Silas et al. Human-centric Computing and Information Sciences 2012, 2:5
http://www.hcis-journal.com/content/2/1/5

Human-centric Computing and Information Sciences
a SpringerOpen Journal

RESEARCH Open Access

A novel fault tolerant service selection framework for pervasive computing

Salaja Silas[1*], Kirubakaran Ezra[2] and Elijah Blessing Rajsingh[1]

* Correspondence:
blessingsalaja@gmail.com
[1]Karunya University, Coimbatore, TamilNadu, India 641114
Full list of author information is available at the end of the article

Abstract

Background: Service selection in pervasive computing is significant as it requires identifying the best service provider based on users requirements. After identifying the best service provider, when a service is being executed, service disruptions happen due to various faults.

Methods: Though attempts are made to provide best services to the user, executing the service without service disruptions becomes an important challenge in the pervasive environment. In this paper, a novel Fault tolerant Service Selection Framework (FTSSF) has been proposed.

Results: Adequate theoretical analysis was carried out and experimental results were obtained for the proposed framework and have been compared with the existing techniques.

Conclusion: The results prove that the proposed framework is efficient and fault tolerant.

Keywords: Pervasive computing, Service selection, Multi criteria decision making problem, PROMETHEE, Monitoring, Fault handling, Fault tolerant

Introduction

Pervasive computing is an emerging area of research where the computing devices are embedded in the environments. The devices that provide services are termed as service providers. Service providers differ in terms of hardware components, operating systems and capability [1]. The pervasive environment consists of a set of service providers and a set of users. Each and every service provider provides a set of services. Pervasive environments can have many service providers providing services with similar functionality but possessing different criteria. The different criteria [2] that influence the service selection in pervasive environments are availability [3], cost [4], reliability [5,6], capability [3], mobility [7], responsiveness [8], trust [3,9] and locality [7]. Users, who require a service, will be requested to provide their preferences for the criteria. The Service Selection Framework [2,10-12] selects a service provider from a finite set of service providers based on a finite set of user preferences and the service is executed by the selected service provider. There is uncertainty in pervasive environments due to mobility, volatile network topology and light weight terminals. Therefore the web service selection models cannot be straight forwardly extended for pervasive computing.

During service execution, service disruptions can happen due to faults in the pervasive environment. Shiva Chetan et al. [13] classified the faults that occur in the pervasive system as service failures, device failures, network failures and application failures. These faults greatly interrupt the service execution and will lead to a situation where service providers are constrained to complete their service requests. During such cases, the Service Selection framework have to invoke once again the service selection methodology to identify the next best service provider and the new service execution has to be restarted. In most cases, multiple services are offered by the service provider. Therefore, during such faults, one or more services will be affected thereby giving rise to higher service delay and service recovery overhead. Though attempts are made to provide best services to the user, executing the service without service disruptions, with minimal service delay and service recovery overhead is an important challenge in the pervasive environment.

This has motivated the authors to investigate and propose a novel Fault tolerant Service Selection Framework for pervasive computing. This novel framework has been presented in this paper. The objective is to provide the best service anywhere, any time without any service disruption and with minimal service delay, minimal service recovery overhead and high success rate. The proposed framework has been designed to have mechanisms to automatically complete the execution of the disrupted service during fault.

In this paper, adequate theoretical analysis was also carried out and experimental results were obtained for the proposed framework and have been compared with the existing techniques. The experimental results prove that the proposed framework is efficient and fault tolerant. The success rate of the proposed framework is also high with minimal service recovery overhead and minimal service delay. It was also observed that the load on the service provider affects the service recovery overhead and the mobility affects the fault tolerance behavior of the system.

First, the related work has been discussed. Second, the fault tolerant service selection framework has been proposed. Third, the theoretical analysis on the proposed framework has been discussed. Fourth, the experimental results have been discussed and then concluded.

Related work

In recent years, researchers have focused on proposing fault tolerant mechanisms to provide seamless services in web service, MANET and pervasive computing. Some of the relevant fault tolerant schemes are discussed in this section.

San-Yih Hwang et al. [14] proposed to use finite state machine to model the permitted invocation sequences of web service operations. The approach applied to real industrial applications with a handful of atomic web services because it conformed to industrial standards and allowed for quick WS selection at runtime in a dynamic environment. However, when the number of atomic web services became large, construction of a composition took too much time, which made the approach impractical.

Chia-Feng Lin et al. [15] proposed Relaxable QoS based Service Selection (RQSS) that helped to composite web application development by discovering feasible web services based on functionalities and QoS criteria of user requirements. The RQSS recommended prospective service candidates to users by relaxing QoS constraints, if no suitable or available web service could exactly fulfill user requirements. The RQSS

algorithm increased the system availability and reliability to a certain extent but failed to address the performance of the algorithm during faults.

Chen-Liang Fang et al. [16] proposed a fault tolerant web service model called fault tolerant SOAP or FT-SOAP through which web services could be built with higher resilience to failure. The major contribution of FT-SOAP was to prove that web services built on a SOAP framework enjoy higher flexibility compared to those built on CORBA.

The web service selection models cannot be straight forwardly extended for pervasive computing due to its volatile network topology.

Koushanfer et al. [17] proposed heterogeneous back-up scheme that addressed the problem of embedded sensor network fault tolerance, where one type of resources was substituted with another. The heterogeneous fault tolerance techniques for sensor networks included the ones where communication and sensing were mutually backing up each other. The heterogeneous back up scheme provided a low cost, low overhead, high resilient fault tolerant technique but it have not considered about the application failure.

Weigang Wu et al. [18] proposed a permission-based message efficient mutual exclusion (MUTEX) algorithm for mobile ad hoc networks (MANETs). The proposed algorithm tolerated link or host failures, using timeout-based mechanisms and was able to handle dozes and disconnections [18] of mobile hosts. Permission based MUTEX algorithm was efficient, reliable and independent of any logical topology. The MUTEX based algorithm fails when there is a communication fault.

Shameem et al. [19] proposed Self Healing for Autonomic Pervasive Computing (SHAPC) that stored all the crucial information's including log status of the faulty device. The healing manager re-collected all the information for the device to restore to its previous condition. Information distribution process distributed the essential information among the other existing devices. This process assisted the faulty device to securely maintain all the important information's. Re-assignment process was responsible for finding an alternate device that was available and compatible with the faulty device that was required for smooth functionality. The main drawback of SHAPC is that the system requires user intervention whenever the selection of the service was based on the user's preferences. Shameem et al. [20] proposed a middleware service for pervasive advertisements to improve mobile business. The authors considered ubiquitous access, privacy and security. The middleware did not consider communication failures or network failures.

Peizhao hu et al. [21] proposed a model-based autonomic context management system for pervasive computing that dynamically configured and reconfigured its context information by gathering and preprocessing functionality, which provided fault tolerant provisioning of context information. The approach used standard based descriptions of context information sources that increased openness, interoperability and scalability of context-aware systems. The model saved energy, communication and processing resources, as sensors were attached to the context management system and activated dynamically. Peizhao hu et al. [21] discussed the fault due to sensor failures but had not considered other failures such as communication failures, service failures, application failures and network failures.

Amir Padovitz et al. [22] proposed an ECORA framework for context-oriented pervasive computing and reasoned about context under uncertainty. The framework followed an agent-oriented hybrid approach, combining centralized reasoning services with context-aware, reasoning capable mobile software agents. Amir Padovitz et al. [22] stated that the following combination was important to develop an adaptive context-aware pervasive computing system. They were (1) a unifying context model with algorithms to reason about context under uncertainty, (2) event-based communication as an awareness mechanism, and (3) the ability of components to move as an agility mechanism. It did not consider the communication failure.

Themistoklis et al. [23] proposed a Starfish based self-healing framework for pervasive computing systems that followed the Self-Managed Cell architectural paradigm. Starfish was an instantiation of an SMC for wireless sensor networks. It had an embedded policy system that allowed reconfiguration on individual nodes, remote access control to remote resources. It supported adaptation on nodes thereby allowing deployment of new strategies at run-time. Starfish based self healing framework enabled to recover only from sensor failures and did not consider other faults in pervasive computing.

Salaja et al. [2] proposed a Service Selection Framework adopting PROMETHEE methodology (SSF-P) for pervasive environments. The service authority of SSF consisted of Service Registration Unit, User Registration Unit, Service Selection Unit, Service Delivery Unit, Feedback unit and Trust Management Unit. The service registration overhead and the service selection time were very minimal, but the service recovery overhead and the service delay were high. And the framework was also not fault tolerant.

Therefore an attempt has been made to propose a Fault Tolerant Service Selection Framework for pervasive environment that would be tolerant to all types of fault with minimal service recovery overhead and service delay without compromising on service registration overhead and success rate. In the next section, the proposed novel Fault Tolerant Service Selection Framework has been discussed in detail.

A novel fault tolerant service selection (FTSS) framework

The Figure 1 shows the proposed fault tolerant service selection framework. The Framework consists of Service Registration Unit, User Registration Unit, Service Selection Unit, Service Delivery Unit, Monitoring & Fault handling unit and Trust Management Unit. Each Service provider furnishes the services it wishes to provide and registers it with the service registration unit. The service provider database consists of details of the service provider and the service database consists of details of different services that the service authority can provide. The user registration unit facilitates willing users who are interested in availing the services to register and also provides access control mechanisms for successful interactions.

The service selection unit helps in identifying the required service provider based on the users requirements. It selects the best service through PROMETHEE methodology [2] and provides to the user. PROMETHEE Method [24-28] is an outranking method used to solve multi-criteria problems, which are considered to be NP- complete [7]. PROMETHEE methodology has been implemented to select the best Service Provider based on the users' requirement [2]. This methodology provides the users to select the

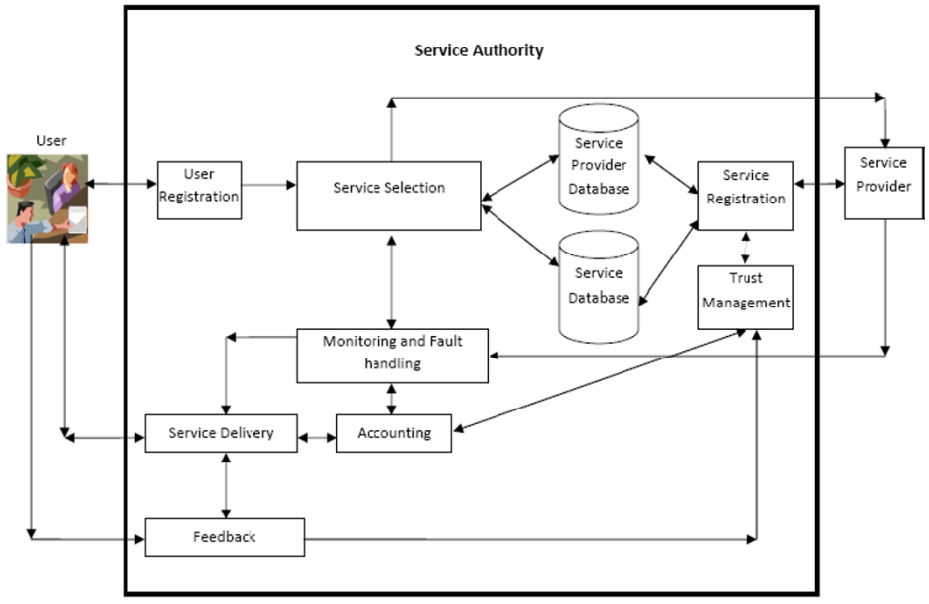

Figure 1 Proposed Fault Tolerant Service Selection Framework.

criteria of their own interest based on their requirement and also to give preferences to those criteria in terms of weights.

The method requires two inputs for processing. They are information from i) the user and ii) the Service provider. The user provides their preference for all k criteria *cr* as weights $\{w_j, j = 1,2,...k\}$ that are normalized to 1. The normalized user preferences are called as weighted user preferences. The service providers provide the information of all their services. The service providers that offer relevant services are grouped to form n_{fs} feasible service providers.

The preference function is calculated for maximization criterion as well as the minimization criterion. For maximization criterion, the preference function P_j (s_x,s_y) gives the preference of service provider s_x over service provider s_y for the observed deviations as defined below.

$$P_j(s_x, s_y) = F_j[d_j(s_x, s_y)] \forall s_x, s_y \in S \tag{1}$$

where $d_j(s_x,s_y) = cr_j(s_x)\text{-}cr_j(s_y)$ for which $0 \leq P_j(s_x,s_y) \leq 1$ where S is a set of service providers offering a particular service. For minimization criterion, the preference function is calculated by the following P_j (s_x,s_y) is defined as

$$P_j(s_x, s_y) = F_j[-d_j(s_x, s_y)] \forall s_x, s_y \in S \tag{2}$$

The Qualitative Preference function is defined as

$$P(d) = \begin{cases} 0, d \leq 0 \\ 1, d > 0 \end{cases} \tag{3}$$

The pair wise comparisons of various criteria are performed from the feasible set of service providers and the aggregated preference indices are defined. Let $s_x,s_y \in S$, where S is a set of service providers offering a particular service, then the Aggregated

Preference Indices which is the weighted summation of all the service provider is given by

$$\begin{cases} \pi(s_x, s_y) = \sum_{j=1}^{k} P_j(s_x, s_y) w_j \\ \pi(s_y, s_x) = \sum_{j=1}^{k} P_j(s_y, s_x) w_j \end{cases} \tag{4}$$

π (s_x,s_y) expresses the degree in which the service provider s_x preferred over the service provider s_y over all the criteria and π (s_y,s_x) expresses the degree in which the service provider s_y is preferred over service provider s_x. The net outranking flow for each service provider s_a is the net difference between the positive and negative outranking flows and is obtained using

$$\phi(s_a) = \frac{1}{n_{fs} - 1} \left(\sum_{x \in s} \pi(s_a, s_x) - \sum_{x \in s} \pi(s_x, s_a) \right) \tag{5}$$

where n_{fs} is the number of feasible service providers. The service selection unit selects the service provider with the highest net outranking flow as the best service provider.

The accounting unit computes the total cost of the service and the service delivery unit delivers the services to the user. The Feedback unit collects the user's satisfaction for the delivered service and provides it to the trust management unit. The feedback unit also calculates the service delay and provides the service satisfaction to the trust management unit. The trust management unit facilitates computation of trust value for every registered service provider and is stored in the service trust matrix.

Monitoring & Fault handling unit is the heart of the framework that provides fault tolerance behavior. It monitors the services of the service provider and initiates corrective measures whenever there is a fault thereby providing fault tolerance to the framework. It keeps track of all the services that are allotted to the registered users' and monitors whether the execution of the allotted service has been completed successfully. This is achieved by maintaining a log of all jobs (services under execution) consisting of Job ID, service ID, user ID, Service Provider ID, Job status, report and the next three ranked service provider ID. For effective monitoring, each service execution process of the service provider is virtually divided into phases. The end of every phase of the execution phase is indicated by check points. The Monitoring & Fault handling unit monitors the service execution at every check point and takes a report of successful completion of every phase.

When a fault occurs, the Monitoring & Fault handling unit waits for 't' time to examine whether it is a transient fault. If the services are not restored within 't' time, the Monitoring & Fault handling unit automatically hands over the report taken at the check point to the next ranked service provider to complete the task and the service execution is continued without any further delay. To avoid additional storage overheads at Monitoring & Fault handling unit, the report taken at checkpoint is overwritten during the next checkpoint.

Monitoring and Fault Handling Algorithm

Step 1: Set Alloted_service_log parameters (J_ID, S_ID, U_ID, SP_ID, Status, Ranked_-Service_Providers) in the Monitoring and Fault handling unit.

```
Step 2: For each service provider do
  { If (beacon signal == True)
    { For every job do
      {Collect Job Status and Report at every check point
      Verify and Update the Alloted_service_log
        If ( Job Status == Complete)
        {Generate Job_Completion report (J_ID,S_ID,U_ID,SP_ID,
          cost, delivery time, output, status)
        Send Job_Completion report to the service delivery unit
        }
      }
    }Else{ Wait for 't' time
      If (beacon signal == false)
      { For every Job do
        {Update trust value
        Retrieve last check point report
        Select the next ranked service provider
        Send handover request to the service provider through the service selection
unit.
        Update Allotted_Service_Log.
      }
    }
  }
}
```

Theoretical analysis

In this section, the theoretical analysis has been carried out to determine the service recovery overhead, the maximum number of jobs that can be completed by n_p service providers and the success rate for the proposed framework with and without fault. If n_{fs} be the feasible set of service providers, k be the number of criteria, y be the number of faulty jobs and n_f be the number of faulty service providers, then the Service recovery Overhead O_{cr} for SSF-P [2] is found to be

$$O_{cr} = n_f \gamma n_{fs}\left(k\left(n_{fs} - 1\right) + 5\right) \tag{6}$$

And the Service Recovery Overhead O_{cr} for FTSSF is found to be

$$O_{cr} = n_f \gamma \tag{7}$$

If T is the total available time, t is the execution time of each job and N is the total number of jobs submitted for execution, then

Case (i): When there are NO Faults

When there are no faults, the number of jobs N_c that can be completed by ONE service provider is found to be

$$N_c = \left(\frac{T}{t}\right) \tag{8}$$

However, if there are n_p service providers, then the number of jobs N_c that can be completed is found to be

$$N_c = \sum_{j=1}^{n_p} \left(\frac{T_j}{t_j}\right) \tag{9}$$

Then, for FTSSF, the success rate (SR) for n_p service providers is found to be

$$N_c = \sum_{j=1}^{n_p} \left(\frac{T_j}{t_j}\right) \tag{10}$$

Case (ii): When there are Faults

When there are faults, the number of jobs N_c that can be completed by ONE service provider is found to be

$$N_c = \left(\frac{T}{t + t_r}\right) \tag{11}$$

where t_r is the recovery time of a faulty job.

However, if there are n_p service providers, then the number of jobs N_c completed by n_p service providers with n_f faulty service providers is found to be

$$N_c = \sum_{j=1}^{n_f} \left(\frac{T_j}{t_j + t_{rj}}\right) + \sum_{j=1}^{n_p - n_f} \left(\frac{T_j}{t_j}\right) \tag{12}$$

Then for FTSSF, the Success rate (SR) for n_p service providers with n_f faulty service providers is found to be

$$SR = \sum_{j=1}^{n_f} \left(\frac{T_j}{N_j(t_j + t_{rj})}\right) + \sum_{j=1}^{n_p - n_f} \left(\frac{T_j}{N_j t_j}\right) \tag{13}$$

Simulation results

The proposed fault tolerant service selection framework was implemented and the simulation results were obtained for the proposed framework FTSSF, SSF-P [2] and SHAPC [19]. The objective in comparing the proposed approach with SHAPC [19] is that SHAPC provided an excellent self healing methodology for pervasive computing and it outperformed all the other related frameworks. The FTSSF and SSF_P is compared to show the improvement in performance in terms of service delay and service recovery overhead without compromising on success rate and service registration overhead. The proposed framework was simulated using Microsoft Visual studio 2008 c# in .NET framework 3.5. Microsoft SQL server was used as a backend to store the information about the service providers. The values of the parameters that are used in the simulation are given in Table 1.

Table 1 Simulation Parameters

Sl. No.	Simulation parameters	Values
1	Number of Clusters	5 to 50
2	Mobility	0 to 100 m/s
3	Mobility model	Random way point model
4	Number of service providers per cluster	2 to 20
5	Number of services in a service provider	2 to 25
6	Simulation Time	0 - 1000 sec
7	Number of Users	1 to 1000
8	Number of criteria	2 to 10
9	Percentage of faulty Service Provider	1% to 50%
10	Service Provider Load	1% (min) to 50% (max)

Analysis on successful job completion

A pervasive environment was generated. The service providers were permitted to register initially for providing services to the users. The users who are interested in availing the services were also permitted to register. The mobility was set as 20 m/s. The total number of jobs completed at a service provider with Zero fault for SSF-P, FTSSF for different simulation time was obtained and is shown in Figure 2. Faults were induced in the pervasive environment and the number of successful job completion in a particular service provider for FTSSF, SSF-P and SHAPC was obtained and is shown in Figure 2.

It is observed that the proposed framework improves the number of successful job completion. This is due to the fact that the proposed framework restores the service execution quickly thereby enabling more number of jobs to be completed successfully.

Performance analysis in terms of success rate

The number of Service providers was set as 10. Each service provider was permitted to offer services. The maximum load on the service provider was set as 20%. The mobility was set as 20 m/s. Faults were induced in the environment such that 10% of the service provider is at fault. The success rate for SSF-P (Zero Fault), FTSSF, SSF-P and SHAPC was obtained and is shown in Figure 3.

The results show that at a particular time t, the success rate of proposed framework is high. This is primarily because of the effective monitoring and fault handling of the proposed framework.

Effect of mobility on success rate

The number of Service providers was set as 10. Each service provider was permitted to offer services. The maximum load on the service provider was set as 20% and the service providers were set to move randomly. The mobility of the service provider was varied as 20 m/s, 40 m/s, 60 m/s, 80 m/s and the effect of mobility on the Success rate of the proposed framework was studied and is shown in Figure 4

It is found that the mobility of the service provider affects the success rate.

Performance analysis in terms of service recovery overhead

The mobility was set as 20 m/s. The number of Service providers was set as 10. Each service provider was permitted to offer services. The maximum load on the service

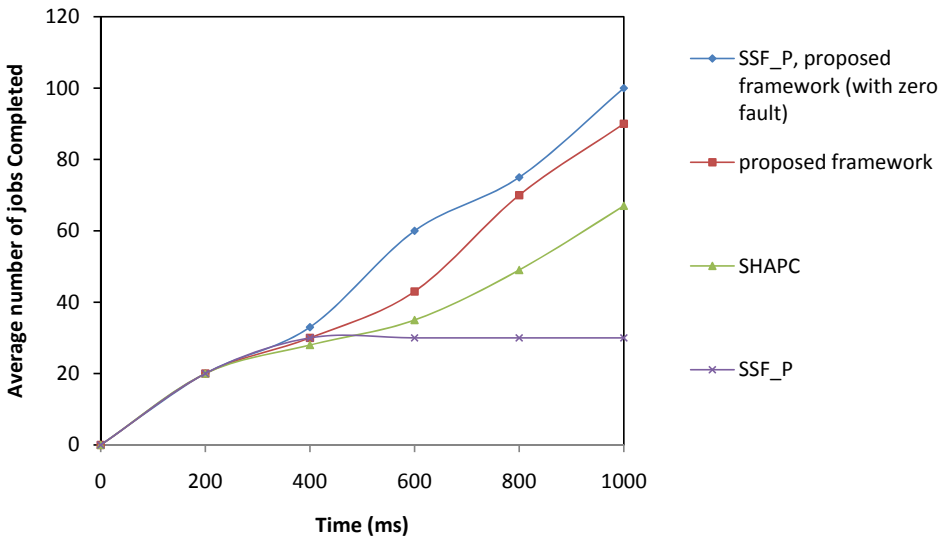

Figure 2 Average Number of Jobs Completed.

provider was set as 20%. Faults were induced in the environment such that 10% of the service provider is at fault. The Service recovery overhead for the proposed framework, SSF-P and SHAPC was obtained and the results are shown in Figure 5.

It was found that the service recovery overhead for the proposed framework is very minimal and the proposed framework provides significant improvement in the service recovery overhead.

Effect of service recovery overhead for different load

The number of Service providers was set as 10. Each service provider was permitted to offer services. The maximum load on the service provider was set as 20%. Mobility was set as 20 m/s. The load has been varied. Faults were induced in the environment. For

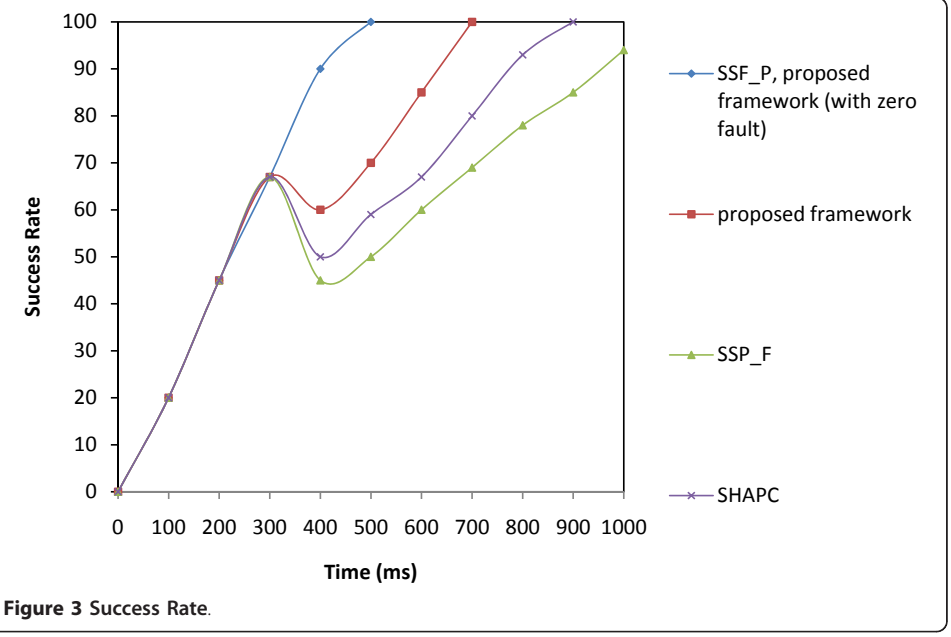

Figure 3 Success Rate.

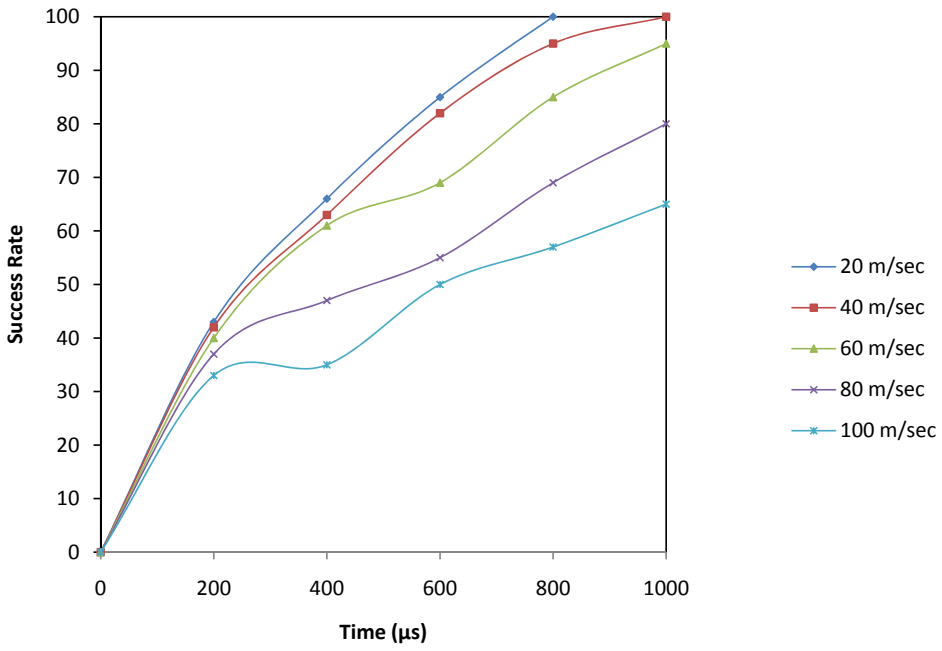

Figure 4 Effect of mobility on the success rate.

different percentage of faulty service providers, the effect of service recovery overhead for different loads on the service provider for the proposed framework was obtained and is shown in Figure 6.

It was observed that as the load on the service provider is increased the service recovery overhead also increases.

Analysis on service delay

The mobility was set as 20 m/s. The number of Service providers was set as 10. Each service provider was permitted to offer services. The maximum load on the service

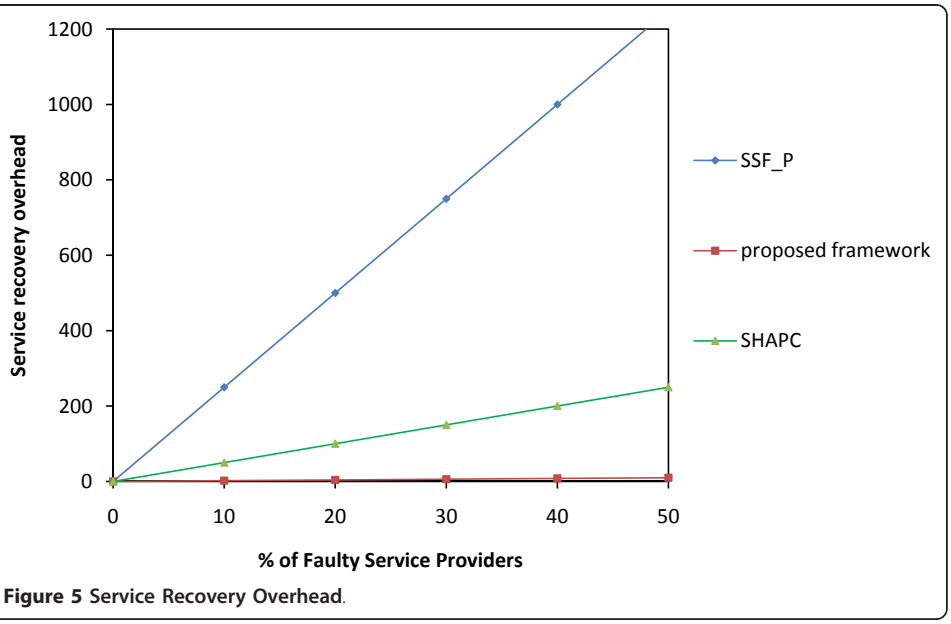

Figure 5 Service Recovery Overhead.

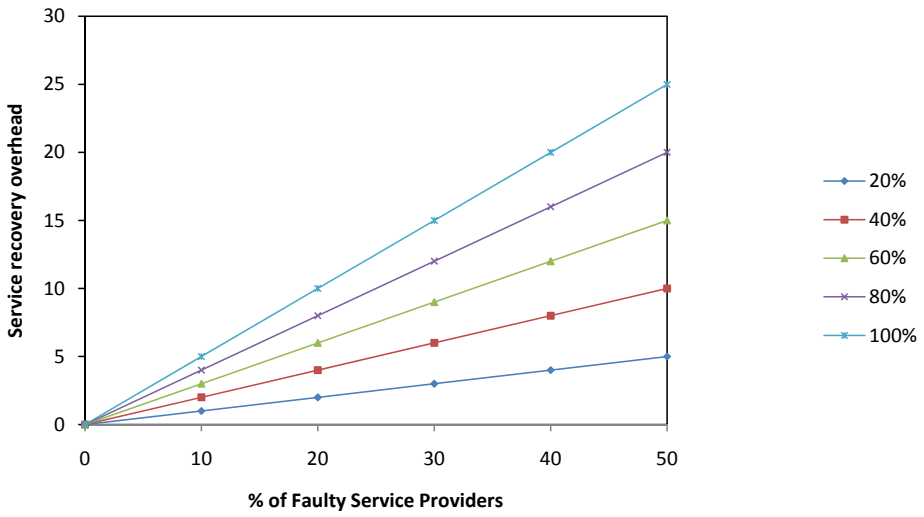

Figure 6 Effect of load on Service Recovery overhead.

provider was set as 20%. Faults were induced in the environment such that 10% of the service provider is at fault. The average service delay for the proposed framework, SSF-P and SHAPC was obtained and the results are shown in Figure 7.

It has been observed from Figure 7, the service delay is much lower for the proposed framework. In addition, the service delay increases exponentially for every increase in percentage of faulty service providers.

Conclusion

In this paper, a novel Fault Tolerant Service Selection Framework (FTSSF) for Pervasive Computing was proposed. The proposed framework was simulated and the experimental results on the number of successful jobs completed, success rate, service

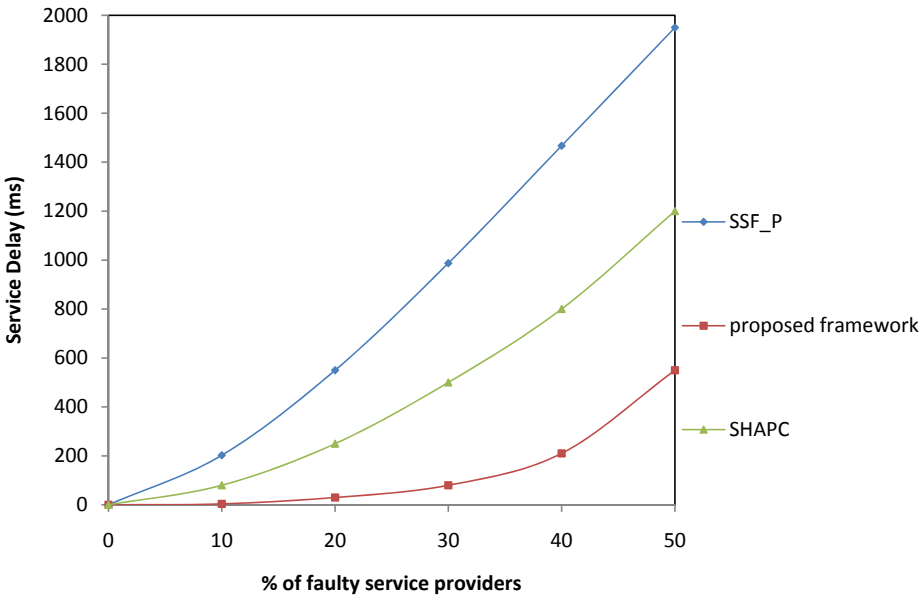

Figure 7 Service Delay.

recovery overhead, service delay was obtained for the proposed framework, SSF-P [2] and SHAPC [19]. The experimental results prove that the proposed framework is efficient and fault tolerant. It was also observed that the mobility affects the fault tolerance behavior of the system.

Acknowledgements
The authors wish to thank Karunya University for providing infrastructure for carrying out the simulation and financial support. The authors thank the senior professors and the technical experts for providing valuable suggestions to improve the quality of the research paper.

Author details
[1]Karunya University, Coimbatore, TamilNadu, India 641114 [2]BHEL, Trichy, India

Authors' contributions
SS analyzed the requirement of fault tolerant behavior, designed the framework, conducted the experiment and drafted the manuscript. KE supported in carrying out the experiment and drafted parts of the manuscript and revised it. EBR contributed on the design of the framework and revised the manuscript content to high professional standards. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 29 November 2011 Accepted: 11 March 2012 Published: 11 March 2012

References

1. Kim MJ, Kumar M, Shirazi BA (2006) Service discovery using volunteer nodes in heterogeneous pervasive computing environments. Pervasive and Mobile Computing 2(3):313–343. doi:10.1016/j.pmcj.2006.04.002.
2. Silas S, Rajsingh EB, Ezra K (2011) Scalable and reliable methodology for service selection in pervasive computing. Proc 2011 3 rd Int Conf Electron Comput Technol 1:188–191
3. Ahamed SI, Sharmin M (2008) A trust-based secure service discovery (TSSD) model for pervasive computing. Comput Commun 31(18):4281–4293. doi:10.1016/j.comcom.2008.07.014.
4. Mokhtar SB, Preuveneers D, Georgantas N, Issarny V, Berbers Y (2008) EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support. J Syst Softw 81(5):785–808. doi:10.1016/j.jss.2007.07.030.
5. Chakraborty D, Joshi A (2003) Anamika: distributed service composition architecture for pervasive environments. ACM SIGMOBILE Mobile Comput Comm 7(1):38–40. doi:10.1145/881978.881989.
6. Sailhan F, Issarny V (2005) Scalable service discovery for MANET. PERCOM '05 Proc Third IEEE Int Conf Pervasive Comput Commun 235–244. doi:10.1109/PERCOM.2005.36
7. Kalasapur S, Kumar M, Shirazi BA (2007) Dynamic service composition in pervasive computing. IEEE Trans Parallel Distr Syst 18(7):907–918
8. Dabrowski C, Mills K, Quirolgico S (2007) Understanding failure response in service discovery systems. J Syst Softw 81:785–808
9. Wang Y, Vassileva J (2007) Toward trust and reputation based web service selection: a survey. Proc Int Trans Syst Sci Appl ITSSA J 3(2):118–132
10. Sharmin M, Ahmed S, Ahamed SI (2005) SAFE-RD (Secure, Adaptive, Fault Tolerant, and Efficient Resource Discovery) in Pervasive Computing Environments. Int Conf Inf Technol, Coding Comput 2:271–276
11. Cai H, Xiaohui Hu, Qingchong Lu, Cao Q (2009) A novel intelligent service selection algorithm and application for ubiquitous web services environment. Expert Syst Appl 36(2):2200–2212. doi:10.1016/j.eswa.2007.12.071.
12. Wang H-C, Lee C-S, Ho T-H (2007) Combining subjective and objective QoS factors for personalized web service selection. Expert Syst Appl 32(2):571–584. doi:10.1016/j.eswa.2006.01.034.
13. Chetan S, Ranganathan A, Campbell R (2005) Towards fault tolerant pervasive computing. IEEE Tech Soc Mag 24(1):38–44. doi:10.1109/MTAS.2005.1407746.
14. Hwang S-Y, Ee-Peng Lim, Lee C-H, Chen C-H (2008) Dynamic web service selection for reliable web service composition. IEEE Trans Serv Comput 1(2):104–116
15. Lin Chia-Feng, Sheu Ruey-Kai, Chang Yue-Shan, Yuan Shyan-Ming (2011) A relaxable service selection algorithm for QoS-based web service composition. Inf Softw Technol 53(12):1370–1381. doi:10.1016/j.infsof.2011.06.010.
16. Fang C-L, Liang D, Lin F, Lin C-C (2007) Fault tolerant web services. J Syst Archit 53(1):21–38. doi:10.1016/j.sysarc.2006.06.001.
17. Koushanfar F, Potkonjak M, Sangiovanni-Vincentell A (2002) Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks. Sensors' 2002: Proc IEEE 2:1491–1496
18. Wu W, Cao J, Yanga J (2008) A fault tolerant mutual exclusion algorithm for mobile ad hoc networks. Pervasive and Mobile Computing 4:139–160. doi:10.1016/j.pmcj.2007.08.001.
19. Ahmed S, Ahamed SI, Sharmin M, Hasan CS (2009) Self-healing for autonomic pervasive computing. Autonomic Communication, Springer US 285–307. doi: 10.1007/978-0-387-09753-4
20. Ahmed S, Sharmin M, Ahamed SI (2005) PerAd-service: A middleware service for pervasive advertisement in m-business. COMPSAC: 29th Annu Int Comput Softw Appl Conf 2:17–18
21. Hu P, Indulska J, Robinson R (2008) An Autonomic Context Management System for Pervasive Computing. Sixth Annu IEEE Int Conf Pervasive Comput Commun 2008:213–223

22. Padovitz A, Loke SW, Zaslavsky A (2008) The ECORA framework: A hybrid architecture for context-oriented pervasive computing. Pervasive and Mobile Computing 4:182–215. doi:10.1016/j.pmcj.2007.10.002.
23. Bourdenas T, Sloman M, Lupu EC (2010) Self-healing for pervasive computing systems. Architecting dependable systems VII. Springer-Verlag Berlin, Heidelberg pp 1–25
24. Gao Z-P, Chen J, Qiu X-S, Meng L-M (2009) QoE/QoS driven simulated annealing-based genetic algorithm for Web services selection. J China Universities of Posts Telecomm 16(1):102–107
25. Ozerol G, Karasakal E (2008) A parallel between regret theory and outranking methods for multicriteria decision making under imprecise information. Theor Decis 65(1):45–70. doi:10.1007/s11238-007-9074-y.
26. Wei-xiang L, Bang-yi L (2010) An extension of the Promethee II method based on generalized fuzzy numbers. Expert Syst Appl 37(7):5314–5319. doi:10.1016/j.eswa.2010.01.004.
27. Behzadian M, Kazemzadeh RB, Albadvi A, Aghdasi M (2010) Decision Support PROMETHEE: a comprehensive literature review on methodologies and applications. Eur J Oper Res 200(1):198–215. doi:10.1016/j.ejor.2009.01.021.
28. Roux O, Duvivier D, Dhaevers V, Meskens N, Artiba A (2008) Multicriteria approach to rank scheduling strategies. Int J Prod Econ 112(1):192–201. doi:10.1016/j.ijpe.2006.08.020.

doi:10.1186/2192-1962-2-5
Cite this article as: Silas *et al.*: A novel fault tolerant service selection framework for pervasive computing. *Human-centric Computing and Information Sciences* 2012 **2**:5.