

FTSCP: An Efficient Distributed Fault-Tolerant Service Composition Protocol for MANETs

Zhenguo Gao¹, Sheng Liu¹, Ming Ji¹, Jinhua Zhao², and Lihua Liang¹

¹ College of Automation, Harbin Engineering University, Harbin, 150001, China
{gag, liusheng, jim, lianglh}@hrbeu.edu.cn

² School of Astronautics, Harbin Institute of Technology, Harbin, 150001, China
zhaojinhua@hit.edu.cn

Abstract. Service composition, which enables users to construct complex services from atomic services, is an essential feature for the usability of Mobile Ad hoc Networks (MANETs). Service composition in MANETs should be fault-tolerant and distributed. Efforts in this area are rare and sounds not meet the requirements very well. In this paper, we present a distributed Fault-Tolerant Service Composition Protocol (FTSCP) for MANETs. FTSCP has two main features. Firstly, it is efficient for that all atomic services consisted in a composite service are discovered in just one service discovery session. Secondly, it is fault-tolerant since that service composition process is entirely under the supervision of Execution Coordinator (EC) and inaccessible services can be rediscovered transparently. Both mathematical analysis and simulation results show that FTSCP outperforms another broker-based service composition protocol for MANETs in both terms of packet overhead and promptness.¹

1 Introduction

Service composition refers to the process of combining simple services to form a larger, more complex service[1]. This offers users a great degree of transparency in discovery and selection of required services, instead of having to be cognizant of all the details about the simpler services that constitute the complex ones. A service can be any software or hardware entity that can be utilized by others.

In the context of service composition, an atomic service is a service provided by one single node that does not rely on any other services to fulfill user requests; a composite service is compound service constructed from other composite services or atomic services; the number of necessary atomic services consisted in the composite service is called as composite service size.

Service composition has been extensively studied in the context of wired networks where service composition always adopts a centralized approach[2,3,4]. In

¹ This work is supported by National Postdoctoral Research Program (No.NPD060234), Postdoctoral Research Program of HeiLongJiang (No.HLJPD060234), Fundamental Research Foundation of Harbin Engineering University (No.HEUFT06009), Fostering Fundamental Research Foundation of Harbin Engineering University (No.HEUFP07001).

centralized approaches, a particular node acting as a service composition manager supervises the whole process of service integration and execution.

With the popularity of portable devices, MANETs are abstracting more research efforts. Most efforts are focused on some other aspects, such as MAC protocol, routing, broadcasting, etc. Only very limited efforts has been performed on service composition in MANETs, such as [1,5]. Furthermore, these approaches are not satisfiable for variable reasons, such as more packet overhead, longer response time, etc.

Because of the highly dynamic topology of MANETs, service composition architectures should be distributed, fault-tolerant, and context aware.

In this paper, we present a Fault-Tolerant Service Composition Protocol (FTSCP) that meets the first two requirements. Context aware characteristics will be introduced in future work. Additionally, all atomic services can be discovered in just one service discovery session in FTSCP. Mathematical analysis and extensive simulations confirms the efficiency of FTSCP.

The organization of the rest of the paper is as follows: Section 2 gives an overview of related works. Section 3 shows the architecture of FTSCP and its operation procedure. Section 4 compares the performance of FTSCP and the approach proposed in[1] through mathematical analysis and extensive simulations. Section 5 concludes the paper.

2 Related Works

Research efforts in service composition follow two directions. One direction of research tries to define semantic-rich machine-readable description languages that can be used to describe services and workflows appropriately. The other direction aims in developing architectures that facilitate service composition. We focus on service composition architectures since we believe that it is critical in MANETs and requires a different approach from service composition architectures in wired context.

Current service composition architectures have been designed with the inherent assumption of fixed network infrastructure and high bandwidth communication channels, such as eFlow[2], CMI[3], Ninja[4], etc. These architectures are based on a centralized manager or broker that manages the various tasks of service composition, such as service discovery, combination of different services, and management of the information flow between services.

Because of the highly dynamic nature of MANETs, service composition architecture in MANETs calls for an alternate design approach. Notifying this, some research efforts have been performed in these areas[1][5]. Chakraborty et. al[1]proposed a distributed broker-based protocol for service composition in pervasive/ad hoc environments. In their approach, service composition is delegated to a broker selected during broker arbitration phase and then the broker searches for all necessary component services one by one. Obviously, broker arbitration and iterated service discovery scheme causes too much packet overhead, which is a great drawback since bandwidth is a critical resource in MANETs. Basu et al.[5] described a hierarchical task-graph based service composition protocol for

MANETs. In their work, a composite service is represented as a task-graph with leaf nodes representing atomic services. Sub trees of the graph are constructed in distributed mode, and service composition process is coordinated by the client. However, the coordinator uses flood scheme to search for all atomic services, which leads to larger packet overhead.

3 FTSCP

Our FTSCP service composition architecture consists of 7 components, as shown in Fig. 1. UnRecoverable Exception Manager (UREM) deals with all exceptions happened during the overall service composition period. It usually sends a notify message to the user application. Composite Service Request Parser (CSRP) extracts atomic services from composite service request received from user application. A notification will be sent to UREM in case of exception. Service Discovery Manager (SDM) finds all atomic services necessary to compose the requested composite service in just one service discovery session. Execution Coordinator (EC) schedules and supervises the executions of all atomic services. Under the supervision of EC, all atomic services cooperate in harmony to fulfill the requested composite service. Result Constructor (RC) combines the results of all atomic services and gets the final integrated results. Service Relationship Info Manager (SRIM) stores the information obtained by the CSRP when parsing composite service request. Context Manager (CM) collects and stores context information, which facilitates context-aware semantic service matching during service discovery sessions.

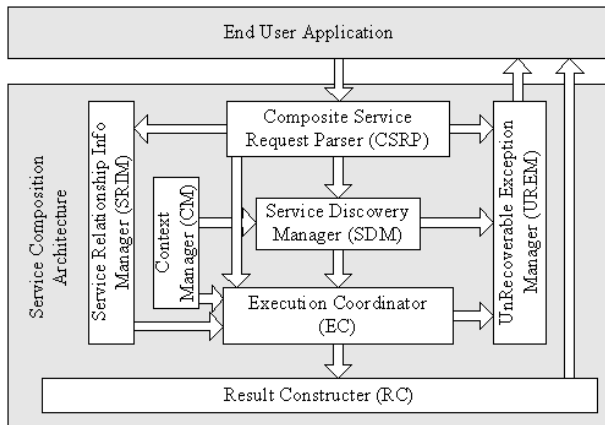


Fig. 1. Service Composition Architecture

FTSCP essentially consists of flowing four phases: 1) Composite service request parse phase; 2) Service discovery phase; 3) Service execution coordination phase; 4) Result construction phase.

3.1 Composite Service Request Parse Phase

In this phase, CSRП receives service composite request from end user application. CSRП extracts all atomic services necessary to fulfill the composite service request as well as the dependencies between these services. This information is stored in SRIM. This information is necessary for service scheduling. Composite service requests describes a task that requires coordination of atomic services as well as other composite services. We designed a new Document Type Definition (DDT) file for describing service composite request as follows.

```

<!ELEMENT Request (Service|AtomicService)*>
<!ELEMENT Service (
    (Service|AtomicService, Element2Operator, Service|AtomicService)
    |(Factor, Element1Operator, Service|AtomicService)
    |AtomicService)*>
<!ATTLIST Element2Operator name
    (SeqNormal|SeqArbitrary|SeqInitial|ConcurrentNoCom|ConcurrentCom)
    #REQUIRED>
<!ATTLIST Element1Operator name (Repeat) #REQUIRED>
<!ATTLIST Factor Value CDATA #REQUIRED "1">
<!ATTLIST CompositeService Name CDATA #IMPLIED >
<!ATTLIST AtomicService Name CDATA #REQUIRED>
    
```

Coordination relationships between atomic services are listed in Table 1.

Table 1. Notations used in the paper

Relation	Meaning	DTD Keyword
$S_1 \odot S_2$:	service S_1 and S_2 should be executed in sequence.	SeqNormal
$S_1 \diamond S_2$:	either service S_1 or S_2 can be executed first.	SeqArbitrary
$S_1 \vdash S_2$:	service S_1 must be activated before S_2 .	SeqInitial
$S_1 \parallel S_2$:	S_1 and S_2 can run in parallel without inter-communications.	ConcurrentNoCom
$S_1 \Downarrow S_2$:	S_1 and S_2 should run in parallel with inter-communications.	ConcurrentCom

3.2 Service Discovery Phase

In this phase, all atomic services will be discovered by SDM using some service discovery protocols implemented in the system. All service discovery protocols relies on some kind of service request packet spreading scheme to search for matched services. Service request packets and corresponding service reply packets makeup the main part of packet overhead. For each necessary atomic service, if one service discovery session is performed, more request packets will be generated. To reduce packet overhead, we implements a multi-task service discovery protocol (referred as MDFNSSDP)[6] which can find all atomic services in just one service discovery session. For each atomic service, there may be multiple matched services. The service with maximum priority is selected. All founded services are cached in SDM for possible latter use. During service discovery, context information can be get from CM to facilitate service matching operation.

3.3 Service Execution Coordination Phase

In this phase, all discovered atomic services will be performed under the supervision of EC. During this period, all atomic services cooperate in harmony to fulfill the requested composite service. The operation cycle in EC is as follows.

- **Step1.** From the information get from SRIM, EC knows the dependent relationships among atomic services. From the service descriptions of discovered atomic services from SDM, EC knows how to invoke services. Thus, EC can then schedule the execution of these services.
- **Step2.** EC invokes services according to the its schedule and supervise the operation of these services. During this process, some runtime information and temporary results are cached into Run Time DataBase (RTDB). The information in RTDB is used to facilitate the work of EC. If all services are finished successfully, results are passed to the RC component and this stage completes. Otherwise, in case of any exception during the execution, EC first checks if the exception is recoverable. If it is unrecoverable, an exception is thrown to UREM and the stage finished. If the exception is recoverable, such as that a service becomes unreachable, then goes to step 3.
- **Step3.** EC starts a run time service discovery session to find servers that provides the losted service. Recall that in service discovery phase, all founded service information are cached. Thus, a matched service that can replace the lost service may be found directly from SDM. Thus, new service discovery session is not necessary. Information in RTDB is used to in semantic service matching. Then it goes to step1.

Thus, the system is resistant to topology changes, and so it is suitable for MANETs where topology variation is frequent.

3.4 Result Construction Phase

Result Construction Phase When service execution coordination phase is finished successfully, it changes to result construction phase. In this phase, results of all atomic services are combined and the final integrated results are given to the user application. Till now, user application's service composition request is completed.

4 Performance Analysis and Simulations

4.1 Select Comparative Solutions

We implemented our service composition protocol in Glomosim[9]. To make comparative study, the Broker-based Distributed Service Composition Protocol (BDSCP) proposed in [1] is also implemented.

BDSCP consists of four phases. 1) In Broker Arbitration Phase, the client broadcasts a Broker Request Packet, and then in a following fixed period TBAP,

the node waits for broker reply packets. The broker request packet floods in the network for limited hops. All nodes receiving the packet reply with a Broker Reply Packet enclosing its runtime priority value. The client then selects the neighbor with biggest priority value as its broker. 2) In Service Discovery Phase, the selected Broker uses the underlying GSD service discovery protocol to discover all required atomic services one by one. 3) In Service Integration Phase, atomic services are scheduled. 4) In Service Execution Phase, atomic services are executed according to the schedule.

4.2 Performance Metrics

The superiority of FTSCP over BDSCP mainly results from two aspects: 1) Broker arbitration phase is eliminated since broker arbitration is helpless in MANETs, and 2) One service discovery session searching for all atomic services is used instead of one service discovery session for each atomic service. Following performance metrics are used in our analysis and simulation study.

- **Number of Composition Preparation Packets:** In BDSCP, composition preparation packets include 1) broker request packets and broker reply packets, and 2) service request packets and service reply packets. Whereas in FTSCP, only the second part is included. This metric is averaged over all composite service requests successfully instantiated.
- **Delay of Composite Service Discovery:** It is the interval between the generation of a service composition request and the time when all necessary atomic services are founded. This metric is averaged over all composite service requests that are successfully instantiated. It measures the promptness of service composition protocols.
- **Composition Efficiency:** Composition efficiency refers to the fraction of service composite requests that all atomic services are discovered successfully.

4.3 Performance Analysis

In this section, mathematical analysis will be performed to estimate the performance of FTSCP and BDSCP in terms of performance metrics listed in previous section. Although MDFNSSDP outperforms GSD very much both in packet overhead and promptness[6], the analysis in the section is based on the following assumptions for simplicity: 1) M_{SDP} (averaged number of service request packets and service reply packets sent in one service discovery session) in FTSCP is equal to that in BDSCP, 2) T_{SDP} (averaged interval between the time that a service discovery session is started and the time the first reply packet with matched services is received) in FTSCP is equal to that in BDSCP, 3) Failure service discovery session is not retried.

Analysis on Number of Composition Preparation Packets. In FTSCP, composition preparation packets include service request packets and reply packets. Additionally, only one service discovery session is performed. Hence M_{FTSCP} (number of composition preparation packets in FTSCP) equals M_{SDP} .

In BDSCP, besides service request packets and service reply packets, composition preparation packets also include broker request packets and broker reply packets. Furthermore, for each atomic service, one service discovery session is performed. Hence, M_{BDSDP} (number of composition preparation packets in BDSCP) can be calculated as equation . Here M_{SDP} is the averaged number of service request packets and service reply packets sent in one service discovery session, M_{BAP} is averaged number of broker request packets and broker reply packets, M_{BQP} is averaged number of broker request packets, M_{BPP} is averaged number of broker reply packets).

$$M_{BDSCP} = M_{BAP} + M_{SDP} \times k = (M_{BQP} + M_{BPP}) + M_{SDP} \times k \quad (1)$$

In broker arbitration phase in BDSCP, broker request packets flood through the client’s b -hop neighbor sets. Noticing that last-hop nodes do not rebroadcast packets, M_{BQP} (averaged number of broker request packets sent in broker arbitration phase in BDSCP) can be calculated as follows. Here ρ is averaged number of nodes in unit area, r is Radio range, b is the hop limit of broker request packets in BDSCP, and $n_1 = \rho\pi r^2$

$$M_{BQP} = \sum_{h=0}^{b-1} n(h) \approx \rho\pi(r(b-1))^2 = n_1(b-1)^2 \quad (2)$$

Each node receiving broker request packets will send back a broker reply packet in unicast mode. In case of $b > 1$, a broker reply packet will travel several hops to reach the source node that requires brokers. These relayed packets should also be calculated in this metric. Hence,

$$M_{BPP} = \sum_{h=1}^b hn(h) \approx \sum_{h=1}^b h(\rho\pi(rh)^2 - \rho\pi(r(h-1))^2) = n_1 \frac{4b^3 + 3b^2 - b}{6} \quad (3)$$

Combining Eqn. 1, 2, and 3, we have:

$$M_{BDSCP} = (M_{BQP} + M_{BPP}) + kM_{SDP} = n_1 \frac{4b^3 + 9b^2 - 13b + 6}{6} + kM_{SDP} \quad (4)$$

Compared with BDSCP, the number of saved composition preparation packets of FTSCP is:

$$M_{BDSCP} - M_{FTSCP} = n_1 \frac{4b^3 + 9b^2 - 13b + 6}{6} + (k-1)M_{SDP} > 0 \quad (5)$$

Analysis on Delay of Composite Service Discovery. T_{FTSCP} (delay of composite service discovery in FTSCP) and T_{BDSCP} (delay of composite service discovery in BDSCP) can be calculated as eqn. 6 and 7, respectively. Here T_{BAP} stands for the period that a client waits for broker reply packets in BDSCP.

T_{SDP} is the averaged interval between the time that a service discovery session is started and the time the first reply packet with matched services is received.

$$T_{FTSCP} = T_{SDP} \quad (6)$$

$$T_{BDSCP} = T_{BAP} + k \cdot T_{SDP} \quad (7)$$

Combining Eqn. 6, and 7, we have:

$$T_{BDSCP} - T_{FTSCP} = T_{BAP} + (k - 1) \cdot T_{SDP} > 0 \quad (8)$$

Analysis on Composition Efficiency. A succeeded composite service request requires that all atomic services are discovered successfully. Hence, $P_{ComSuc}(k)$ (the probability of a composite service with size k being succeeded) can be calculated as eqn 9. Here k is composite service size, p is the probability of founding a requested atomic service in one service discovery session. There is no difference in this term between FTSCP and BDSCP.

$$P_{ComSuc}(k) = p^k \quad (9)$$

4.4 Simulation Models

In our simulation studies, the Distributed Coordination Function (DCF) of IEEE 802.11 is used as the underlying MAC protocol. Random Waypoint Mode (RWM) is used as the mobility model. In this model, a node moves towards its destination with a constant speed $v \in [V_{MIN}, V_{MAX}]$, which means a variable uniform distributed between V_{MIN} and V_{MAX} . When reached its destination, a node will keep static for a random period $t_P \in [T_{MIN}, T_{MAX}]$. Then the node will randomly select a new destination in scenario and move to the new destination with new speed. The process will repeat permanently. In our simulations, t_P is fixed to 0s.

The advantages of FTSCP over BDSCP that can be easily demonstrated through simulations rely in the phases before service schedule and execution. Service schedule and execution cause little impacts on evaluating the relative qualities of FTSCP and BDSCP. Hence, service schedule and execution are omitted in our simulation for simple.

In all simulations, some basic parameters are set as shown in Table. 2. Simulation scenarios are created with 100 nodes initially distributed according to the steady state distribution of random waypoint mobility model. At the beginning of each simulation, predefined number of nodes are randomly selected as servers. These selected servers provide randomly selected services. During each simulation, 10 service composition requests are started at randomly selected time by randomly selected nodes. Atomic services consisting of each service composition request are also selected randomly.

Basic parameters of the underlying service discovery protocols are set as shown in Table 3.

Table 2. Basic parameters

Parameters	Value	Parameters	Value
scenario area	1000m × 1000m	service composition request number	10
node number	100	composite service size	1~7
radio range	250m	service discovery protocol in FTSCP	MDFNSSDP
simulation time	1000s	service discovery protocol in BDSCP	GSD
bandwidth	2Mbps	hop limit of broker packets in BDSCP	1
mobility	RWP	initial node placement	steady state
T_{BAP}	0.1s		

Table 3. Basic parameters in GSD and MDFNSSDP

Parameters	Value	Parameters	Value
service number	80	hop limit of advertisement packets	1
service group number	1	service number in each group	10
service number in each group	10	service advertisement packet valid time	30s
service advertisement interval	20s	hop limit of service request packets	3

4.5 Simulation Results

In this section, we inspect the impacts of composite service size on performance metrics through extensive simulations. In all the following figures, error bars report 95% confidence. In case of results of composition efficiency, the lower confidence limits of the single side confidence intervals are shown.

To inspect the impacts of composite service size, 2 other simulation sets are run. In all these simulations, node speed is fixed to 0m/s. Each simulation set

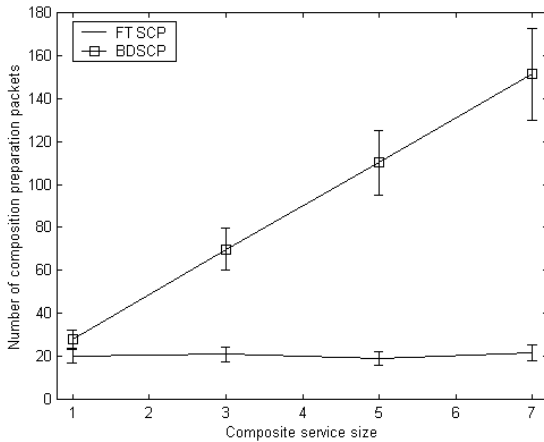


Fig. 2. number of composition preparation packets vs. composite service size

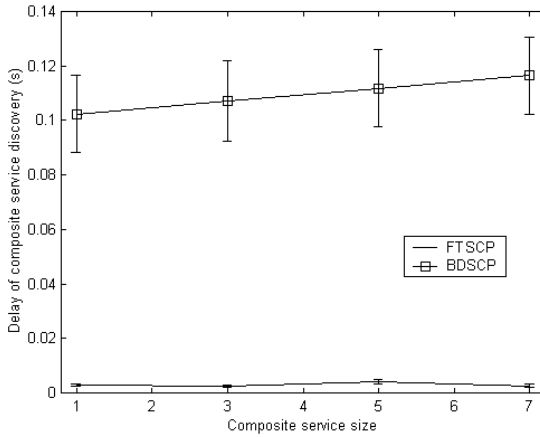


Fig. 3. delay of composite service discovery vs. composite service size

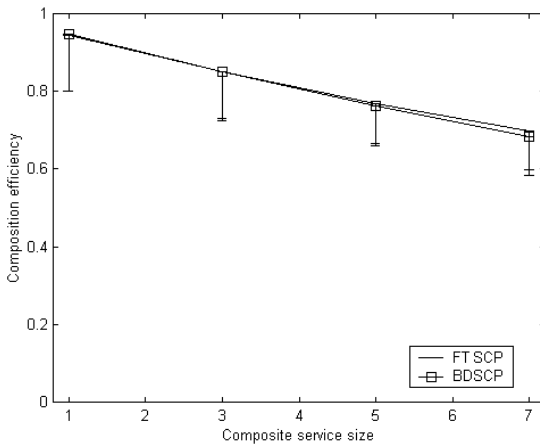


Fig. 4. composition efficiency vs. composite service size

includes 4 subsets where composite service size is set to 1, 3, 5, and 7, respectively. Each subset consists of 100 similar simulations.

Fig.2 shows the impact of composite service size on number of composition preparation packets. In BDSCP, for each atomic service, one service discovery session has to be performed. Thus, as composite service size increases, the number of composition preparation packets will increase. In FTSCP, however, no matter how many atomic services are consisted in the requested composite service, only one service discovery session will be performed. Hence, the number of these packets keeps almost constant.

Fig.3 shows that FTSCP is much more prompt than BDSCP. Delay of composition service discovery in BDSCP increases gradually as composite service size increases. This results from more service discovery sessions. While in FTSCP, this metric keeps almost constant. These results verify Eqn. 8.

Fig.4 shows the impact of composite service size on composition efficiency. In both FTSCP and BDSCP, as composition size increases, composition efficiency decreases quickly. This verifies the analysis result of Eqn.9.

5 Conclusions

In this paper, we have presented a distributed Fault-Tolerant Service Composition Protocol (FTSCP) for MANETS. In FTSCP, service composition process is entirely under the supervision of EC and inaccessible services can be rediscovered transparently. Hence, FTSCP is fault-tolerant. In service discovery phase, context information is used to perform semantic-rich service matching and select the best service. Hence, FTSCP is context-aware. In FTSCP, all atomic services consisted in a composite service are discovered in just one service discovery session. Therefore, it is efficiency. Mathematical analysis and simulation results confirm the superiority of FTSCP over another broker-based service composition protocol for MANETS in terms of packet overhead and promptness.

References

1. Chakraborty, D., Joshi, A., Yesha, Y., Finin, T.: Service composition for mobile environments. *Journal on Mobile Networking and Applications (MONET)*, 435–451 (2005) (special issue on Mobile Services)
2. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.: Adaptive and dynamic service composition in eflow. Technical Report, HPL-200039, Software Technology Laboratory, Palo Alto, USA (2000)
3. Schuster, H., Georgakopoulos, D., Cichocki, A., Baker, D.: Modeling and composing service-based and reference process-based multienterprise processes. In: Wangler, B., Bergman, L.D. (eds.) *CAiSE 2000*. LNCS, vol. 1789, pp. 247–263. Springer, Heidelberg (2000)
4. Katz, R.H., Brewer, E.A., Mao, Z.M.: Fault-tolerant, scalable, wide-area internet service composition. Technical Report. UCB/CSD-1-1129. CS Division. EECS Department. UC. Berkeley (2001)
5. Basu, P., Ke, W., Little, T.D.C.: A novel approach for execution of distributed tasks on mobile ad hoc networks. In: *WCNC 2002*. Proceedings of IEEE Wireless Communications and Networking Conference, Orlando, USA, pp. 579–585. IEEE Computer Society Press, Los Alamitos (2002)
6. Gao, Z.G., Liu, S., Liang, L.H., Zhao, J.H., Song, J.G.: A Generic Minimum Dominating Forward Node Set based Service Discovery Protocol for MANETS. In: *HPCC 2007*. 2007 International Conference on High Performance Computing and Communications (accepted 2007)

7. Chakraborty, D., Joshi, A., Yesha, Y., Finin, T.: GSD: a novel group-based service discovery protocol for MANETs. In: MWCN 2002. Proceedings of the 4th IEEE Conference on Mobile and Wireless Communications Networks, pp. 140–144. IEEE Computer Society Press, Los Alamitos (2002)
8. Dey, K.A., Abowd, D.G., Salber, D.: A context-based infrastructure for smart environment. In: MANSE 1999. Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments, Dublin, Ireland, pp. 114–128 (1999)
9. Glomosim, Wireless Adaptive Mobility Lab. DEPT of Comp. SCI, UCLA, Glomosim: a scalable simulation environment for wireless and wired network system, Available: <http://pcl.cs.ucla.edu/projects/domains/glomosim.html>