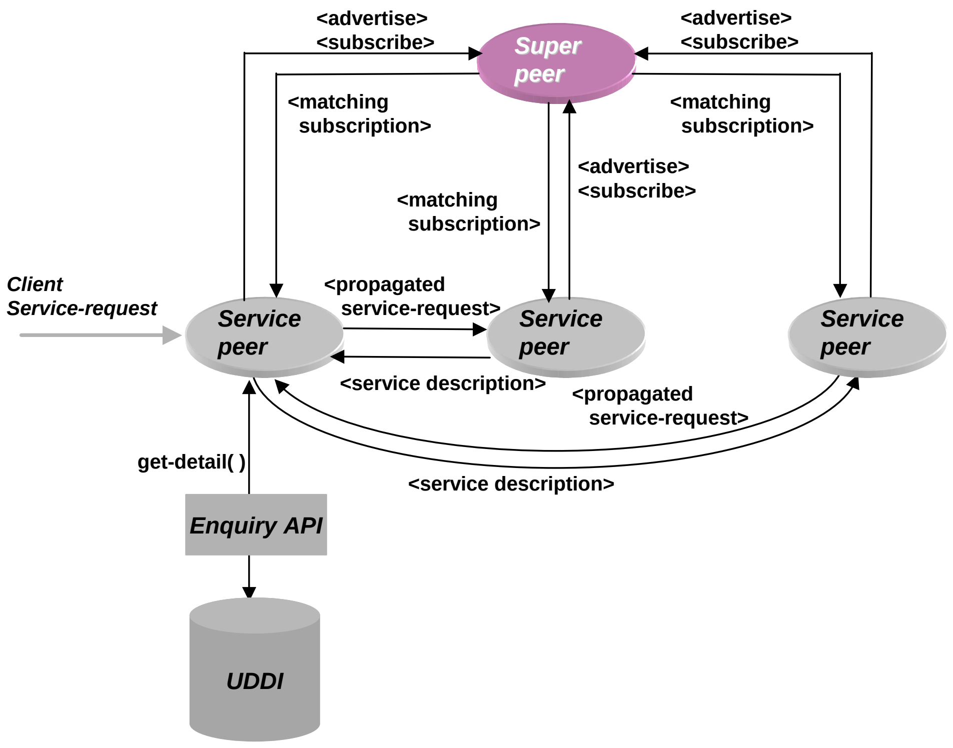


Figure 2: Service request execution within the P2P service network.

way the new peer is informed about existing peers whose publications match its subscription needs and can thus establish its own peer group within the syndication.

Each peer within a service-syndication knows only about the identity of some of the peers in its own syndication. Each peer may act as a mini-directory to specific peers within its syndication to which it is acts as a subscriber. In this way a specific peer can form its own group within the syndication dynamically and has the responsibility to index the ”zone” of the advertisement space that matches its own subscription needs. This peer group is referred to as the *peer-acquaintance group* (PAG). Peers collaborate by propagating service requests to peers within their PAG to which they subscribe (henceforth named *acquainted peers*) and by first receiving descriptions of their service content and by then orchestrating their respective services into service compositions after interacting with the enquiry UDDI API, see Figure 2. Acquainted peers respond to a service request issued by a *local peer* in their syndication by returning a high-level description of their service content such as their `BusinessKey`, its `Tmodel` structure and its `bindindTemplate`. After receiving this information, the local peer invokes the `get_detail()` operations of the UDDI API to retrieve detailed information about the service port-types, elements and bindings of the services offered by its acquainted peers. In this way value-added services can be provided to service clients (requesters).

Advertise/subscribe is a *P2P communication protocol* that enables an exchange of asynchronous notifications between loosely coupled peers in a P2P network of web-services and is based on the publish/subscribe event notification scheme [2]. The key characteristic of the advertise/subscribe protocol is that it de-couples a service supplier (which advertises services) from a client (which subscribes to available data items). As a result the supplier does not have to have any knowledge about which peers are the recipient(s) of its services. This scheme is an attractive option for P2P systems where the peers publish their services when they join the network. At the same time, they can also subscribe to services they are interested in so that peer-syndications can be established dynamically based on matching publications to related subscriptions.

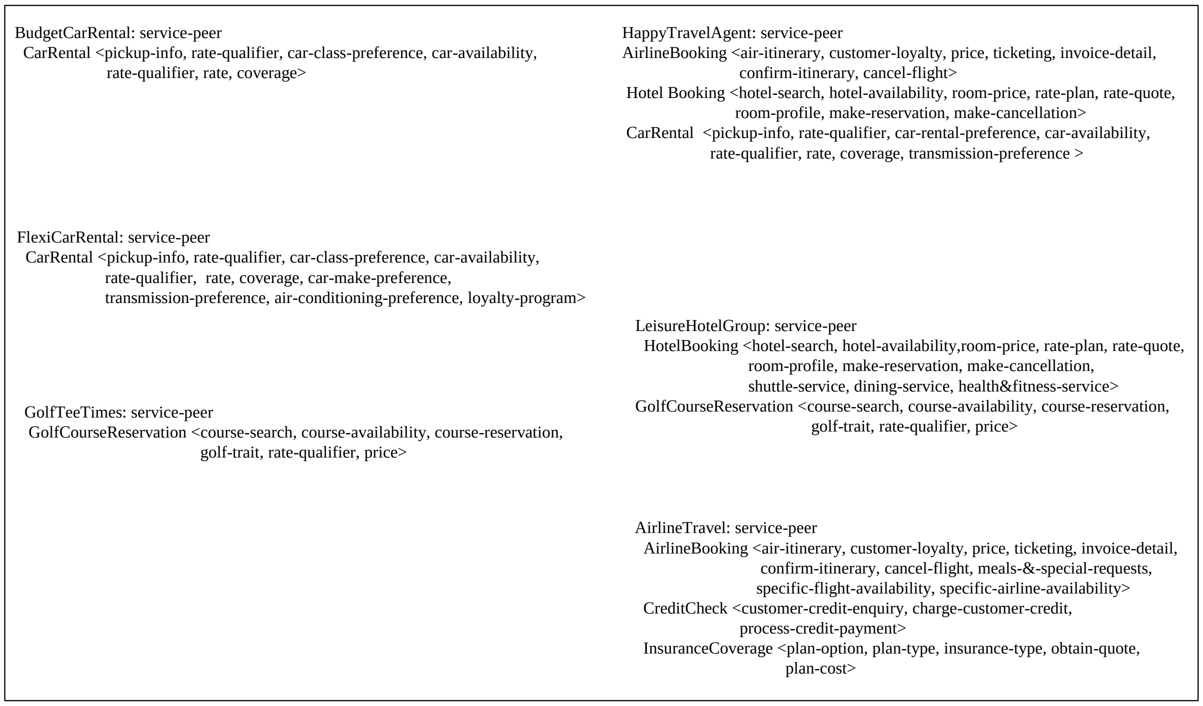
The above P2P service network combines aspects of the directory services P2P model exemplified by Napster [12] and the "pure" P2P architecture exemplified by Gnutella and Freenet [8, 5]. It follows a federated approach where a relatively small number of super-peers provide directory services to peer groups. Peers register high-level information about themselves, such as their name, address and names of the service elements they are willing to provide to other peers in the syndication, with a super-peer. However, they do not use the super-peer to locate or communicate with each other. Instead, peers form cohesive groups (within a syndication) that provide common information and services, e.g., hotel accommodation, recreation activities, car rental, etc, depending on their clients' requirements and interests. Each peer builds up a (constantly changing) peer-group of other peers and stores locally some minimal information about them. Whenever a peer receives a request for service that it cannot fully satisfy it routes segments of it to appropriate peers within its peer-group for execution. Thus, service requests may get propagated from peer to peer within a PAG and responses follow the same path back before interacting with the UDDI enquiry API, see Figure 2. The primary advantages of this approach are scalability and lack of logical centralization.

4 Advertise/Subscribe in the P2P web-services network

The peer-advertise(publish)/subscribe protocol is a communication mechanism that enables the loose coupling of peers in peer-to-peer service networks. The participants of such networks exchange notifications about service advertisements and subscriptions via asynchronous notifications. In the following we describe a service syndication scenario and introduce a formal model for advertisement and subscription matching.

4.1 Service syndication scenario

We base our service syndication scenario on the business domain of e-travelling and, in particular, on the specifications of the open travel agency [13]. OTA has specified a set of standard business processes for searching for availability and booking a reservation in the airline, hotel and car rental industry, as well as the purchase of travel insurance in conjunction with these services. OTA specifications use XML for structured data messages to be exchanged over the Internet.



BudgetCarRental: service-peer
CarRental <pickup-info, rate-qualifier, car-class-preference, car-availability, rate-qualifier, rate, coverage>

FlexiCarRental: service-peer
CarRental <pickup-info, rate-qualifier, car-class-preference, car-availability, rate-qualifier, rate, coverage, car-make-preference, transmission-preference, air-conditioning-preference, loyalty-program>

GolfTeeTimes: service-peer
GolfCourseReservation <course-search, course-availability, course-reservation, golf-trait, rate-qualifier, price>

HappyTravelAgent: service-peer
AirlineBooking <air-itinerary, customer-loyalty, price, ticketing, invoice-detail, confirm-itinerary, cancel-flight>
Hotel Booking <hotel-search, hotel-availability, room-price, rate-plan, rate-quote, room-profile, make-reservation, make-cancellation>
CarRental <pickup-info, rate-qualifier, car-rental-preference, car-availability, rate-qualifier, rate, coverage, transmission-preference >

LeisureHotelGroup: service-peer
HotelBooking <hotel-search, hotel-availability,room-price, rate-plan, rate-quote, room-profile, make-reservation, make-cancellation, shuttle-service, dining-service, health&fitness-service>
GolfCourseReservation <course-search, course-availability, course-reservation, golf-trait, rate-qualifier, price>

AirlineTravel: service-peer
AirlineBooking <air-itinerary, customer-loyalty, price, ticketing, invoice-detail, confirm-itinerary, cancel-flight, meals-&-special-requests, specific-flight-availability, specific-airline-availability>
CreditCheck <customer-credit-enquiry, charge-customer-credit, process-credit-payment>
InsuranceCoverage <plan-option, plan-type, insurance-type, obtain-quote, plan-cost>

Figure 3: Sample advertisements in the e-travel service syndication.

In a publish/subscribe P2P network such as the one illustrated in Figure 2, peers first "agree" to use a common subscription and advertisement template. Peers subscribe their needs (e.g., what services they need to be notified of) with the super-peer in a form of a pattern that needs to be matched.

Figure 3 illustrates possible advertisements made by airline associations, car rental agencies, hotel corporations and leisure operators that have advertised their services as part of an e-travel service syndication. For reasons of brevity we show only six service peers in this figure. Each advertisement consists of the peers name, e.g., FlexiCarRental, along with a `portType` name, e.g., CarRental, and the name of the operations contained in it. We assume that all service providers in the marketplace (and syndication) use standard names for their port types and associated operations. This is common practise with vertical e-marketplaces.

A peer-advertisement is a well-formed XML document, which consists of a set of elements that are arranged in a hierarchy with a single root element named after the advertisement, followed by an XML view of the peer's service operations. A peer-subscription follows a similar convention. For example, Figure 4 shows an excerpt of the advertisement and subscriptions of the `BudgetCarRental`. This figure shows that this peer is interested in subscribing to services offered by hotel service providers as well as other services by car rental operators such as `car-make-preference` and `transmission-preference`.

Different peers in a service-syndication, such as those shown in Figure 3, may compete with each other by offering similar services and by attempting to provide better quality services to their customers in terms of cheaper prices, better package deals, or comprehensive information. At the same time they need to share information, collaborate, and form partnerships by offering composite service suites to provide added-value services to their potential customers in order to gain maximum competitive advantages in their market sector.

4.2 A Formal model for publication and subscription matching

A peer advertises (publishes) a set of port-types, each having a set of operations characterizing the service it offers. For a given service syndication we assume that the set of all possible port-types and operation names **PT** and **OP**, respectively, are well-defined. All advertisements are maintained by a super-peer known to all potential peers **P** (see Figure 1). The sets **PT**, **OP**, and **P** contain all permissible port type, operation, and peer names, respectively.

4.2.1 Publication Contexts

A *publication* is a set of pairs (pt, O) with $pt \in \mathbf{PT}$ and $O \subseteq \mathbf{OP}$. More concretely, we can express an advertisement as:

$$\{(pt_1, \{o_{pt_1,1}, \ldots, o_{pt_1,k_1}\}), \ldots, (pt_l, \{o_{pt_l,1}, \ldots, o_{pt_l,k_l}\})\}. \tag{1}$$

The set of all advertisements known in a specific service syndication at a particular time is called its *publication context*. Publication contexts and the matching of a peer's subscription against a given publication context can be modelled mathematically in terms of formal concept analysis [7],

```
<BudgetCarRental>
   <adverisement source="BudgetCarRental-ad.xml">
     <portType name="CarRentalPortType">
        <operation name="PickUpInfo">
        </operation>
        <operation name="RateQualifier">
        </operation>
           ... ... ... ...
     </portType>
   </advertisement>
   <subscription source="BudgetCarRental-sub.xml">
     <portType name="CarRentalPortType">
        <operation name="car-make-preference">
        </operation>
        <operation name="transmission-preference">
        </operation>
           ... ... ... ...
     </portType>
     <portType name="HotelBookingPortType">
        <operation name="hotel-search">
        </operation>
        <operation name="room-price">
        </operation>
           ... ... ... ...
     </portType>
     <portType name=".... ">
        <operation name="...">
        </operation>
        <operation name="...">
        </operation>
           ... ... ... ...
     </portType>
   </subscription >
</BudgetCarRental >
```

Figure 4: Sample service peer advertisement and subscription.

which relies on the theory of ordered sets and complete lattices.

A given publication context can be represented by a matrix that relates a set of peer names with port-type names and a set of accompanying operation names that peers in the P2P network have advertised. The peer names are represented by rows in the matrix, while the port-types and their associated operations are represented by columns in the matrix. A cross in row P and column pt indicates that peer P has advertised the port-type pt. Table 1 illustrates a publication context for a subset of the port-types and operations used in the example in Figure 3[1].

Formally, a publication context $C := (P, Pt, I)$ consists of a set $P \subseteq \mathbf{P}$ of peer names, a publication $Pt \subseteq \{(pt, O) \mid pt \in \mathbf{PT} \wedge O \subseteq \mathbf{OP}\}$, and an incidence port-type $I \subseteq P \times Pt$. The fact

[1]Due to space limitations, we henceforth represent in all tables and figures operations corresponding to the constructs in Figure 3 by their initial letters only.

	AirlineBk(ai,cl,p,t,id,ci,cf)	AirlineBk(ai,cl,p,t,id,ci,cf,msr,s,fa,saa)	CarRental(pi,rq,ccp,ca,rq,r,c)	CarRental(pi,rq,ccp,ca,rq,r,c,tp)	CarRental(pi,rq,ccp,ca,rq,r,c,tp,acp,lp)	CreditCheck(cce,ccc,pcp)	HotelBk(hs,ha,rpr,rp,rq,rpf,mr,mc)	HotelBk(hs,ha,rpr,rp,rq,rpf,mr,mc,ss,ds,hfs)	GolfCourseRes(cs,ca,ct,rq,p)
AirlineTravel		×				×			
BudgetCarRental			×						
FlexiCarRental					×				
GolfTeeTimes									×
HappyTravelAgent	×			×			×		
LeisureHotelBooking								×	×

Table 1: Example of a publication context for the travel syndication service depicted in Fig. 3

that a peer p has advertised a certain port-type (pt, O) is expressed as $(p, (pt, O)) \in I$. The set of port-type names in Pt is defined by:

$$port-type(Pt) = \{pt \mid (pt, O) \in Pt\} \tag{2}$$

and the set of operations advertised for some port-type name $pt \in PT$ is defined by:

$$operation(pt, Pt) = \{o \mid (pt, O) \in Pt \wedge o \in O\}. \tag{3}$$

The following expression Q' computes the set of *commonly reachable port-types* for a set of peers in $Q \subseteq P$:

$$Q' := \{t \in Pt \mid (p, t) \in I \text{ for all } p \in Q\} \tag{4}$$

Similarly, we define T' as the set of peers that advertised (published) all port-types in a set of port-types $T \subseteq PT$:

$$T' := \{p \in P \mid (p, t) \in I \text{ for all } t \in T\} \tag{5}$$

When we apply definitions 2, 3, 4 and 5 to the publication context C in Table 1, we obtain:

$$port{-}type(C) = \{\texttt{AirlineBooking}, \texttt{CarRental}, \texttt{CreditCheck}, \texttt{HotelBooking}, \texttt{GolfCourseReservation}\}$$

$$operation(\texttt{AirlineBooking}, C) = \{\texttt{ai}, \texttt{cl}, \texttt{p}, \texttt{t}, \texttt{id}, \texttt{ci}, \texttt{cf}, \texttt{msr}, \texttt{s}, \texttt{fa}, \texttt{saa}\}$$

$$\{\texttt{AirlineTravel}\}' = \{\texttt{AirlineBooking}(\texttt{ai}, \texttt{cl}, \texttt{p}, \texttt{t}, \texttt{id}, \texttt{ci}, \texttt{cf}, \texttt{msr}, \texttt{s}, \texttt{fa}, \texttt{saa}), \texttt{CreditCheck}(\texttt{cce}, \texttt{ccc}, \texttt{pcp})\}$$

and

$$\{\texttt{AirlineBooking}(\texttt{ai}, \texttt{cl}, \texttt{p}, \texttt{t}, \texttt{id}, \texttt{ci}, \texttt{cf}, \texttt{msr}, \texttt{s}, \texttt{fa}, \texttt{saa})\}' = \{\texttt{AirlineTravel}\}$$

A simple matrix such as the one illustrated in Table 1 can easily represent the elementary publication context represented by our running example and can be used to compute the results shown in the previous examples. However, for extended publication contexts this construct is impractical and error prone as it does not cross-correlate the entries.

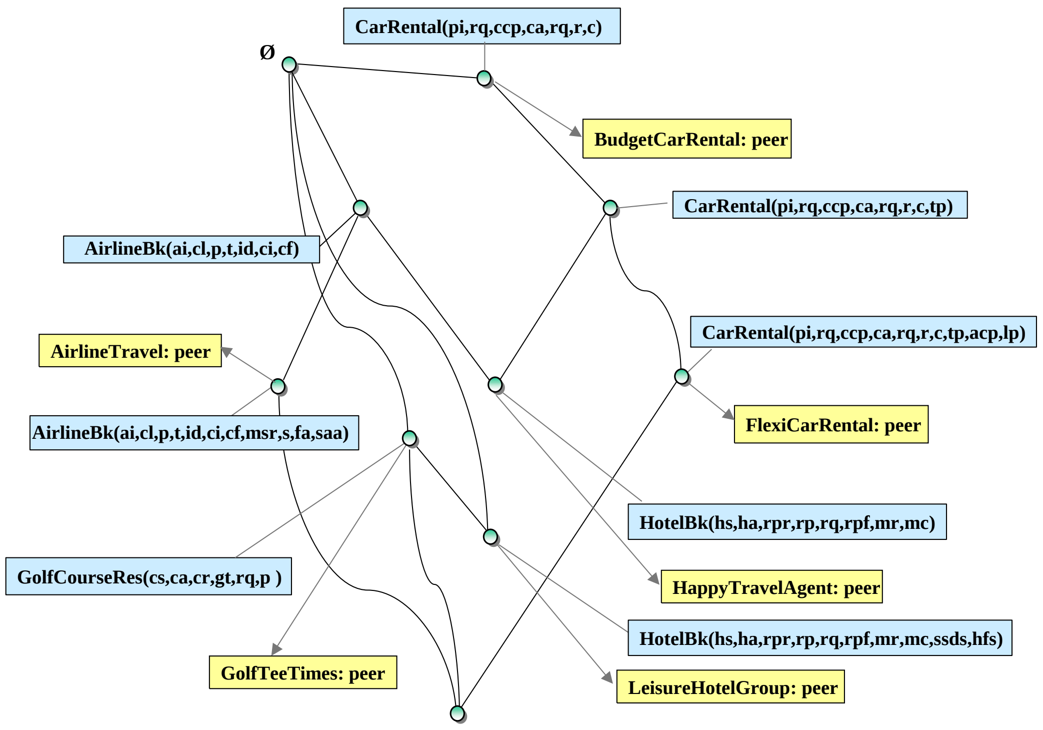

CarRental(pi,rq,ccp,ca,rq,r,c)
Ø
BudgetCarRental: peer
CarRental(pi,rq,ccp,ca,rq,r,c,tp)
AirlineBk(ai,cl,p,t,id,ci,cf)
CarRental(pi,rq,ccp,ca,rq,r,c,tp,acp,lp)
AirlineTravel: peer
AirlineBk(ai,cl,p,t,id,ci,cf,msr,s,fa,saa)
FlexiCarRental: peer
HotelBk(hs,ha,rpr,rp,rq,rpf,mr,mc)
HappyTravelAgent: peer
GolfCourseRes(cs,ca,cr,gt,rq,p)
HotelBk(hs,ha,rpr,rp,rq,rpf,mr,mc,ssds,hfs)
GolfTeeTimes: peer
LeisureHotelGroup: peer

Figure 5: Concept lattice derived from Table 1

A more elegant way to visualise publication contexts and perform context-based computations, is by means of constructing a *concept lattice* $\mathbf{C}(P, Pt, I)$ derived from a given publication context (P, Pt, I) using an efficient algorithm that relies on the notion of formal concepts and formation of sub-context spaces [6]. This algorithm relies on the notion of a *formal concept* of a context (P, Pt, I), which is defined as a pair (Q, T) with $Q \subseteq P, T \subseteq PT, Q' = T$ and $T' = Q$. Q is called the *extent* and T is the *intent* of the concept (Q, T). If two concepts (Q_1, T_1) and (Q_2, T_2) are two concepts of a context, (Q_1, T_1) is called a *subconcept* of (Q_2, T_2) if $Q_1 \subseteq Q_2$ or, equivalently, if $T_2 \subseteq T_1$.

Peer	*Subscription*	*Acquaintances*
Subscription1	{ `CarRental(rq,ccp,rq,r,c` }	{ `BudgetCarRental`, `HappyTravelAgent`,`FlexiCarRental` }
Subscription2	{ `CarRental(rq,ccp,rq,r,c,cmp,tp,ac` }	{ `FlexiCarRental` }
Subscription3	{ `HotelBk(hs,ha,rpr,rp,rq,mr,mc)`, `GolfCourseRes(cs,ca,rq,p)` }	{ `LeisureHotelGroup` }
Subscription4	$\emptyset$	P
Subscription5	{ `HotelBk( )` }	$\emptyset$

Table 2: Examples of peer-subscriptions and their acquaintances.

The concept lattice of the sample publication in Table 1 is shown in Figure 5. There are two types of nodes in this lattice: concept (publication) nodes and peer nodes illustrated by the thinner lines in the figure. All concept nodes reachable upwards from a peer node in the lattice forms the complete concept set that this peer has advertised. For instance, the node labelled with the peer name `FlexiCarRental` has advertised (has the intent):

{ `CarRental(pi,rq,ccp,ca,rq,r,c)`, `CarRental(pi,rq,ccp,ca,rq,r,c,tp)`,
`CarRental(pi,rq,ccp,ca,rq,r,c,tp,acp,lp)` }.

Conversely, all the peer nodes reachable downwards from a concept node in the lattice forms the set of peers that have advertised a common concept. For instance, the concept node `GolfCourtReservation()` is supported by the peer nodes `LeisureHotelGroup, GolfTeeTimes`.

The node labelled with the peer name `BudgetCarRental`, has the extent:

{ `BudgetCarRental`, `FlexiCarRental`, `HappyTravelAgent` }

The extend of a node signifies the nodes reachable via this node by traversing downwards paths.

4.2.2 Peer-subscriptions

A subscription S is specified, just like a publication, as a set of pairs $(s, O))$ with $s \in \mathbf{PT}$ and $O \subseteq \mathbf{OP}$. We say that a peer p *matches* a subscription S if it has published at least the port-types listed in the subscription, i.e., $S \subseteq \{p\}'$. All peers in a publication context (P, Pt, I) that match a subscription form the *peer-acquaintances* of this publication context:

$$[S] := \{p \in P \mid S \subseteq \{p\}'\}. \tag{6}$$

Table 2 shows simple hypothetical examples of peer-subscriptions and their resulting peer-acquaintances.

These examples indicate that Subscription 2 extends Subscription 1 by requesting an additional operations to be matched. We observe that Subscription 2 is a superset of Subscription 1, while the acquaintances of Subscription 2 form a subset of the acquaintances of Subscription 1. An empty subscription is matched by all peers, while subscriptions 5 yields an empty set of acquaintances because the requested combination of operations is not supported by any peer in the syndication that provides `HotelBooking` functionality.

Intuitively, we want the subscription {`HotelBooking()`} to be inter-related with other similar subscriptions within the publication context in Table 1 since there are a number of peers, e.g., `HappyTravelAgent` and `LeisureHotelGroup` that have advertised operations contained in this port-type. This construct means that the service client is interested in some unspecified element of the port-type `HotelBooking` . It is thus useful to view the port-type `HotelBooking()` as a subset of the port-type `HotelBooking(hs, .., mc)` and `HotelBooking(hs, .., mc, ..  , hfs)`. In general, we want a subscription construct (s, O) to be matched by a peer that has published a port-type (pt, B) if $s = pt$ and $O \subseteq B$.

To achieve this, we extend a given publication context (P, Pt, I) to the context $(P, \overline{Pt}, \overline{I})$, where:

$$\overline{Pt} = \{(pt, B) \mid pt \in port-type(Pt) \wedge B \subseteq operation(pt, Pt)\} \tag{7}$$

and

$$\overline{I} = I \cup \{(p, (pt, B)) \mid \text{ if } (p, (pt, O)) \in I \text{ for all } O \subseteq \mathcal{P}(B))\} \tag{8}$$

That is, we supplement the port-types $(pt, O_1), \ldots, (pt, O_n)$ advertised in Pt by additional port-types (pt, B), where B denotes all possible subsets of $O_1 \cup, \cdots \cup O_n$, and we introduce a cross in row p and the new column $(pt, O_i \cup \cdots \cup O_k)$ if we find a cross in all columns $(p, (pt, O_j))$ in the original publication context (for $j = i, i+1, \ldots, k$). Table 3 shows, for illustration purposes, only part of the new matrix resulting from extending Table 1 with definitions 7 and 8. All new columns are marked in a light grey colour. Figure 6 depicts the extended concept lattice (with sub-typing relationships) corresponding to Table 3 and the sample service advertisements in Figure 3. The super-peer organizes the advertisement (publication) space around this type of advertisement lattice and stores it in its own directory (registry in Figure 1). The advertisement-lattice can be used to assist locating peers related to a query that can not be completely decomposed by its query hosting peer, this is explained in some detail in the following section.

To exemplify this consider the extended concept lattice shown in Figure 6 and the subscription `HotelBooking()`. Firstly we need to allocate a concept node that matches this subscription. Then the peer node associated with this concept node (i.e., `HappyTravelAgent`) and all peer nodes associated with the nodes reachable from edges descending from this concept node (i.e., `LeisureHotelGroup`) form the acquaintances of this subscription. If we now attempt to match Subscription 5 in subscription Table 2 against the extended context $(P, \overline{Pt}, \overline{I})$, we obtain the peer

	AirlineBk( )	AirlineBk(ai,cl,p,t,id,ci,cf)	AirlineBk(ai,cl,p,t,id,ci,cf,msr,s,fa,saa)	CarRental( )	CarRental(pi,rq,ccp,ca,rq,r,c)	CarRental(pi,rq,ccp,ca,rq,r,c,tp)	CarRental(pi,rq,ccp,ca,rq,r,c,tp,acp,lp)	HotelBk( )	HotelBk(hs,ha,rpr,rp,rq,rpf,mr,mc)	HotelBk(hs,ha,rpr,rp,rq,rpf,mr,mc,ss,ds,hfs)
AirlineTravel	×	×	×							
BudgetCarRental				×	×					
FlexiCarRental				×	×	×	×			
HappyTravelAgent	×	×		×	×	×		×		
LeisureHotelBooking								×	×	×

Table 3: Example of a supplemented publication context for the travel domain depicted in Fig. 3.

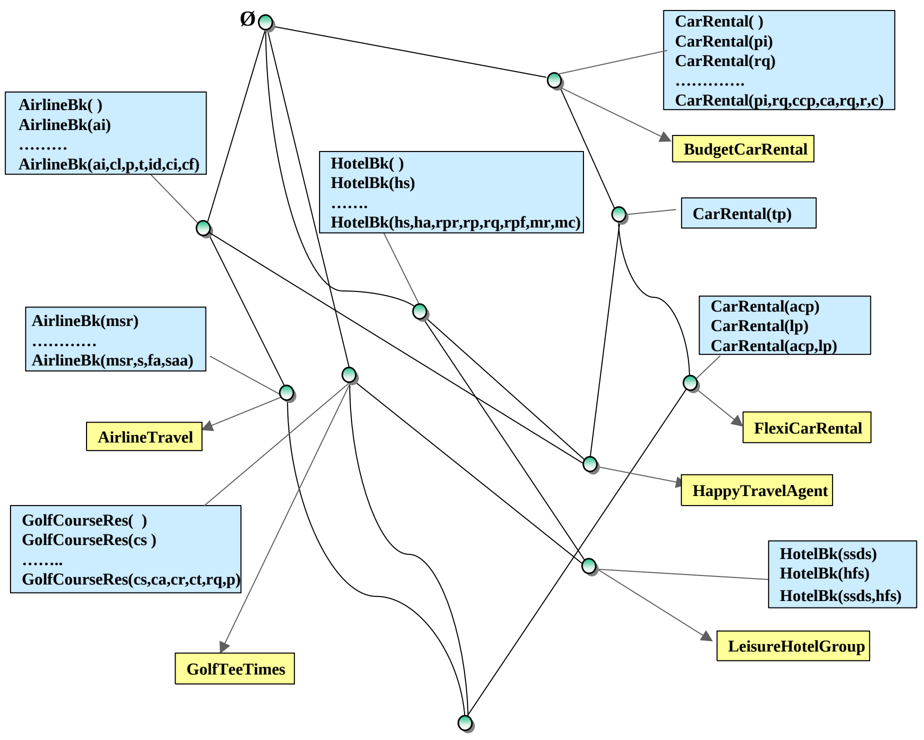


Figure 6: Concept lattice including subtype relationships among advertised services.

acquaintances { HappyTravelAgent, LeisureHotelGroup }.

In addition to the above, we also require that a subscription such as $S = \{\texttt{HotelBooking}(), \texttt{AirlineBooking}(\texttt{ai}, \texttt{cl}, \texttt{pl})\}$ (partially) matches a certain publication context if at least one of the elements in S is matched against an advertisement such as $\texttt{HotelBooking}(\texttt{hs}, \texttt{ha}, .., \texttt{mc})$. We refer to this type of matching as a *partial match*. A partial match of a subscription S over a publication context $C := (P, Pt, I)$, in terms of the function *match*, is defined as follows:

$$match(S, C) = \{p \in P \mid S \cap \{p\}' \neq \emptyset\} \tag{9}$$

4.3 Peer group formation

A peer group is formed by applying the *match* function, defined above, on a peer's subscription against the publication context C. Suppose that the subscription of the peer BudgetCarRental is: $S := \{\texttt{CarRental}(\texttt{tp}), \texttt{HotelBooking}(\texttt{hs}, .., \texttt{mc}))\}$

In other words, we assume that a service provider like BudgetCarRental is interested in providing additional car rental functionality to its own including transmission preference (which it does not provide) as well as hotel booking functionality to its potential clients.

A *matched-subscription lattice* for BudgetCarRental, shown in Figure 7, is generated by matching subscription S against the extended advertisement lattice of the super-peer, shown in Figure 6. The matched-subscription lattice is a true subset of the extended advertisement lattice stored at the super-peer and indicates the peers and matching concepts that conform to BudgetCarRental's subscription. This figure shows that BudgetCarRental has formed a PAG with the following peers: HappyTravelAgent, LeisureHotelGroup and FlexiCarRental. In this PAG LeisureHotelGroup provides the additional hotel functionality requested, FlexiCarRental provides the additional car rental functionality requested, while HappyTravelAgent provides both requested functionalities. Such type of information along with the matched-subscription lattice is stored locally at this peer's site so that it can be used in the future to locate relevant peer acquaintances when attempting to process a service request.

A peer's PAG is fluctuating and dynamic with peer relationships breaking and reforming dynamically depending on the peer's interests and the number of peers entering or leaving its group as a result of this.

Whenever a service request is posed at a local peer, its set of subscriptions needs to be evaluated to determine whether it can support this new request. In case that execution of the new request is not fully supported locally, then a new subscription (for this peer) reflecting the missing information will be generated from the service request and will be sent to the super-peer. Eventually, if it is possible, a new matched-subscription lattice, which satisfies the new subscription, will be generated and sent to the request-hosting peer.

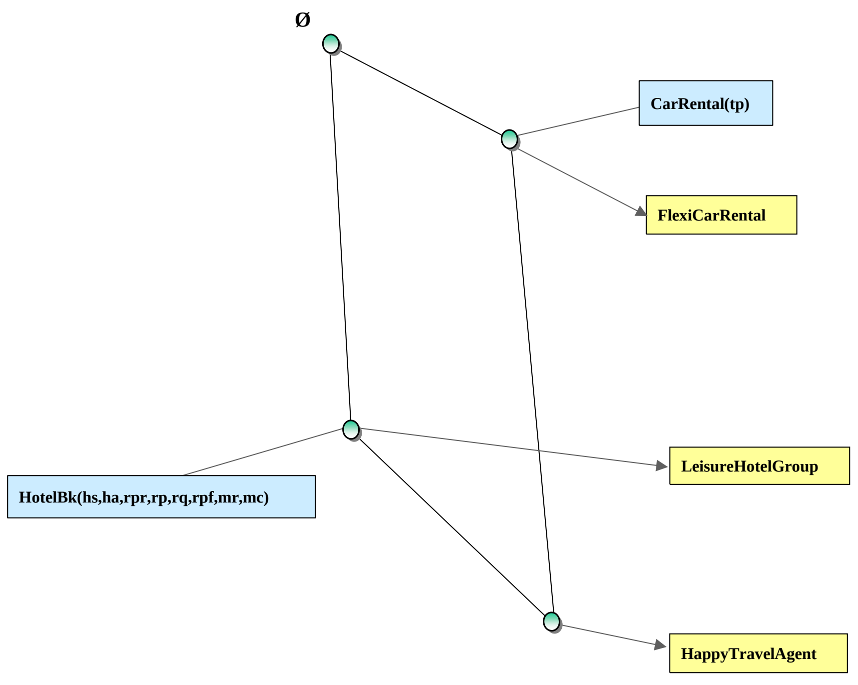


Figure 7: A matched-subscription lattice for `BudgetCarRental`

When a peer decides to leave the P2P network, a notification will be sent from the super-peer to all relevant peers, which in turn will update their matched-subscription lattices. This happens in accordance with the lattice updating algorithms found in [6].

5 Service request processing

Once a peer has formed A PAG it is relatively easy to process requests that deal with information and services supplied by its acquainted peers. When a request arrives, it can be answered however the local peer desires. In many cases, the local peer will use its matched-subscription lattice as a point of reference to resolve a set of acquainted peers in its PAG to which the request would be routed. The response to a request includes descriptions of the service content that the acquainted peers provide in connection to the sub-request they receive. Usually, acquainted peers respond to a service request issued by a peer in their PAG by returning a description of their content that includes `ServiceList`, `bindDetail` and `tModelList` elements, see Figure 2. In this way the local peer receives information about their `BusinessKey`, `Tmodel` structure and their `bindindTemplate`. After receiving this information, the local peer can invoke the `get_Detail` operations of the UDDI enquiry API to retrieve detailed information about the service port types, elements and bindings of the services offered by its acquainted peers.

Assume that `BudgetCarRental` that has formed a PAG with the peers `HappyTravelAgent`, `LeisureHotelGroup` and `FlexiCarRental` and now it needs to handle a sub-request that requires hotel booking functionality, e.g., search for a particular type of hotel. After receiving this request `BudgetCarRental` determines on the basis of its matched-subscription lattice, shown in Figure 7, that its acquainted peers `HappyTravelAgent`, `LeisureHotelGroup` can handle this type of request. Subsequently, this local peer requests technical information from these two peers such as their `ServiceList`, `bindDetail` and `tModelList` structures. After receiving these elements the peer `BudgetCarRental` can interact with UDDI to find more technical details about the technical capabilities of its peer acquaintances and their services. For this purpose the `get_Deatail` operations of the UDDI enquiry API are used. These operations are used to retrieve technical details such as access information required to invoke a service, the technical fingerprint that can be used to recognize a web-service that implements a particular behaviour or its programming interface. This technical information describes the format input messages should be sent in, what protocols are appropriate, what security is required, and what form of a response will result after sending input messages. More specifically, the `get_ServiceDetail`, `get_bindingDetail`, and `get_tModelDetail` functions are invoked to retrieve the above technical details that are required to interact with a service endpoint. If UDDI and WSDL are used together, the `overviewDoc` element of the `tModel`, that is used to provide an overview description of the `tModel` and its intended use, is a WSDL service interface definition.

Now assume that after receiving this technical information the local peer `BudgetCarRental` has decided to invoke one (or both) of its two acquainted peers `HappyTravelAgent` and `LeisureHotelGroup`. Also assume that the service client is interested in hotels within 5 kms North West of Schiphol airport. This request may be expressed as shown in Figure-8. For this purpose we use the Hotel Search Request message provided by the OTA specifications for hotel industry messages (section-5 of OTA 2001 Message specifications). This message provides the ability to search for a list of hotel properties that meet certain criteria, and supply information in return that is related to the specific request. Geographic data, such as proximity to a specific location, landmark, attraction or destination point, could be used to constrain the summary response to a limited number of hotels.

```
<Request HotelSearchRQ xmlns="http://www.opentravel.org/OTA"
        xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
        xsi:noNamespaceSchemaLocation="OTA_HotelSearchRQ.xsd">

        <HotelSearchCriterion Type="Area" MatchType="Exact"
                                         ImportanceType=="Mandatory" >
          <HotelSearchValue>
              <RefPoint Distance="5" DistanceMeasure="km"
               Direction="NW">Schiphol Airport </RefPoint>
          </HotelSearchValue>
       </HotelSearchCriterion>
</Request>
```

Figure 8: Sample service request.

In Figure-8 the statement < HotelSearchRQ > identifies the request as targeting hotel property data. This statement may have one to many < HotelSearchCriterion > child elements that identify a single search criterion by means of criteria types. A < MatchType > attribute indicates whether the match to a string value must be exact or only partial. In many cases the types in the request message may include partial matches to string values such as partial addresses, e.g., street names without a number, or partial telephone numbers. To allow the responding web-service implementation to search for appropriate hotels and respond to preference criteria in the order of importance to the service client, an < ImportanceType > attribute is used. This construct indicates whether the input criterion is mandatory, of high, medium or low priority. The < HotelSearchValue > element is a required child element of < HotelSearchCriterion > that contains the values expected by the `Type` attribute. When the request is a search for a hotel by geographical area, as indicated by the search criterion [Type= "area"], a specific bounded geographic area is specified to locate a hotel. This may include a < RefPoint > element that allows for a search by proximity to a designated reference point, indicating the distance to/from a reference point and direction from the reference point.

The response to the request in Figure-8 returns a list of hotel properties that meet the criteria of the request. The response message does not attempt to provide any information about the availability of a property, it rather supplies an identification of the hotels selected, and descriptive information that may allow the service requester to make a final selection in a more logical or personalized fashion. A sample response to the request of Figure-8 is found in Figure-9.

In this manner the local peer `BudgetCarRental` can offer hotel booking functionality by collaborating with its acquaintances in its PAG to allow for added-value functions to be created as an aggregation of services from multiple service providers.

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_HotelSearchRS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="OTA_HotelSearchRS.xsd" xmlns="http://www.opentravel.org/OTA">
<Success/>

<HotelSearchRecord HotelName="Hilton Hotel" Relevance="100">
     <HotelReference ChainCode="HH" BrandCode=".." HotelCode=".."/>
      <LocationDescription> 3.5 kms NW of Schiphol Airport</LocationDescription>
      <SearchValueMatch Match="true">Deluxe</SearchValueMatch>
      <MarketingText>Pool, Spa, and Health Club on premises</MarketingText>
</HotelSearchRecord>

    ... ... ...

<TotalReturns> 5 </TotalReturns>
</OTA_HotelSearchRS>
```

Figure 9: Sample service response.

6 Related Work

P2P systems are usually characterised by a centralized Napster model [12] where a central server routes all the queries, a fully decentralized Gnutella model [8] where a every query is propagated to every peer or a hybrid model where peers can act as hubs and can register with other hubs as information providers to field queries from other peers based on arbitrary content description registrations.

In [1] a classification scheme is proposed for categorizing distributed (hybrid P2P) searches as *content-agnostic* and *content-based*. In content agnostic searches, the organization of the peers does not depend directly on the resources they index or point to, rather the focus lies on the organization of peers and the maximum distance between peers. With this scheme, queries need to reach a subset of peers that contain a complete set of published advertisements, in order to guarantee an exhaustive search. In contrast to this, in content-based approaches, the content of queries is used to efficiently route messages to the most relevant peers. The peers may also be organized based on the content they index to allow fast traversal.

In this paper we reported on a hybrid P2P content-based network that unifies peer-to-peer with service-oriented computing (exemplified by the web-services paradigm). This approach occupies a middle ground between the centralized Napster and decentralized Gnutella configurations. In the following we will discuss several research activities that are in the spirit of content-based P2P search models and briefly compare them with our approach.

Recent work in content-based search include content-addressable networks such as CAN [14], Chord [3], and Pastry [4] as well as some variations of publish/subscribe networks [11].

The CAN network resembles a distributed, Internet-scale, hash table where the basic operations performed are insertion, lookup and deletion of (key,value) pairs. CAN is composed of many nodes (peers) where each peer logically occupies (stores) a zone of the entire hash table and holds information about a small number of adjacent zones in the table. Content and queries are mapped into d dimensions using d global hash functions. Routing is performed from the source to destination along the coordinate that reduces the remaining distance to destination. The CAN algorithm assumes that peers have somewhat similar capabilities and resources, and that they are equally reliable. Moreover, peers in the CAN approach only maintain knowledge of their local neighbours.

Chord [3] is another scheme for content mapping to different peers. It resolves a key to a specific peer using consistent hashing. The search cost in Cord is logarithmic in the number of peers. Extensions to the current Chord search algorithm include a location table where each peer can maintain information about other peers it encountered, and how close they are in physical distance. Another optimisation is to allow peers to have several ids so that they can even out the distribution of keys to the different peers. This can also make more reliable and powerful peers responsible for indexing more of the content space.

The publish/subscribe model has been proposed for file-sharing networks [11], using an equivalent query/advertise model. Clients publish their resources with servers, using a pattern. The patterns are then propagated to other servers in the network and aggregated. Queries are routed to clients who published a relevant pattern. A disadvantage of this approach is that it is not scalable as subscriptions need to be forwarded to all servers.

On the industrial research side the work that comes closer to the P2P database network reported herein is Sun Microsystem's Project Juxtapose (http://www.jxta.org) – usually referred to as JXTA [15]. JXTA defines a set of XML-based (language and network agnostic) protocols for interoperation and an open network programming platform to enable P2P services and applications. One of the core concepts in JXTA is the notion of an advertisement, an XML-based document that names, describes and publishes the existence of a P2P resource. JXTA specifications include a set of advertisements that can be broken down into subtypes using XML schemas. Key entities defined by XML include peers, any entity that can understand protocols required by other peers; messages that are designed as datagrams containing an envelope and a set of protocol headers with associated bodies; identifies to refer top entities such as peers, advertisements and services; and peer groups, which are collections of cooperating peers that provide a common set of circumstances.

There are certain similarities between our approach and the ones described above, especially JXTA. These include advertisement, registration and query (request) resolution. The main difference between our approach and JXTA is that JXTA tends to centralize registration/subscription information and control in hubs, whereas in our approach peers register only high-level information about themselves, such as their name, address and names of the service elements they are willing to share with other peers, with a super-peer. However, they do not use the super-peer to locate or communicate with each other. Instead, each peer builds up a (constantly changing) peer-group of other peers and stores some minimal information about them.

The content-based P2P networks described above place emphasis on discovery of content rather than on a logical organization of the information space and on establishing relationships between constructs in the advertisement space. A key difference between our approach and those mentioned above is that is that our approach structures the advertisement and subscription space in lattices that logically organize the advertisement/publication content by establishing semantic links and relationships between content concepts such as port-type and operation names.

7 Conclusion

Despite the fact that service-oriented and peer-to-peer-computing have some significant amount of overlap, e.g., each seeks to become a common means for publishing and discovery across the Internet, their current manifestations still remain quite diverse. In this paper we highlighted key intersect points that enable possibilities for using these two technologies together and presented an architectural approach and formal framework towards unifying them.

We introduced a federation of UDDI-enabled, peer registries that operate in a decentralized fashion, rather than requiring each peer to publish their own service descriptors locally or centrally (on the UDDI). Federations represent common interest groups of peers that band together to ensure that they provide added-value syndicated-services to their customers.

Service-providers (peers) establish advertisements describing the port-types and operations of their available services. Requests in this service-oriented P2P network are represented by the content of messages describing services of interest. Upon receiving a service request from a local peer, acquainted peers generate response messages whose content describes the attributes of their specific services. These responses are injected into the network and routed back to the request originator.

We also used a mathematical model in terms of formal concept analysis, which relies on the theory of ordered sets and complete lattices to represent advertisement (publication) contexts and subscriptions and showed how subscriptions and related publications can be fully and partially matched. Finally, we explained how peer (service)-syndications seek to promote in-demand services by offering sets of related services throughout the federation rather than on a single location, which is the standard case with current technology.

There are several benefits to the advertise/subscribe model we employed for P2P networks of web-services. Request routing is based on complex content and semantic relationships between sets of similar services offered by different providers. This reduces the sizes of routing tables and gives scope to the search while permitting group specific searches and policies. In particular, because of the use of super-peers, which have some form of central control, it is much easier to implement policies for security, membership and accounting.

References

[1] S. Botros and S. Waterhouse, "Search in JXTA and other Distributed Networks," Proc. 2001 Int'l Conf. Peer-to-Peer Computing, 2001;
available online at http://people.jxta.org/stevew/BotrosWaterhouse2001.pdf.

[2] A. Carzaniga, D. Rosenblum, and A. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service", in ACM Trans on Computer Systems, Vol. 19, No. 3, Page 332-383, 2001.

[3] F. Dabek, et. al. "Building Peer-to-Peer Systems with Chord, a distributed Lookup Service", http://pdos.lcs.mit.edu/chord2001.

[4] P. Druschel and A. Rowstron "Pastry: Scalable Distributed Object Location and Routing for Large Scale Peer-to-Peer Systems", ACM SIGCOMM, 2001.

[5] Freenet Home Page. http://freenet.sourceforge.com

[6] Algorithmen zur formalen Begriffsanalyse. In: B. Ganter, R. Wille, and K.E. Wolff (eds), *Beitrge zur Begriffsanalyse*. B.I.-Wissenschaftsverlag, Mannheim, 1987, pp. 241-254

[7] B. Ganter, R. Wille. *Formal Concept Analysis*. Springer 1999

[8] Gnutella Development Home Page. http://gnutella.wego.com

[9] L. Gong. "Project JXTA: A Technology Overview",
http://www.jxta.org/project/www/white_papers.html.

[10] D. Gonovi "Web-services over P2P Networks", in Web Services Journal, pp. 8-13, April 2002.

[11] D. Heimbigner "Adapting Publish/Subscribe Middleware to Achieve Gnutella-like Functionality", ACM Symposium on Advanced Computing (SAC), 2001.

[12] www.napster.com, 2001.

[13] Open Travel Alliance (OTA), "Document 2001C Specifications", http://www.opentravel.org, 2001.

[14] S. Ratnasamy, et. al. "A Scalable Content Addressable Network", ACM SIGCOMM, 2001.

[15] S. R. Waterhouse, D. M. Doolin, G. Kan and Y. Faybishenko, "JXTA Search: a Distributed Search Framework for Peer-to-Peer Networks" in IEEE Internet Computing, vol. 6, pp 68-73, 2002.

[16] B. Yang, H. Garcia-Molina. "Comparing Hybrid Peer-to-Peer Systems". In Procs of VLDB 2001, Roma, September, 2001.