Formation of Service Value Networks for Decentralized Service Provisioning

Sebastian Speiser[1], Benjamin Blau[2], Steffen Lamparter[1,2], Stefan Tai[1,2]

[1] Institute AIFB, Universität Karlsruhe (TH), Germany
lastname@aifb.uni-karlsruhe.de
[2] Karlsruhe Service Research Institute (KSRI), Universität Karlsruhe (TH), Germany
firstname.lastname@ksri.uni-karlsruhe.de

Abstract. The provisioning of complex services requires tight collaboration between diverse service providers and their customers harmonizing supply and demand chains to a highly flexible, dynamic and decentralized service value network. Peers in such a network autonomously delegate (sub-)tasks which cannot be done efficiently by themselves to other more suitable peers in their community. In this paper, we propose an architecture for such service communities that features decentralized service provisioning based on current Web technologies. In this context, we present an algorithm for efficient service value network formation and show by means of a simulation that sufficiently sized service networks can fulfill practically all customer requests. When compared to the optimal (central) case, there is a modest price increase for the customers but the overall welfare decreases only insignificantly.

1 Introduction

Complex (or *composite*) services "typically involve the assembly and invocation of many pre-existing services possibly found in diverse enterprises [1]," and thus, a network of service providers and consumers. In modern services-led economies, such networks increasingly are loosely-coupled configurations of legally independent firms.

The formation of such networks is driven by the value that the network generates for its customers. With increasing competition and specialization in the services sector, and the continuous introduction of new services offerings, value-driven network formation and transformation is of predominant importance. Dynamic *service value networks* are often considered as the only strategic alternative to provide complex services [2–5].

Due to low lock-in and lock-out costs, service value networks are characterized by a high rate of fluctuation of service providers. Monetary barriers for entering and leaving strategic positions within the network are relatively low compared to traditional supply chains in product-oriented economies. Such an environment fosters flexible and interchangeable business relations that co-produce value in a highly dynamic manner. Service networks describe the possible cooperations between legally independent actors that enable co-generation of value by fulfilling complex customer requests.

The high degree of flexibility and alteration of business connections in service networks implies a vast amount of information regarding participating players and their

interrelations to be available. An efficient collection and management of this information in a central repository is however not feasible. It is neither desirable nor possible for all directly and indirectly connected partners of a network to provide continuous, consistent information to a central repository; central service repositories have found little acceptance in practice for even less demanding simple service discovery scenarios.

Consequently, this paper analyzes the formation of service value networks in a decentralized service provisioning environment. Based on customers' requests for the completion of complex tasks, peers in such a network autonomously delegate (sub-) tasks to other service providers within their partner networks. This iterative process fosters the evolution of a network-based value generation driven by customers' needs and co-opetition of specialized service providers.

In this paper, we propose an architecture for such service networks that features decentralized service provisioning based on current Web technologies. In this context, we present an algorithm for efficient service value network formation and show by means of a simulation that sufficiently sized service networks can fulfill practically all customer requests. When compared to the optimal (central) case, there is a modest price increase for the customers but the overall welfare decreases only insignificantly.

The paper is structured as follows. In Section 2 we define our model of composed services and service value networks. Based thereon in Section 3 we introduce our proposed algorithms for decentralized service provisioning. Section 4 describes a simulation of the algorithms and compares them to centralized approaches. A system architecture that can be used to implement our approach is presented in Section 5. Related work is discussed in Section 6 before we finally conclude and give an outlook on future work in Section 7.

2 Service Value Networks

A customer may have a need for a complex service that no single service provider can fulfill all by himself. Decomposing the complex service into more basic services that each can be delivered by at least one provider enables the satisfaction of the customer's need. The cooperation of the different providers and the relationship to the customer form a service value network, as a value is cogenerated by delivering services through a network of actors.

The received value for the customer is determined by his valuation of the service minus the price he has to pay. He will only accept an offer if the price is below his valuation. Providers receive a value given by the received payments minus the payments to other providers and the internal costs they are imposed when executing a service or part of it themselves. We want to maximize the welfare in the network which is the sum of the received values for all participants. A payment results in the same amount of decreased value for the payer and increased value for the receiver. Therefore payments are neutral to the welfare, which is only determined by the customers valuation of the service minus the internal costs of the involved providers. Therefore the goal for the network is to minimize the internal costs and ensure that the total price is below the customer's valuation.

In a centralized approach an algorithm that has access to all providers, their capabilities and the associated costs can compute optimal service assignments for every customer request. Since such a central authority has failed in practice, we propose and analyze a decentralized scenario where a customer sends service requests to a limited set of providers, which in turn cooperate with a limited set of other providers. Every provider knows about his capabilities and costs. As he is aiming at making a profit he will add a margin to his costs and then advertise his capabilities with associated price functions to his partners. Through these advertisements a provider knows which services from his partner he can combine with his own capabilities to form new services he can deliver. As the provider can also charge money for the delegation of services, it may be desirable to become a pure service trader, whose business assets are his relationship to customers and other providers.

In this section we introduce a formal model that is the foundation for the algorithms proposed in Section 3. First we define a *service specification* which specifies how *atomic services* are combined to form more complex *composed services*. Afterwards service providers and customers are introduced that are connected in service network. For the fulfillment of service requests the network dynamically forms a service value network (SVN). At last we define the value that is generated by a SVN.

Service Specification Services that cannot be decomposed in smaller sub-services are called *atomic*. In contrast a *composed* service is equivalent to its *component* services that can in turn again be atomic or composed. Let S be the set of all services. The set of ordered pairs E_S represents the component relationship meaning that $s_1 \in S$ is a component service of $s_2 \in S$, iff $(s_1, s_2) \in E_S$. The directed graph $G_S = (S, E_S)$ is called *service specification*. We require G_s to be acyclic. Furthermore we define the two disjoint sets of *atomic* services $S_a = \{s \in S \mid \nexists s' \in S : (s', s) \in E_S\}$ and *composed* services $S_c = \{s \in S \mid \exists s' \in S : (s', s) \in E_T\} = S \setminus S_b$. The following functions will be helpful:

- $\texttt{components}(s) = \{s' \in S \mid (s', s) \in E_S\}$, returns all component services of service s.
- $\texttt{offspring}(s) = \bigcup_{s' \in \texttt{components}(s)} \left(\{s'\} \cup \texttt{offspring}(s')\right)$, returns recursively all services that are direct or indirect components of s.
- $\texttt{complexity}(s) = |(\{s\} \cup \texttt{offspring}(t)) \cap S_b|$, returns the complexity of a service, defined as the number of atomic services required to create an equivalent service.
- $\texttt{fathers}(s) = \{s' \in S \mid (s, s') \in E_S\}$, returns the services of which s is a component.

Service Provider Each service provider p is able to execute the services denoted by his capability set $\phi_p \in R$. For the execution of a service $s \in \phi_p$ the provider is charged with internal costs, determined by his cost function $c_p : S \longrightarrow \mathbb{R}$. As a provider aims at making a profit he charges a price that adds a margin to his costs, given by the margin function $m_p : \mathbb{R} \longrightarrow \mathbb{R}$. The charged price is given by the function $f_p(s) = m_p(c_p(s)), s \in S$. The provider is able to subcontract services to a set of cooperating

service providers, denoted as V_p. The costs for p when subcontracting $s \notin \phi_p$ are given by $c_p(s) = \min_{p' \in V_p} f_{p'}(s)$.

For each service s that a provider p can offer either himself or in cooperation he maintains the information if the service should be executed or delegated as a whole or decomposed into its components. This information is represented in the function $d_p : S \longrightarrow \{false, true\}$, which evaluates to true if the corresponding service should be decomposed. For all services that should not be decomposed the provider has a mapping $sa_p : S \longrightarrow P$ that assigns a service to a provider, possibly himself.

Customer Besides service providers that delegate services to subcontractors we consider pure customers that want to consume services but do not provide services themselves nor forward requests. Let W be the set of customers. A customer $w \in W$ has a valuation for the services he wants to request, given by his utility function $u_w : S \longrightarrow \mathbb{R}$. The service providers to which w sends service requests are given by the set V_w.

Service Value Network The relationships between customers, providers and among providers are represented by the directed graph $G = (P \cup W, E)$ where an edge $(x, y) \in E$ denotes that x sends requests to y. The edges are given by $E = \{(x, y) \in P \times P \mid y \in V_x\} \cup \{(x, y) \in W \times P \mid y \in V_x\}$. The graph G is called *service network*. The customer that requests a service and the providers that are involved in the fulfillment of the request form a *service value network*. As this process is invoked for every service request it is a *dynamic formation*.

Let $P' \subset P \cup W$ be the set of participants in a service value network fulfilling the request for service s by customer w. Service delegations are represented as tuples $(p_1, p_2, s) \subset P' \times P' \times S$, meaning that p_1 delegates service s to p_2. Internal executions are also treated as service delegations with $p_1 = p_2$. Let E' be the set of all service delegations then the directed graph $G' = (P', E')$ denotes the service value network. For the reader's convenience we define the following three sets for a service value network:

- The customer's request $R_{G'} = \{(w, p, s) \in E' \mid w \in W\}$.
- $I_{G'} = \{(p_1, p_2, s) \in E' \mid p_1 = p_2\}$, containing the internal executions.
- $D_{G'} = E' \setminus (R_{G'} \cup I_{G'})$, the set of service cooperations.

Welfare in Service Value Networks The welfare cogenerated in a service value network is given by the sum of received values for all participants. For a service value network $G' = (P', E')$ which serves customer w with service s, we define the welfare $\mathtt{wf}_{G'}$ as

$$\underbrace{\sum_{(w',p,s') \in R_{G'}} u_{w'}(s')}_{\text{customer's valuation}} + \underbrace{\sum_{(p_1,p_2,s') \in D_{G'}} f_{p_2}(s') - f_{p_2}(s')}_{\text{received payments - paid prices}} - \underbrace{\sum_{(p,p,s') \in I_{G'}} c_p(s')}_{\text{internal costs}}$$

$$= u_w(s) - \sum_{(p,p,s') \in I_{G'}} c_p(s').$$

This shows that the welfare is independent of payments and can be maximized if the internal costs that occur during the execution of a service are minimized. The payments play however a role as the total price has to be below the customer's valuation.

3 Network Formation and Service Delivery Algorithms

The algorithms can be divided into two groups. The first initializes and maintains the data structures a provider keeps to determine the best executions strategies for each service. Based on this data the second group generates concrete offers upon service requests.

In the code of the algorithms we use the notation $f[x] := y$ with the meaning that after this statement, the function f evaluates to y for the argument x.

Algorithm 1 `initProvider`

Require: Provider p

1: **for all** $s \in S$ **do**
2: $sa_p[s] := p$
3: $d_p[s] := false$
4: **if** $s \in \phi_p$ **then**
5: $f_p[s] := m_p(c_p(s))$
6: **else**
7: $c_p[s] := \infty$
8: **end if**
9: **end for**
10: **for all** $s \in \phi_p$ **do**
11: **for all** $p' \in V_p$ **do**
12: `notifyProvider`(p' , p , s , f_p)
13: **end for**
14: **end for**

Every new provider p that joins the service value network executes the function `initProvider` (see Algorithm 1). First all services $s \in S$ are assigned to p himself and either get associated costs of ∞, if p cannot deliver the service or the price is set by adding the margin on the internal costs. Afterwards for all services in the capability set of the provider a notification is sent to the partner network that p can execute s at a price determined by the function f_p.

A provider p that receives a notification via the function `notifyProvider` about the capability of p' to deliver s at price $f_{p'}$, first checks if the new price is better than his current costs (see Algorithm 2). If this is the case he updates the preferred provider for the service and sets the costs to the received price. He also updates the price $f_p(s)$ he charges for the service to be the costs plus his margin. Then he calls the function `updateCosts`.

This function is provided by Algorithm 3 and first notifies the partner network about the new capability or respectively the new price. Then it is checked if the updated service is part of composed services. In that case for each composed service the sum of

Algorithm 2 `notifyProvider`

Require: Provider p, Provider p', Service s, Function $f_{p'}$

1: **if** $c_p(s) > f_{p'}(s)$ **then**
2: $sa_p[s] := p'$
3: $d_p[s] := \texttt{false}$
4: $c_p[s] := f_{p'}(s)$
5: $f_p[s] := m_p(c_p(s))$
6: `updateCosts`(p,s)
7: **end if**

Algorithm 3 `updateCosts`

Require: Provider p, Service s_c

1: **for all** $p' \in V_p$ **do**
2: `notifyProvider`(p',p,s_c,f_p)
3: **end for**
4: **if** $\texttt{fathers}(s_c) <> \emptyset$ **then**
5: **for all** $s \in \texttt{fathers}(s_c)$ **do**
6: **if** $\sum_{s' \in \texttt{components}(s)} c_p(s') < c_p(s)$ **then**
7: $sa_p[s] := p$
8: $d_p[s] := true$
9: $c_p[s] := \sum_{s' \in \texttt{components}(s)} c_p(s')$
10: $f_p[s] := m_p(c_p(s))$
11: `updateCosts`(p,s)
12: **end if**
13: **end for**
14: **end if**

the prices for the components is compared to the current total price. If a decomposition is cheaper this is saved in the provider's data structure and `updateCosts` is called recursively for the composed services.

Algorithm 4 `makeOffer`

Require: Provider p, Service s
Ensure: Price for offering s

1: **return** $m_p(\texttt{getCosts}(p, s))$

A concrete offer for a service s by provider p is requested by calling the function `makeOffer(`p, s`)`, which simply adds p's margin on his costs for delivering s (see Algorithm 4). The costs are calculated by the function `getCosts` (see Algorithm 5) that distinguishes the following three cases. If s is marked for decomposed execution, the costs for its components are added by recursively calling `getCosts`. If p is the preferred provider for s, the internal costs are taken. Else an offer from the provider that p has assigned for s is requested.

Algorithm 5 `getCosts`

Require: Provider p, Service s
Ensure: Costs that p has to spend for delivering s

```
1: if d_p(s) then
2:     return Σ_{s' ∈ components(s)} getCosts(p, s')
3: else
4:     if sa_p(s) = p then
5:         return c_p(s)
6:     else
7:         return makeOffer(sa_p(s), s)
8:     end if
9: end if
```

Correctness and Scalability Initially a provider assigns all services to himself and memorizes as costs either his internal costs or infinity for services that he is not capable of doing. In the succeeding phase the provider broadcasts his capabilities. We can assume that each provider adds a margin that leads to increasing prices ($\forall x \in \mathbb{R}, p \in P : m_p(x) > x$). Therefore at some point providers will dismiss further notifications based on their price. The notification phase is very similar to the *Routing Information Protocol* (RIP) [6]. With RIP routers on the internet exchange information about which networks they can reach. The costs are measured as the number of intermediary routers that are used by a router to reach the network. The different network speeds are equivalent to the margins charged by providers. RIP has proven to be correct by operating reliably the internet and other networks. However it has some scalability issues and is therefore replaced more and more by link-state based routing protocols, like *OSPF* [7]. These protocols are not applicable to our problem domain, as they require that providers can gather complete topological information about the service network, including the margins of other providers. This is not a realistic option. The scalability issues will not be a problem in the near future as the service networks are supposed to be much smaller than the internet.

Routers operating with RIP resend their routing information every 30 seconds and delete routes that are not confirmed by such resendings. In this way the protocol deals with the removal of links or routers. This can also be applied to our algorithms in order to react to price increases or false advertisements. False advertisements are notifications of providers that they can deliver a service for a given price but always return higher prices in the offer phase. Other providers can detect such a behavior and remove the provider from their partner network.

In the offer making phase a provider knows exactly if he should decompose a service and to which other providers services are delegated. This efficiency for the frequent service offers is bought with increased costs for the propagation of changes in the service value network.

4 Network Simulation

We ran simulations of service value networks in order to compare them to an approach with a central registry. The following questions are analyzed with the simulation results:

1. What is the performance of the notification algorithm?
2. How many service requests can be fulfilled?
3. How large is the price increase for customers?
4. What is the impact on the welfare?

For the experiment we first create a service specification consisting of n_s services. Composed services have at least 2 and at most 5 components. Each of the n_p providers is capable of doing a randomly selected service s with internal costs randomly selected between $0.2 \cdot \texttt{complexity}(s)$ and $0.8 \cdot \texttt{complexity}(s)$. It is insured that every service has at least one provider. The margin function for a provider p is created by drawing a random number M_p of the interval $[0.1, 0.2]$ and taking it at as a profit percentage: $m_p(x) = (1+M_p) \cdot x$. The partner network is dependent on the chosen density d. Between the providers $n_e = d \cdot \frac{n_p(n_p-1)}{2}$ random partnerships are established. The customer's valuation of a service s is given by $\texttt{complexity}(s)$.

For the evaluation of our approach we compare it to the case where a central service registry exists. This can be modelled as a provider that has partnerships with all other providers and charges no margin. The registry is able to serve a customer all services with optimal price. In some comparisons we assume that the registry also knows about the internal costs of individual providers and can therefore provide service executions with optimal welfare.

We ran an experiment testing the performance of the algorithm and the quality of the service value networks as the number of providers grows. For each run we created a random service specification consisting of $n_s = 50$ services and a random service network with density $d = 0.1$. The experiment was run 100 times for each n_p which was varied from 50 to 400 in steps of 25 and the averages values of the following variables were recorded:

- $\texttt{ex}(n_p)$: decentral execution ratio, meaning the probability that a request to a random provider for a random service results in a price that is lower than the valuation of the service.
- $\texttt{wf\_dec}(n_p)$: the average percental welfare decrease when a customer requests a random provider for a random service, compared to an optimal execution.
- $\texttt{price\_inc}(n_p)$: the average percental price increase for a customer requesting a random provider for a random service, compared to requesting the same service at a central registry.
- $\texttt{n\_notifications}(n_p)$: number of notifications that are sent between providers until all service pricing information is exchanged.

The values $\texttt{wf\_dec}(n_p)$ and $\texttt{price\_inc}(n_p)$ are only collected for executable services, in order to avoid infinite values. Note that all service offers from the central registry are priced below the valuation and therefore executable (as the maximum costs 0.8 times the maximum margin 1.2 results in maximum price of 0.96 per atomic service).

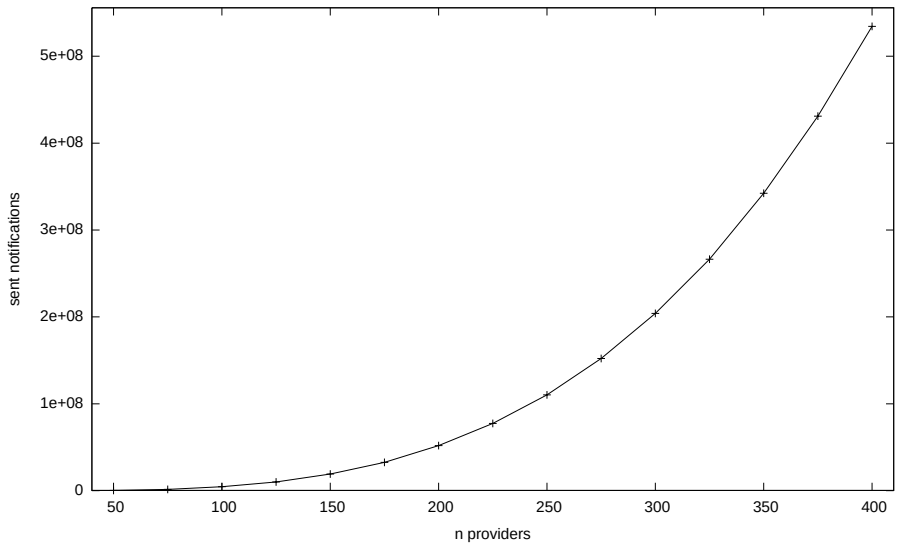

Fig. 1. Number of notifications $\texttt{n\_notifications}(n_p)$ until service delegations are stable for varying n_p from 50 to 400. Fixed: 50 Services, 0.1 density.

Performance of Algorithm (Question 1) The number of needed notifications in the experiment is plotted in Figure 1. Our experimental data suggest that it grows with speed $O(n^3)$. As the number of providers also grows, this means an average number of $O(n^2)$ notifications per provider. Therefore the algorithm can be considered efficient although it was simulated with sequential notification phases. More realistic is that all providers concurrently start their notifications, which leads to fewer notifications as there is no worst-case where each provider one after another notifies about a new best price.

Decentral Execution Ratio (Question 2) In Figure 2 we plot the ratio of executable customer requests in our experiment. Even with a small number of providers the ratio is around 80% to 90% and converges with increasing number of providers fast to the optimal value of 100%. In our experiment from 200 providers on, meaning on average every service can be executed by 4 providers, almost every request can be fulfilled.

Price Increase for Customer (Question 3) We see in Figure 3 that the price increase for customers gets smaller with increasing size of the service network. In our simulation there is an inherent reason why a zero price increase is not possible: every provider can execute himself only one service and therefore has to delegate all other services, which means for almost all requests at least two providers are involved that both charge a margin. In the central case we assumed that the registry does not charge money and can therefore offer the best available price. The operation of such a central registry however is associated with costs that have to be reimbursed either by charging a margin or

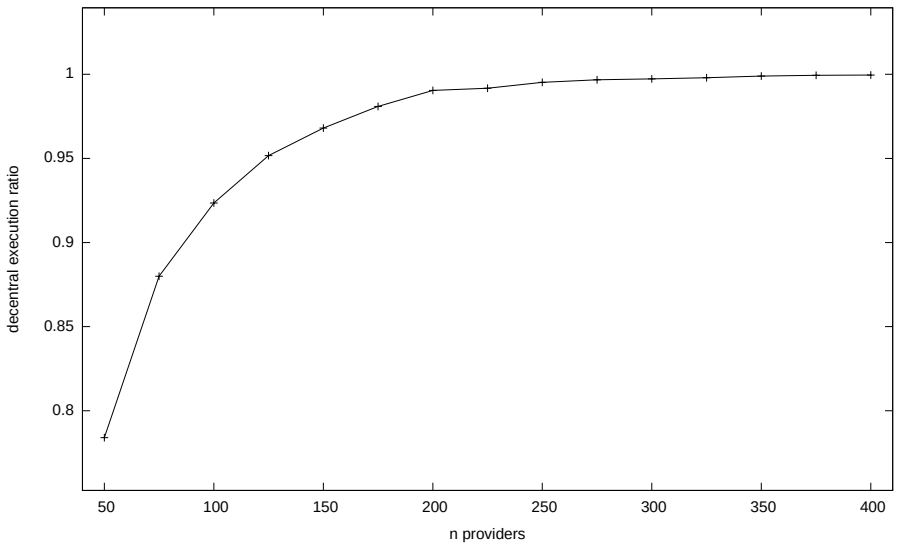

Fig. 2. Decentral execution ratio $\mathtt{ex}(n_p)$ for varying n_p from 50 to 400. Fixed: 50 Services, 0.1 density.

receiving payments from the service providers which will increase their costs. Therefore we conclude that with increasing size of the service network the prices get more competitive when compared to a central registry scenario.

Welfare Decrease (Question 4) The participants in a service network are self-interested. The proposed algorithms aim at keeping prices low in order to stay competitive while ensuring that a given margin is earned. Our main concern is how the welfare of a decentral formed service value network is compared to a centrally planed cooperation. We observed in the simulation that the welfare decrease behaves similar to the price increase (see Figure 3). However the decrease is always very small (below 0.6%) even for small networks.

We showed that the algorithm is efficient and delivers results that are competitive to a central approach with a registry that operates for free, which is optimal but unrealistic. Although customers have to pay slightly higher prices, practically all requests can be fulfilled without a significant welfare decrease. We also observed that a larger service network is better both for the customer and the overall welfare.

We assumed that the cooperations in a service network are equally distributed. A lot of real world networks including the internet have links that are power-law distributed [8]. This means that there are a few nodes with a great number of links and many nodes with only small numbers of links. Such a link distribution occurs when it is more probable that new links are created with already heavily linked nodes. This may also be a realistic assumption for service networks. Customers may prefer sending their

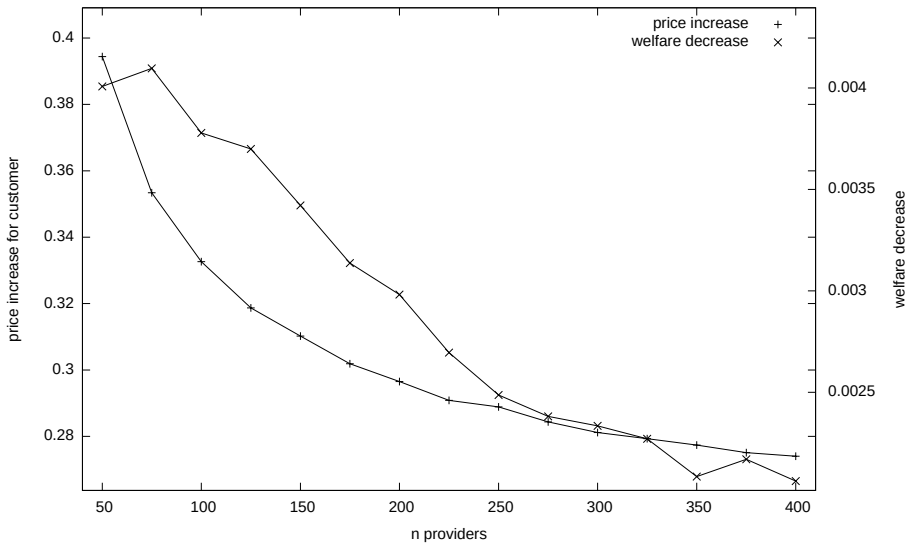

Fig. 3. Price increase $\texttt{price\_inc}(n_p)$ for customer and welfare decrease $\texttt{wf\_dec}(n_p)$ for varying n_p from 50 to 400. Fixed: 50 Services. 0.1 density.

requests to providers with a large partner network and new providers possibly want to establish partnerships with large providers in hope of many requests. In future work we aim at evaluating our algorithms on power-law networks both with simulations and analytics.

5 System Architecture

In order to realize the algorithms introduced above, we describe a platform aimed at enabling the agile formation and transformation of service value networks. The platform is an open distributed system based on service-oriented architecture principles. At its core is a service middleware, which provides a set of community services and SVN management services, including implementations of the algorithms introduced. Besides providing means for community management and task allocation the platform aims at increasing the connectivity and the size of the service network. As shown by the simulations in Section 4, this leads to a higher welfare in the network.

In the architecture, a service network is represented as set of digital communities. Each community consists of members (people, organizations) who know each other and provide services to or consume services from their neighbors by means of the platform. A service provider or customer may belong to one or several communities. Members of a community use a Web client to work with each other; the web client includes functions such as browsing and searching service offerings by other members of the community, offering new services, testing and invoking services, tagging and rating services. In

addition, the platform provides monitoring and management services that can be used to implement a governance model for the SVN [9].

For encoding communication primitives such as service offers and requests we currently use a central platform ontology as described in [10]. The way atomic services can be composed to complex services is explicitly modeled in the ontology – i.e. we currently do not apply any automated planing algorithms for composing complex services but all possible compositions are explicitly modeled in the ontology. This approach is illustrated by the following example.[3] According to the service specification outlined in Section 2, Equations 1 and 2 define the set of `AtomicServices` S_a and the set of `ComplexServices` S_c as subclasses of the set of `Services` S. Equations 3-5 represent a very simple example how the complex service `CloudService` can be modeled based on the two atomic services `StorageService` and `ComputationService`.

$$\texttt{AtomicService} \sqsubseteq \texttt{Service} \tag{1}$$

$$\texttt{ComplexService} \sqsubseteq \texttt{Service} \sqcap \exists \texttt{composedOf.Service} \tag{2}$$

$$\texttt{StorageService} \sqsubseteq \texttt{AtomicService} \sqcap \exists \texttt{StorageCapacity} \tag{3}$$

$$\texttt{ComputationService} \sqsubseteq \texttt{AtomicService} \tag{4}$$

$$\texttt{CloudService} \sqsubseteq \texttt{ComplexService} \sqcap \exists \texttt{composedOf.StorageService} \sqcap \exists \texttt{composedOf.ComputationService} \tag{5}$$

Based on such a formalization each service provider is able to determine whether a service s_p provided by himself or by one of his neighbors is suitable for a certain service request s_r. In technical terms this is realized by verifying if the requested service s_r is logically subsumed by the provided service s_p (or one of the services s_p is composed of), i.e. for a match $s_r \sqsubseteq s_p$ has to hold. Calculating the subsumption hierarchy between a set of description logic concepts is a standard reasoning task which is provided by all off-the-shelf description logic reasoners.

The formation and transformation of SVNs is driven by the availability of new service offerings that introduce a competitive advantage. For this purpose, either existing community members can add new service offerings (provided by themselves, or by a third-party), or new service providers join the service network. For this purpose, a set of membership services exist, such as joining by invitation, or joining by application. The platform can accommodate a variety of community membership models, as different membership services may be available [12].

We are currently exploring extensions to the platform in line with the architecture described in [13], in order to introduce a model for the composition of services into mashups, further allowing the sharing of code and data within a community to develop new services.

[3] The example is formalized in OWL abstract syntax as introduced in [11].

6 Related Work

Traditionally, coordination between Web service providers and customers is done by setting up central service repositories, which can be used to retrieve relevant services according to a customer's service request. Most prominent in this context has been the Universal Description, Discovery and Integration (UDDI) registry [14] which provides a data model for describing services offers and a corresponding discovery mechanism. However, UDDI registries have not been accepted by industry and most publicly available UDDI registries have been shut down by 2006.

Several decentralized service discovery mechanisms have been presented in literature. Most of them distribute service descriptions over several peers and use an indexing mechanism to efficiently route the queries through the P2P network (e.g. [15–17]). However, building and maintaining such indexes contradicts our assumption that each peer in the network has only knowledge about his direct neighbors and usually peers are not willing to share this knowledge since it can be an important business asset.

Related problems which are extensively discussed in distributed computer systems literature are (decentralized) task/job allocation and scheduling problems. There is a wide range of approaches which differ in their underlying assumptions, such as cooperative vs. non-cooperative agents, agents with complete vs. incomplete information (e.g. about costs), centralized vs. decentralized planer, etc. (e.g. see [18, 19]). A major problem which is specific to our scenario and is thus neither addressed by current task allocation nor task scheduling algorithms is the fact that peers are arranged in complex networks and each peer does not want to reveal his business contacts to other peers – i.e. no peer in the network has complete knowledge about the network structure.

There are a few approaches that explicitly consider a network of providers [20] - especially in the area of supply chain management [21, 22]. However, the algorithms assume peers with different capacity (i.e. available resources) but the same homogeneous functionality (such as computing power). Since in our network the services are highly specialized and therefore provide different functionality, these algorithms are not directly applicable. In order to determine the suitability of a peer for a certain task our approach features an expressive task ontology, which is required to overcome the heterogeneity of the Web environment.

7 Conclusion

In this paper we considered the problem of decentralized service value network formation. We provided an algorithm that distributes a service request over a network of self-interested, non-cooperative service providers and thereby creates an efficient service value network. The algorithm is novel compared to existing approaches as peers in the network do not have to provide any information about their business network to their customers. Thereby, new business models for service intermediaries are enabled, whose only business asset is a strong partner network. Up to now such business models which purely rely on social networking and contacts have been restricted to the offline world. We showed by means of a simulation that the algorithm is tractable for reasonable sized scenarios as the number of required notifications in a network with n

providers is $O(n^3)$. In addition, the results show that the algorithm performs quite well in terms of welfare descrease and price increase compared to a scenario with central repository. In fact, the total loss in welfare is only between 0.2% and 0.6%.

There are several directions in which we plan to extend this work. First, we plan to replace the current uniform distribution used to create the service network with a power-law distribution which seems to be a more realistic assumption for social networks as well as Web environments [8]. We plan an analytical and experimental evaluation how this change impacts the performance of our algorithms. Second, we plan to extend the algorithms for quality of service aspects. Modeling the trade-off between quality and price requires the introduction of a multi-attribute price and value function for providers and customers, respectively. In addition, functions have to be defined that allow us to determine the overall quality of a complex service by aggregating the quality of the atomic services. Third, we plan to assess whether introducing a market mechanism such as a path auction might further increase the efficiency of the service allocation.

References

1. Papazoglou, M.: Web Services: Principles and Technologies. Prentice Hall (2007)
2. Tapscott, D., Lowy, A., Ticoll, D.: Digital Capital: Harnessing the Power of Business Webs. Harvard Business School Press (2000)
3. Hagel III, J.: Spider versus Spider. The McKinsey Quarterly (1) (1996) 4–5
4. Zerdick, A., Picot, A., Schrape, K., Artope, A., Goldhammer, K., Lange, U.T., Vierkant, E., Lopez-Escobar, E., Silvertone, R.: E-economics. Strategies for the Digital Marketplace. Springer (2000)
5. F., S.: Formation and Early Growth of Business Webs: Modular Product Systems in Network Markets. Physica-Verlag Heidelberg (2004)
6. Hedrick, C.: Routing Information Protocol. RFC 1058 (Historic) (June 1988) Updated by RFCs 1388, 1723.
7. Moy, J.: OSPF Version 2. RFC 2328 (Standard) (April 1998)
8. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. In: SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication, Cambridge, Massachusetts, United States (1999) 251–262
9. Desai, N., Mazzoleni, P., Tai, S.: Service clubs: A structuring mechanism for service-oriented business ecosystems. In: Proceedings of the Inaugural Conference on Digital Ecosystems and Technologies (DEST 2007), Cairns, Australia, IEEE (2007)
10. Lamparter, S., Ankolekar, A., Oberle, D., Studer, R., Weinhardt, C.: Semantic specification and evaluation of bids in web-based markets. Electronic Commerce Research and Applications (ECRA) (2008) . In Press.
11. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. In Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: Description Logic Handbook, Cambridge University Press (2003)
12. Tai, S., Desai, N., Mazzoleni, P.: Service communities: applications and middleware. In Wohlstadter, E., ed.: Proceedings of the 6th International Workshop on Software Engineering and Middleware, SEM 2006, Portland, Oregon, USA, November 10, 2006, ACM (2006) 17–22

13. Maximilien, E.M., Ranabahu, A., Tai, S.: Swashup: situational web applications mashups. In: Companion to the 22nd ACM SIGPLAN conference on Object oriented programming systems and applications companion (OOPSLA'07), Montreal, Quebec, Canada, ACM (2007) 797–798
14. OASIS Technical Committee: OASIS UDDI Specification. http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm (2004)
15. Schmidt, C., Parashar, M.: A peer-to-peer approach to web service discovery. World Wide Web Journal **7**(2) (2004)
16. Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., Miller, J.: METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. Inf. Technol. and Management **6**(1) (2005) 17–39
17. Vu, L.H., Hauswirth, M., Aberer, K.: Towards P2P-based Semantic Web Service Discovery with QoS Support. In et al., C.B., ed.: Workshop on Business Processes and Services (BPS), in conjunction with the Third International Conference on Business Process Management, Nancy, France, September 6-7, 2005. Lecture Notes in Computer Science (LNCS) (2006) 18–31
18. Feigenbaum, J., Shenker, S.: Distributed algorithmic mechanism design: Recent results and future directions. In: Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Atlanta, Georgia, USA (2002) 1–13
19. Manisterski, E., David, E., Kraus, S., Jennings, N.R.: Forming efficient agent groups for completing complex tasks. In: 5th Int. Conf. on Autonomous Agents and Multi-Agent Systems, Hakodate, Japan (2006) 834–841
20. de Weerdt, M., Zhang, Y., Klos, T.: Distributed task allocation in social networks. In: AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, Honolulu, Hawaii (2007) 1–8
21. Easwaran, A., Pitt, J.: Supply chain formation in open, market-based multi-agent systems. International Journal of Computational Intelligence and Applications **2**(3) (2002) 349 – 363
22. Walsh, W., Wellman, M.: Decentralized supply chain formation: A market protocol and competitive equilibrium analysis (2003)