Disruption-Aware Service Composition and Recovery in Dynamic Networking Environments

Shanshan Jiang
EECS Department
Vanderbilt University
shanshan.jiang@vanderbilt.edu

Yuan Xue
EECS Department
Vanderbilt University
yuan.xue@vanderbilt.edu

Douglas Schmidt
EECS Department
Vanderbilt University
d.schmidt@vanderbilt.edu

ABSTRACT

The dynamic and heterogeneous natures of large-scale systems pose fundamental challenges to the design of service composition methods with minimum service disruptions. Improving reliability has long been a topic of extensive research in large-scale systems. Little existing work, however, has considered service deliveries spanning multiple components and taken both failure duration and frequency into account.

This paper proposes a new service composition and recovery framework designed to achieve minimum service disruptions. The framework consists of two-tiers: *service routing*, which selects the service components, and *network routing*, which finds the network path that connects these service components. Our framework is based on a novel concept: *disruption index*, which characterizes different aspects of service disruptions, including frequency and duration. We formulate the problem of minimum-disruption service composition and recovery (MDSCR) as a dynamic programming problem and give its optimal solution under the assumption of complete knowledge of future failure distribution. We then present our MDSCR heuristic, which approximates the optimal solution with one-step lookahead prediction, where service link lifetime is predicted through statistical regression. We present the preliminary performance results of our algorithm via simulation study.

Keywords

Service Disruption, Service Composition

1. INTRODUCTION

Component-based service development focuses on building large software systems by integrating newly-developed and/or previously-existing service components [3]. At the heart of this approach is the assumption that certain parts of software systems appear/reappear with sufficient regularity that common parts (*i.e.*, the *components*) can be used as the basis for assembling a large-scale system. The flexibility of component-based software can help reduce development costs, enable fast system assembling, and reduce the maintenance burden for large-scale systems [11].

When since large-scale systems are deployed over highly dynamic wireline and wireless networks, they are vulnerable to a wide spectrum of disturbances, including failures of their hosting platforms and network infrastructures, persistent and transient traffic congestions, and/or user mobility (if wireless network is used). Seamlessly integrating service components into a synthetical service with a high assurance of resilience over dynamic networking environments is therefore extremely hard due to the following challenges:

• Large-scale systems involve *heterogeneous* applications and users with different *subjective* evaluation towards system reliability and availability. Existing research focuses primarily on low-level system reliability metrics, such as normalized reliability of composed service graph [5, 10], and neglects higher-level user-perceived performance. New component composition and recovery strategies are therefore needed to support these different user groups and reflect subjective human elements.

• The *highly dynamic and unpredictable* behavior of large-scale systems prevents the application of static reliability analysis. Existing research [10] on reliable component deployment assumes a static network setting where network topology, node and link reliability are fixed and known *a priori*. Since these assumptions are unrealistic for large-scale systems, new reliability and availability analytical models are needed to capture the traditional concepts of instantaneous robustness as well as the time-sequenced concepts of robustness that arise in dynamic large-scale systems.

To address the challenges described above, we posit the need to enhance the foundations of large-scale system reliability analysis to support high-dynamic system development and deployment. In particular, research is needed on the following topics:

• Developing a service composition and recovery framework that integrates different service failure factors, *i.e.*, node failure, link failure, and QoS failure, and coordinates the service composition and recovery process at two levels, *i.e.*, service level and network level.

• Defining a metric that quantitatively characterizes the

ASE Workshop on Automating Service Quality, November 2007, Atlanta, Georgia, USA

user-perceived system availability and the impact of service disruption in large-scale systems. This metric is critical to direct and evaluate the design of service composition and recovery algorithms. It needs to fully characterize the impact of service disruption, including failure frequency and duration, and consider different types of failures that cause the disruptions (*i.e.*, network-level link failure and service-level QoS failure).

• Investigating theoretical models that account for key spatial and temporal factors that influence the disruption computation complexity of large-scale systems. Based on the defined disruption metric, these models will develop optimization-based formulations for the component composition and recovery problem under system dynamics. The goal is to maximize the user-perceived system availability and minimize the impact of service disruption. These analytical models will also provide valuable theoretical insights to practical algorithm design for large-scale systems.

The remaining of this paper is organized as follows: Section 2, presents our service composition and recovery framework for large-scale systems; Section 3 defines a novel service disruption model; Section 4 uses this model to formulate the Minimum Disruption Service Composition and Recovery problem and present its optimal and heuristic solutions for large-scale systems; Section 5 evaluates our simulation results; Section 6 presents concluding remarks.

2. A SERVICE COMPOSITION AND RECOVERY FRAMEWORK

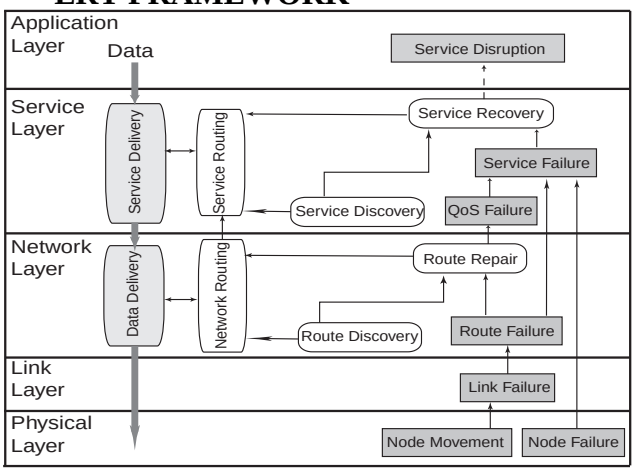

Figure 1: A Service Composition and Recovery Framework

Service composition refers to the process of finding a *service path* in the network, which is a linked sequence of components from many of their replicas. As shown in Figure 1, service composition involves the following two tightly-coupled processes:

• *Service routing*, which selects the service components (out of many replicas) for the service path. It relies on service component discovery [9, 7, 8] to find the candidate service components, then selects the appropriate ones to compose a service path. Formally, a service routing scheme is represented as $\pi_{\mathcal{S}} = (s_1[n_1], s_2[n_2], ..., s_r[n_r])$, where n_k is the hosting node for the selected service component s_k.

• *Network routing*, which finds the network path that connects the selected service components. Formally, the network routing scheme could be represented as a set of paths $\pi_{\mathcal{N}} = \{\mathcal{P}_{(n_k,n_{k+1})}, k = 1, ..., r-1\}$ where $\mathcal{P}_{(n_k,n_{k+1})}$ represents the network path that supports the service link $(s_k[n_k] \rightarrow s_{k+1}[n_{k+1}])$.

These two processes closely interact with each other. The component selection in service routing determines the source and destination nodes in network routing. Likewise, the path quality in network routing also affects the selection of service components in service routing. Collectively, a service composition scheme is represented as $\pi = (\pi_{\mathcal{S}}, \pi_{\mathcal{N}})$.

There are three major causes of service failures: (1) node failure (*e.g.*, the failure of a service component hosting platform) may occur due to hardware faults, node departure or abortion in self-organized systems (such as peer-to-peer systems), (2) link failure (*e.g.* in the network infrastructure) may be caused by node mobility in mobile wireless networks, or by misconfiguration and/or security attacks in Internet, and (3) QoS failure (*e.g.*, a violation of the service QoS, such as delay and jitter) may be caused by dynamic traffic demand and network congestions.

To sustain service delivery, the failed service path must be repaired. This repair process essentially *recomposes the service path* and is called *service recovery*. Service recovery is triggered by service failure detection at either the network level or service-level.

Similar to service composition, service recovery process also involves two processes, namely, *network-level recovery*, which repairs the data path between two components, and *service-level recovery*, which replaces one or more service components. Network-level recovery usually depends on the specific routing protocol in use and the route repair mechanism built within this routing protocol. Service-level recovery involves discovery of new components and establishment of a new service path.

Service recovery differs from service composition since it must consider not only the quality of the recomposed (repaired) path, but also the service path previously in use (the one that just failed). To reduce the repair overhead and recovery duration, we intuitively prefer a service path that could maximally reuse the current nodes/components. Using such a service recovery strategy, however, the new service path may have a poor QoS and/or may fail soon in the future.

3. SERVICE DISRUPTION MODEL

During the service failure and recovery processes, the service is unavailable to end users, thereby causing service disruption. To analytically investigate service composition and recovery strategies that could provide the most smooth and reliable service delivery, we first need to quantitatively characterize the impact of service disruption.

Consider a service $\mathcal{S}$ that starts at time instance 0 and ends

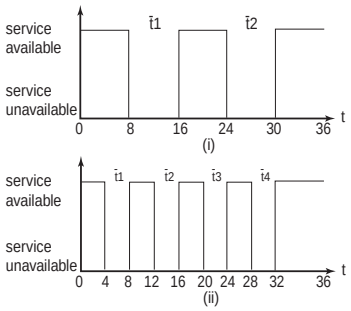

Figure 2: Example Service Disruption Processes

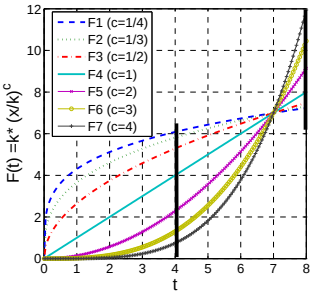

Figure 3: Example Disruption Penalty Functions (k=7 is the intersection point of all the lines)

at T. Figure 2 shows an example of two service disruption processes. Let $\bar{t}_1, \bar{t}_2, ..., \bar{t}_q$ be the sequence of disruption durations. A classical way to model service disruption is *service availability* [2], which is defined as the fraction of service available time during the service lifetime: $A = \frac{T - \sum_{i=1}^{q} (\bar{t}_i)}{T}$. Using availability as the metric to characterize the impact of service disruption, however, we face the following two problems:

• *Service availability cannot characterize the impact of service failure frequency.* For example, in Figure 2, scenario (i) and (ii) have the same service availability ($\frac{24}{36}$). The user-perceived disruption could be different, however, since scenario (ii) has a higher service failure frequency but smaller disruption durations. To precisely model the effect of service disruption, therefore, we need a new metric that characterizes both failure frequency and failure durations.

• *Service availability is hard to compute a priori.* The calculation of service availability is based on the calculation of disruption durations, which include the service failure time and recovery time. Such durations are determined by many factors, such as network topology, routing protocol, and system conditions, which are dynamic and hard to be incorporated into service composition and recovery decisions. To establish a theoretical framework that provides realistic insight to implementation of service composition and recovery strategy, we need a metric that is stable, easily computed, and can provide a good estimation of disruption durations.

To address the first problem regarding the impact of service failure frequency, we associate a *disruption penalty function* $F(\bar{t})$ defined over the disruption duration $\bar{t}$ with an end user. The shape of $F(\bar{t})$ reflects its relative sensitivity to disruption duration and frequency. Figure 3 shows three basic types of failure penalty functions (*i.e.*, convex, linear, concave). We further define *disruption index* D as a metric to characterize the impact of service disruption:

$$D = \frac{\sum_{i=1}^{q} F(\bar{t}_i)}{T} \quad (1)$$

Table 1: Disruption Indices Under Different Penalty Functions

Fi	Fi(4)	Fi(8)	$D_{Proc(i)}$	$D_{Proc(ii)}$
F1 (convex)	6.0861	7.2376	0.4021	0.6762
F2 (convex)	5.8088	7.3186	0.4066	0.6454
F3 (convex)	5.2915	7.4833	0.4157	0.5879
F4 (linear)	4.0000	8.0000	0.4444	0.4444
F5 (concave)	2.2857	9.1429	0.5079	0.2540
F6 (concave)	1.3061	10.4490	0.5805	0.1451
F7 (concave)	0.7464	11.9417	0.6634	0.0829

To show how the disruption index D characterizes the user-perceived disruption effect and integrates both disruption duration and failure frequency, we calculate the disruption indices for the two service disruption processes in Figure 2 using the different failure penalty functions shown in Figure 3. The results are summarized in Table 1.

Table 1 shows that if $F(\bar{t})$ is a convex function then disruption process (ii) has a higher disruption index than process (i), *i.e.*, its end user is more sensitive to failure frequency. When $F(\bar{t})$ is a concave function, disruption process (i) has a higher disruption index than process (ii), *i.e.*, its end user is more impatient to disruptions with long duration. For a linear disruption penalty function the user is neutral, and the disruption index depends on the service availability.

To address the second problem regarding computing service availability, we present simple and stable estimations of disruption durations for network-level recovery and service-level recovery separately.

3.0.1 Estimation for network-level recovery

A network-level recovery is triggered by node/link failures and network-level QoS violations. For network-level recovery, the service components remain the same, *i.e.*, we only need to re-route the network path that connects them. Typical network-level recovery processes in repairing a network path involve discovering an alterative route to replace the broken link/path and restarting the data delivery. Here we use the number of link substitutions in the repair as a simple estimate for the disruption duration introduced by network-level recovery.

Using the number of link substitutions as an estimate for disruption duration introduced by network-level recovery is consistent with typical network repair operations. For example, there are usually two repair mechanisms in wireless ad hoc routing: *local repair* and *global repair*. For local repair, when a link fails, one of its end nodes will try to find an alternative path in the vicinity to replace this link. Local repair therefore involves fewer link substitutions and less recovery time. For global repair, the source node initiates a new route discovery, which takes more time than local repair and involves more link substitutions.[1]

3.0.2 Estimation for service-level recovery

[1]For simple estimation, we do not consider the impact of route caches here.

A service-level recovery can be triggered by failed network-level recovery and/or service nodes failures. It involves three operations: (1) finding the appropriate substitution components, (2) starting the new components and restoring the service states, and (3) finding a network path that supports the connectivity between the new components. Service-level recovery thus takes much more time than network-level recovery. Similar to network-level recovery, the duration of service-level recovery depends largely on the searching/-replacing scope of the service components. We can therefore use the number of substituted components to estimate its recovery duration.

Based on the recovery duration estimation, we now proceed to refine our definition of the disruption index. Consider a service $\mathcal{S}$ that starts at time instance 0 and ends at T. Let $\pi(t_1), \pi(t_2), ..., \pi(t_l)$ be the sequence of service composition schemes used during the service lifetime. The disruption duration $\bar{t}_k$ from service composition $\pi(t_k)$ to $\pi(t_{k+1})$ is estimated as

$$\bar{t}_k = \beta \times N_{\pi(t_k)\rightarrow\pi(t_{k+1})} \quad (2)$$

$$= \beta \times (N^{\mathcal{N}}_{\pi(t_k)\rightarrow\pi(t_{k+1})} + \alpha N^{\mathcal{S}}_{\pi(t_k)\rightarrow\pi(t_{k+1})}) \quad (3)$$

where $N^{\mathcal{N}}_{\pi(t_k)\rightarrow\pi(t_{k+1})}$ and $N^{\mathcal{S}}_{\pi(t_k)\rightarrow\pi(t_{k+1})}$ denote the number of substituted links in network-level recovery (if any) and the number of substituted components in service-level recovery (if any) incurred by the service composition transition from $\pi(t_k)$ to $\pi(t_{k+1})$. β is the parameter that converts the number of substitutions to disruption time. $\alpha \gg 1$, denotes the relative weight between service component substitution and link substitution on disruption duration. Based on the discussions above, the disruption index D could be estimated as

$$\tilde{D} = \frac{\sum_{k=1}^{l-1} F(\beta \times N_{\pi(t_k)\rightarrow\pi(t_{k+1})})}{T} \quad (4)$$

4. MINIMUM DISRUPTION SERVICE COMPOSITION AND RECOVERY PROBLEM

A fundamental research challenge for service recovery is *how to best tradeoff the time and overhead involved in service recovery and the sustainability of composed service path so that end users will perceive minimum disruptions to the service during its lifetime*. To address this challenge, a theoretical framework is needed to analytically study the problem of service composition and recovery strategies to achieve minimum service disruptions. This section establishes such an optimization-based theoretical framework based on dynamic programming.

4.1 Problem Formulation

We first formulate the *minimum disruptive service composition and recovery* (MDSCR) problem. First, we define a service composition and recovery policy as a sequence of service composition schemes: $\Pi = (\pi(t_1), \pi(t_2), ..., \pi(t_l))$, where $0 = t_1 < ... < t_l \leq T$. $\pi(t_k)$ gives the service composition during time $[t_k, t_{k+1})$. Note that Π gives initial service composition $\pi(t_1)$ and all the service recovery schemes $\pi(t_k) \rightarrow \pi(t_{k+1})$, $k = 1, ..., l-1$.

We model a network at time t as $\mathcal{G}(t) = (\mathcal{N}(t), \mathcal{L}(t)\}$, where $\mathcal{N}(t)$ and $\mathcal{L}(t)$ represents the set of nodes and links at time t, respectively. We denote the set of all feasible service composition policies over a network $\mathcal{G}(t)$ as $\Phi(\mathcal{G})$. The goal of the MDSCR algorithm is to find the best policy $\Pi \in \Phi(\mathcal{G})$ that is feasible for $\mathcal{G}(t)$, so that $\tilde{D}(\Pi)$ is minimized. Formally,

$$\textbf{MDSCR}: \quad \textbf{minimize} \quad \tilde{D}(\Pi) \quad (5)$$

$$\Pi \in \Phi(\mathcal{G}) \quad (6)$$

4.2 Optimal Solution

If $\mathcal{G}(t)$ is given, **MDSCR** is essentially a dynamic programming problem. Let $\mathcal{J}(\pi(t_w))$ be the minimum disruption index for the service disruption experienced by the service from time instance t_w when composition scheme $\pi(t_w)$ is used, *i.e.*,:

$$\mathcal{J}(\pi(t_w)) = \min_{\Pi\in\Phi(\mathcal{G})} \frac{1}{T}\sum_{k=w}^{l-1} F(\beta \times N_{\pi(t_k)\rightarrow\pi(t_{k+1})}) \quad (7)$$

Based on dynamic programming, we have

$$\mathcal{J}(\pi(t_w)) = \min_{\pi(t_{w+1})} \{\frac{1}{T}F(\beta \times N_{\pi(t_w)\rightarrow\pi(t_{w+1})}) + \mathcal{J}(\pi(t_{w+1}))\} \quad (8)$$

Based on the analytical properties and the corresponding proofs of the optimal MDSCR solution, which are given in [6] due to space constraints, we can reduce the complexity of MDSCR problem by decomposing it into two sub-problems: (1) the service-level MDSCR problem and (2) the network-level MDSCR problem. We show that for an optimal solution, the service-level recovery is invoked if and only if the network-level recovery can not repair one of the service link in use (*i.e.*, there is no feasible network path connecting these two service components). Here we focus our discussion on service-level MDSCR and rely partially on the existing network routing protocols for network-level MDSCR. Our objective is to minimize the service-level disruption index via service routing. At time t^s_{w+1} with service routing scheme $\pi_{\mathcal{S}}(t^s_w)$, the service recovery scheme that changes the service path from $\pi_{\mathcal{S}}(t^s_w)$ to $\pi_{\mathcal{S}}(t^s_{w+1})$ is given by solving the following equation:

$$\mathcal{J}(\pi_{\mathcal{S}}(t^s_w)) \quad (9)$$

$$= \min_{\pi_{\mathcal{S}}(t^s_{w+1})} \{\frac{1}{T}F(\beta\alpha N^{\mathcal{S}}_{\pi_{\mathcal{S}}(t^s_w)\rightarrow\pi_{\mathcal{S}}(t^s_{w+1})}) + \mathcal{J}(\pi_{\mathcal{S}}(t^s_{w+1}))\}$$

The equation shown above could be used to give the optimal MDSCR solution via standard dynamic programming techniques [1]. In particular, solving $\mathcal{J}(\pi(t_1))$ gives the optimal initial service composition $\pi(t_1)$. At time t_w with service

composition scheme $\pi(t_w)$, solving Eq. (8) gives the optimal service recovery scheme (minimum disruption service recovery) that changes the service composition from $\pi(t_w)$ to $\pi(t_{w+1})$.

4.3 Heuristic Solution Based on One-step Look-ahead Approximation

To derive an optimal solution, the complete knowledge of the current and future network topologies, service component deployments, and failures is required. In mobile wireless networks where mobility-caused link failure is the dominant factor, this implies that the mobility plan is known *a priori*. In most practical scenarios, however, the failure time is unknown, which means that we can not calculate the precise disruption indices between each composition and recovery steps in Eq. (7). Finding the optimal solution to MDSCR problem is infeasible, therefore, since the service recovery decision at t^s_{w+1} requires the knowledge of network topology after this time to calculate the future disruption index $\mathcal{J}(\pi_{\mathcal{S}}(t^s_{w+1}))$. To address this problem, we present a one-step look-ahead approximation method where future disruption index is estimated in the time period until its first service failure.

Formally, let $L_{n_k \rightarrow n_{k+1}}$ be the expected lifetime for the service link $(s_k[n_k] \rightarrow s_{k+1}[n_{k+1}])$. The service routing scheme at time t^s_{w+1} is $\pi_{\mathcal{S}}(t^s_{w+1}) = (s_1[n_1], s_2[n_2], ..., s_r[n_r])$. Its failure rate is estimated as $\gamma_{\pi_{\mathcal{S}}(t^s_{w+1})} = \sum_{k=1}^{r-1} \frac{1}{L_{n_k \rightarrow n_{k+1}}}$. $\mathcal{J}(\pi_{\mathcal{S}}(t^s_{w+1}))$ is estimated as

$$\hat{\mathcal{J}}(\pi_{\mathcal{S}}(t^s_{w+1})) = F(\beta\alpha \times E[N^{\mathcal{S}}]) \times \gamma_{\pi_{\mathcal{S}}(t^s_{w+1})} \quad (10)$$

The initial service composition strategy is to find $\pi_{\mathcal{S}}(t^s_1)$ to minimize $\gamma_{\pi_{\mathcal{S}}(t^s_1)}$. The service-level recovery strategy involves finding a service routing scheme $\pi_{\mathcal{S}}(t^s_{w+1})$ to minimize

$$\frac{1}{T} F(\beta\alpha N^{\mathcal{S}}_{\pi_{\mathcal{S}}(t^s_w) \rightarrow \pi_{\mathcal{S}}(t^s_{w+1})}) + F(\beta\alpha E[N^{\mathcal{S}}]) \gamma_{\pi_{\mathcal{S}}(t^s_{w+1})} \quad (11)$$

Eq. (11) formally characterizes the trade-off between the recovery duration (first term) and the sustainability of the newly composed path (second term) faced by service recovery.

4.4 Lifetime Prediction Based on Statistical Regression

Now the problem left in deriving a practical MDSCR solution is to estimate the service link lifetime, which depends on the lifetimes of its components and their hosting nodes, the lifetime of the underlying network paths, and traffic dynamics. This problem is hard due to the highly dynamic node/link failures, the inter-dependent network link and path failures, and diverse service QoS requirements. To address this challenge, we use a statistics-based method to estimate service link lifetime.

In the existing work, the node lifetime distribution can be derived based on traces collected from real-world systems. For example, in [4] the authors collect traces from five large-scale peer-to-peer systems and estimate the node lifetime distribution using standard Pareto distribution with different parameters.

In what follows, we present a network path lifetime prediction method for mobile wireless network based on linear regression. In particular, we estimate the lifetime of a network path $L_{n \rightarrow n'}$ based on the predicted distance between two components $\tilde{d}_{n \rightarrow n'}(t + \Delta t)$, which is calculated based on the current locations of the hosting nodes, their velocities and the prediction time Δt. The lifetime of a network path is computed via linear regression shown as follows.

$$L_{n \rightarrow n'} = K \times \tilde{d}_{n \rightarrow n'}(t + \Delta t) + B \quad (12)$$

In the simulation study, we derive the corresponding parameters for linear regression for different network setups, and pick the prediction time Δt with the smallest standard error.

5. SIMULATION STUDY

We now study the performance of our algorithm via simulation. This section focuses on the service failures caused by node mobility in a wireless ad hoc network. In the simulated network, 50 nodes are randomly deployed over a $2,000 \times 1,000 m^2$ region. Node mobility follows the random waypoint model with certain *maximum speed* (default value is $10 m/s$) and certain *pause time* (default value is $10 s$).

Each simulation runs for $10^5 s$. The simulated service is composed from 4 components; each component has 5 replicas by default. In the simulation, the prediction time is adjusted based on network setup to achieve the smallest prediction error. The network routing protocol is simulated using AODV in ns-2. The service is sending constant bit rate (CBR) traffic at $1 pk/sec$.

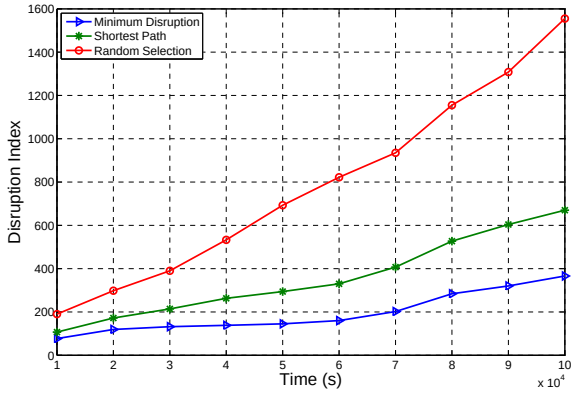

Figure 4: Disruption comparison with MDSCR, SP-SCR, and RSSCR

We compare the disruption index and the throughput of our MDSCR algorithm with those of the shortest path service composition and recovery (SPSCR) algorithm and the random selection service composition and recovery (RSSCR) algorithm over the same network scenario (*i.e.*, each node

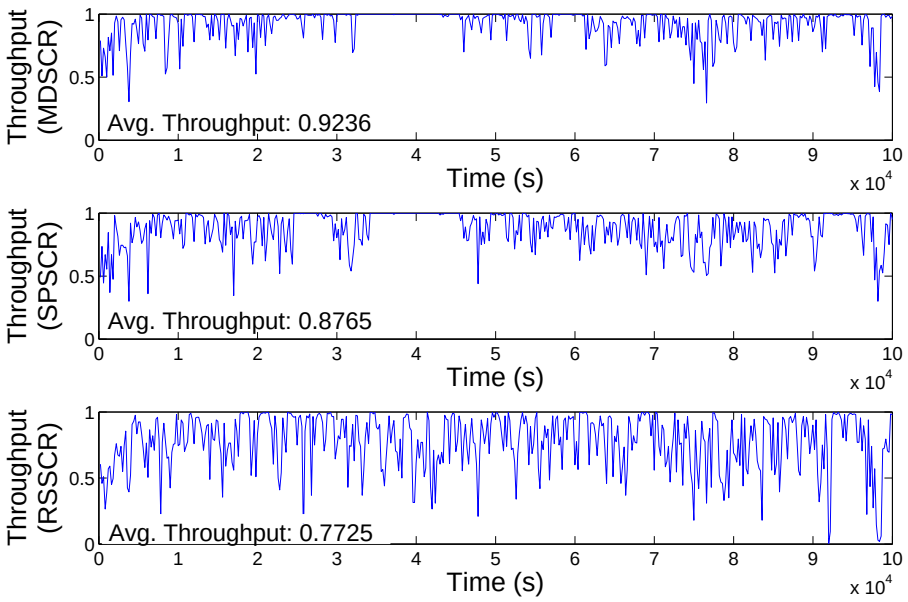

Figure 5: Throughput comparison with MDSCR, SPSCR, and RSSCR

in three runs of the simulation follows the same trajectory). The results are shown in Figure 4 and Figure 5.

The results show that the MDSCR algorithm incurs fewer and shorter disruptions with regard to their frequencies and durations, which leads to a relatively higher and smoother throughput and a smaller disruption index.

6. CONCLUDING REMARKS

This paper systematically investigates the service composition and recovery strategies that improve the performance of service delivery in large-scale systems under frequent link and node failures. It develops a theoretical framework for minimum disruption service composition and recovery based on dynamic programming, and presents a *minimum-disruption service composition and recovery* (MDSCR) heuristic algorithm that provides an effective service composition and recovery solution. Our simulation study over large-scale mobile ad hoc wireless networks shows that the MDSCR algorithm can provide much less disruption to end users than traditional methods, such as shortest path routing and service composition.

7. REFERENCES

[1] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2000.

[2] L. Buttyan and J. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *Proc. of the 1st ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2000.

[3] G. Deng, J. Balasubramanian, W. Otte, D. C. Schmidt, and A. Gokhale. DAnCE: A QoS-enabled Component Deployment and Conguration Engine. In *Proc. of the Working Conference on Component Deployment*, 2005.

[4] P. B. Godfrey, S. Shenker, and I. Stoica. Minimizing Churn in Distributed Systems. In *Proc. ACM SIGCOMM*, 2006.

[5] X. Gu and K. Nahrstedt. Dynamic QoS-Aware Multimedia Service Configuration in Ubiquitous Computing Environments. In *Proc. of IEEE ICDCS*, 2002.

[6] S. Jiang, Y. Xue, and D. Schmidt. Minimum Disruption Service Composition and Recovery in Mobile Ad hoc Networks. In *Vanderbilt University technical report #ISIS-06-711*, 2006.

[7] R. Koodli and C. Perkins. Service Discovery in On-Demand Ad Hoc Networks. In *Internet Draft*, 2002.

[8] U. Kozat and L. Tassiulas. Network layer support for service discovery in mobile ad hoc networks. In *Proc. of IEEE INFOCOM*, 2003.

[9] U. C. Kozat and L. Tassiulas. Service discovery in mobile ad hoc networks: An overall perspective on architectural choices and network layer support issues. *Ad Hoc Networks*, 2(1), 2004.

[10] M. Mikic-Rakic, S. Malek, and N. Medvidovic. Improving Availability in Large, Distributed Component-Based Systems via Redeployment. In *Proc. of International Working Conference on Component Deployment*, 2005.

[11] V. Subramonian, G. Deng, C. Gill, J. Balasubramanian, L. Shen, W. Otte, D. C. Schmidt, A. Gokhale, and N. Wang. The Design and Performance of Component Middleware for QoS-enabled Deployment and Conguration of DRE Systems. *Journal of Systems and Software: special Issue Component-Based Software Engineering of Trustworthy Embedded Systems*, 2006.