Service Composition in Opportunistic Networks

Sagar A. Tamhane, Mohan Kumar
Computer Science and Engineering Department,
University of Texas at Arlington, U.S.A
Email: {sagar.tamhane@mavs.,mkumar@}uta.edu

Andrea Passarella, Marco Conti
Institute for Informatics and Telematics
National Research Council, Italy
Email: {a.passarella, m.conti}@iit.cnr.it

Abstract—**An opportunistic contact between two mobile devices takes place when they are within communication range of each other. Typically, cyber-physical environments comprise a number of mobile devices that are likely to make opportunistic contacts in time and space. In the recent past, researchers have exploited opportunistic links mostly for routing and content sharing. However, less attention has been devoted to the more general concept of opportunistic computing, which consists of utilizing any resource available in a pervasive environment by exploiting opportunistic contacts. In cyber-physical pervasive environments, opportunistic computing will be used to elaborate data coming both from the physical and the cyber worlds, according to the users' needs, and exploiting all available resources. In this paper, we develop a modular middleware architecture for service composition in opportunistic networks. In particular, we demonstrate the utility of the middleware for fault tolerant service composition in such environments. Simulation results demonstrate the effectiveness of the middleware and associated service composition scheme in pervasive environments with underlying opportunistic networks.**

Keywords-**Service Composition, Opportunistic Networks, Middleware, Fault Tolerance.**

I. Introduction

Opportunistic networks (OppNets) are created when pairs of devices communicate through many opportunistic contacts [1]. Devices such as cell phones, PDAs and laptops host services that are useful to applications executing on other devices in the network [2]. Examples of such services include: data compression / decompression, data encoding / decoding, audio / image / video processing, and others. Such services may be available within the opportunistic network, but not within direct communication range of the requesting device. An end-to-end path between a requesting device and the required service may never exist, while using infrastructure-based solutions may prove infeasible. Hence, it is necessary to make services accessible and available anywhere in the environment, perhaps with some delay. This is one of the research challenges [2] that will be one of the driving elements of future cyber-physical systems (CPS). In CPS, data will be generated both in the cyber and in physical worlds, and will undulate across the two, influencing the user behavior in both [3]. Opportunistic computing will allow CPS to enhance raw data, through filtering, adaptation and processing, by exploiting all resources available in a pervasive environment. This will make CPS a resource rich environment, where mobile users enjoy much richer resources compared to those available on their individual devices only.

In this paper[1], we propose a generic framework that exploits pair-wise contacts to enhance availability of all resources, abstracted as services. Pair-wise meeting of devices can be opportunistically exploited to reach services running on other parts of the network, which would not be possible in traditional networking schemes. Service results will be provided to the requesting application either directly during contact, or through an opportunistic path found in the network.

When an application needs to perform a service, it generates a service execution request. To facilitate service execution, each device maintains an index of services discovered. When a requested service is not available in the network, the middleware explores concatenation of multiple (available) services to compose the required service. This concept can be exploited in multiple application scenarios relevant to CPS. Consider the following e-health application. Suppose a user desires to correlate locally sensed personal data about her health conditions (blood pressure, heart rate, etc.) with publicly available data such as pollution levels. In a typical CPS application, this corresponds to correlating light and accelerometer samples from the user's device with acoustic and air samples collected in the environment. In these examples, raw data sampled locally at the user device and remotely need to be filtered, analyzed and fused together.

[1]The research work presented in this paper was carried out with support from the US National Foundations Grants ECCS-0824120 and CSR 0834493 and was partially funded by the European Commission under the SCAMPI (FP7-FIRE 258414), RECOGNITION (FP7 FET-AWARENESS 257756), and EINS (FP7-FIRE 288021) projects. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Novel contributions of this paper include:

1) A middleware architecture that facilitates application requirements and performs resource management in opportunistic networks.
2) Fault tolerant service composition in opportunistic networks, and
3) An extensive simulation study with synthetic and real traces to investigate the effectiveness of the service composition approach.

II. Related Work

Many challenging problems such as routing, modeling of social interaction and distributed mutual exclusion have been addressed for OppNets. Ott and Kutscher [4] propose protocols for implementing HTTP over delay tolerant networks. There is an extensive literature related to routing in opportunistic networks [1]. For example, Hui et al. [5] use the concepts of community and degree centrality to forward data. Boldrini et al. [6] use a context-aware algorithm, that learns users' behavior and social relations, to forward data in OppNets. Passarella et al. [7], [8] develop an analytical model to study the behaviors of service seeking (seekers) and service providing nodes (providers) that spawn and execute service requests, respectively. The model considers the case in which seekers can spawn parallel executions on multiple providers for any given request, and determines: (i) the delays at different stages of service provisioning; and (ii) the optimal number of parallel executions to minimize the expected execution time. In [9], Sadiq et al. propose a novel algorithm that derives efficiency and effectiveness by taking into account the service load and location of devices providing the services, as well as intermittent connectivity, to select a particular service set. Though many past research works have proposed middlewares for particular problems in opportunistic networks, the concept of service composition is relatively new and no generic middleware support exists.

Service composition has previously been investigated for well-connected MANET environments. Kalasapur et al. [10] propose a service composition protocol for pervasive environments with underlying MANETs. Due to lack of hierarchy, as imposed by the protocol described in [10], such service composition scheme is not suitable for opportunistic networks. Chakraborty et al. [11] propose a middleware for service composition in semi-structured MANETs. This scheme is unsuitable for OppNets as it fails in the absence of continuous connectivity.

III. Proposed Middleware

Figure 1 illustrates components of the proposed middleware. It is be noted that the figure is a projection of a comprehensive middleware architecture. Devices may have only a subset of the modules shown in the figure, depending on application and available resources. The proposed middleware is modular, the processing unit may vary from basic sensing and communication abilities in a sensor to a full set of software suite. For example, a mote sensor might have only an ID and sensory data (e.g., temperature).

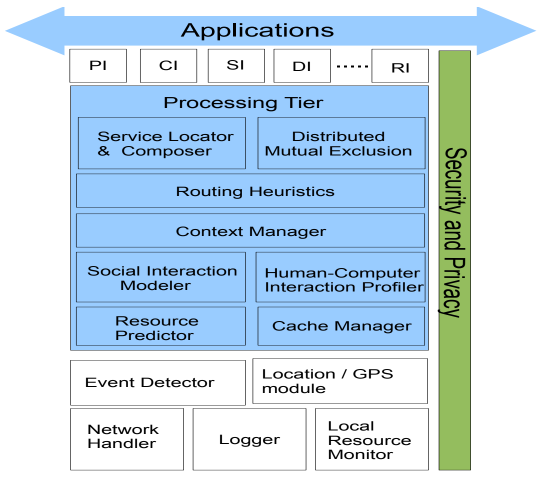


Figure 1. Middleware Architecture

The network handler module performs the actual transmission and reception of messages using one or more types of wireless communication capabilities such as Bluetooth, 802.11. This layer also supports such specifications as Wi-Fi Alliance's Miracast [12]. The logger module is a utility module that can be used by other modules to record events. The resource monitor module periodically checks and records resource levels of the device. Information such as residual battery, CPU utilization, memory utilization are useful to the modules in the processing and application tiers. The event detector module performs sensing of environment, for example, change in light intensity, motion sensing. The event detector receives context rules from the context manager module. Whenever a specified context rule is satisfied, the event detector module informs the context manager module. The location module incorporates facilities such as location prediction using GPS, received signal strength, etc.

The processing tier is responsible for processing various kinds of information received from the device, the applications running on the device and the devices that come in contact opportunistically. The processing tier is composed of multiple modules as shown in Figure 1. The cache manager enables information caching and can be used by other modules. The resource predictor module implements algorithms to predict the resource levels at a future time instant. Such predictions are useful for knowing whether a device will successfully complete a given service. This module also provides the expected life of the device, under device's current usage pattern and resource levels. The social interaction modeler defines rules about social standing of the device (or the owner of the device). It defines metrics such as popularity of a device, contributions made by the device to other devices in the network and communities that a device is part of. Social interaction modeler uses the event detector module to know the ongoing events. For example, BubbleRap [5] uses such social interaction data to perform informed routing in delay tolerant networks. The Human-Computer Interaction (HCI) module is used to record interactions between a user and his device. The routing module implements various algorithms for routing of messages in the OppNet. This module utilizes knowledge gained by the context modeler, social interaction modeler and event detector. The routing module consists of routing algorithms, for example, Hibop [6]. Users or devices may need exclusive access to shared resources in the OppNet. Hence the middleware needs to provide mechanisms for enabling mutual exclusion. The distributed mutual exclusion module implements algorithms such as MEOP [13] for providing mutual exclusion. Security and privacy module can be used to implement security and privacy mechanisms. The social interaction modeler allows a user to set information access control based on privacy settings.

Each device has a set of optional index buffers. The personal information index (PI) contains basic user and device information such as the ID of the device, and name, address, work, home information of the user. The content index (CI) indicates the set of information objects the device is willing to share with other users/applications. The service index (SI) contains information about services discovered by the service locator and composer module. This module also adds the most frequently used service compositions to the service index. The Device Index (DI) is used to store information about the communication cost and inter contact times between the devices. The process of creating and updating DI is explained in detail in Section IV. The reputation index (RI) contains information about reputation and trust values of devices in the network. The Index buffers are optional, varying from a simple source of sensory information to complex graph based indices for service composition.

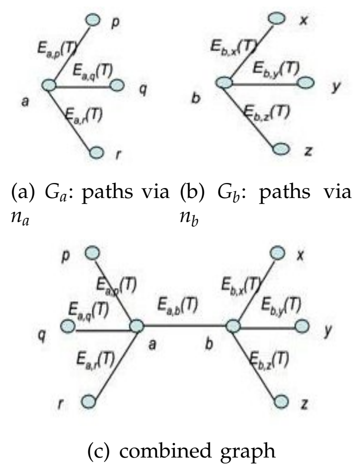
A. Middleware Operations

When two devices come within communication range, they can exchange information during the contact. This exchange is divided into two phases. The first phase consists of a handshake, during which the two devices exchange basic information such as the identity of the device/user and device information (e.g., Nokia 3362 cell phone, 2 MB memory, camera equipped, Bluetooth capable) contained in the PI buffer. The information exchanged during handshake is subjected to privacy settings. When a user comes within communication range of friends, family or other trusted people, more information is made visible to the devices. The second phase starts after the end of handshake. During this phase, the processing tier within the user's device makes a decision on whether it needs to request more detailed information such as the content index (CI), service index (SI) from the other device. Due to limited cache and index space, the user's device might remove information that was least recently used or was received long time back. If the user's device had previously met the other device and has all the previously received information, only updates to the stored information are exchanged.

As an example of usage of middleware modules, we consider the process of service composition. The service composer module can interact with the security and privacy module to gain information about trustability of the devices involved in each of the service composition path. Paths that have untrustworthy devices may not be considered. Interaction with the social interaction module enables scenarios where services hosted on devices within a social group are considered as preferable. The routing module provides the intercontact time details and the expected delay and route for transmission of data from one device to another. The cache manager provides information such as previously used compositions. If a generated composite service contains services that are available locally, the service composition module interacts with the HCI profiler and the resource predictor modules to know the probability that the specified service will execute successfully.

IV. Building the Network Graph

DI contains information that allows nodes to estimate when other nodes will be encountered, and thus the time to access service components. Consider a graph representation of the network where each device is denoted by a node and the contact between two devices (or nodes) is represented by a link or edge between the two nodes. The edge weight represents the expected time of contact between the two nodes. The edge weight can also represent other parameters such as the total contact duration, commonality (e.g: belong to same school), privacy level, or a combination of parameters. Without loss of generality, we use the expected intercontact time as edge weight in the rest of this paper. We represent the pairwise contact between two nodes n_a and n_b by $n_a \Leftrightarrow n_b$, and the expected time of contact by $E_{a,b}(T)$.



(a) G_a: paths via n_a (b) G_b: paths via n_b

(c) combined graph

Figure 2. Sample graphs at nodes n_a and n_b created during pairwise contacts

Consider node n_a, where $a \in (1, 2, ..., M)$ and M is the total number of nodes in the environment. In this paper, we use the terms device and nodes interchangeably. Suppose a device n_a is expected to connect to only a subset of the devices in the network. This set of devices is called as a connectivity set M_a of device n_a. Hence for each device in M_a, the corresponding node has an edge to n_a. The connectivity set M_a indicates the set of nodes that can reach each other by using n_a as the intermediate node. Each node stores a network graph in its DI. The network graph initially indicates the connectivity set of the corresponding device. A sample network graph G_a for n_a is shown in Figure 2(a). In Figure 2(a), n_a and n_p meet with expected time $E_{a,p}(T)$ and n_a and n_r meet with expected time $E_{a,r}(T)$, therefore the expected time for an application on n_r to access a resource on n_p is given by $[E_{a,p}(T)+E_{a,r}(T)]$. A sample of a similar graph G_b for node n_b is shown in Figure 2(b). When two nodes n_a and n_b meet, they can exchange each others graphs stored in the Device Index (DI). As shown in Figure 2(c), the network graphs on both the devices are updated, by merging the corresponding edges of G_a and G_b. The updated graph shows all the paths from n_i to n_j, that use n_a and n_b as intermediate nodes. This combined graph is stored into the DI of both devices. After the network graph is updated, n_a and n_b exchange the updated graph with other devices.

V. Service Composition

Each device hosts certain services and maintains a service graph of the locally hosted services. This service graph is stored in the service index (SI). When two devices meet opportunistically, they exchange their service graphs and create a composite service graph which has information about locally as well as remotely available services. Let $G_x(s)$ and $T_x(s)$ represent the graph of available services and graph of services requested at node n_x. When n_a and n_b meet, they exchange their respective services and an aggregated graph $G(s)$ is created at each node. While the two nodes are in contact, collaborative services are composed and executed using the basic services represented in $G(s)$.

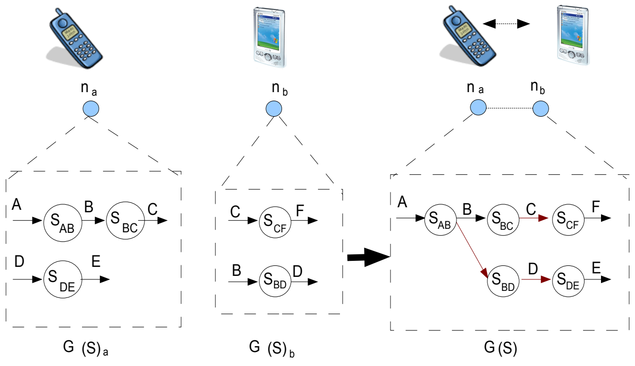


Figure 3. Service Composition

In the example of Figure 3, we consider a task model similar to that used in [10]. There are 26 types of input/output formats, $A, B, C, ..., Z$. Service S_{AB} converts input of format A to output of format B. n_a is required to compose a service to meet the task request $T_a(s) : A \rightarrow F$. Using the aggregated graph $G(s)$, n_a identifies the path from A to F by using services S_{AB}, and S_{BC} on itself and S_{CF} on n_b. In order to execute the composed service, the output parameter C of service S_{BC} is passed on as input parameter to service S_{CF}, on n_b. The output parameter F of S_{CF} is passed from n_b to n_a. Likewise, suppose n_b desires to complete a task $T_b(s) : B \rightarrow E$. The graph $G(s)$ is available at both n_a and n_b. Since $T_b(s)$ is generated on n_b, n_b performs service composition and concatenates S_{BD} with S_{DE}. n_b then locally executes S_{BD}. Since S_{DE} is available on n_a, n_b sends the intermediate result of type D to n_a and invokes S_{DE} on n_a. After completion of S_{DE}, n_a sends the final result of type E to n_b. Thus results are passed on from one service to another across opportunistic connections. While performing service composition there might be multiple sequences of services (and the mapping of services to devices). Metrics such as minimum delay or minimum number of services involved or minimum number of devices involved can be used to select the best sequence.

In general, the selected option will depend on a number of parameters, including trustworthiness, social relationships, etc. The middleware can obtain such information from the dedicated modules presented before. It is out of the scope of this paper to analyze these possibilities in detail, as we prefer to focus on simpler composition mechanisms. In particular, if there are multiple possible paths from the input to the required output, the path that has minimum number of services linked together can be chosen. This selection of path (path with minimum number of services) is henceforth referred to as Minimum Length Composition (MLC). Alternatively, to make the path selection sensitive to delay, we may compute the sum of execution times and intercontact times between consecutive services in the compositions. The path with the lowest sum is then chosen. This approach of selecting a composition with the lowest sum of expected delays, is henceforth called as Minimized Delay Composition (MDC).

VI. Fault Tolerance

Devices in OppNets are resource constrained, personal and mobile. Execution of services on remote devices may suffer from device and communication failures. Hence it is important that the middleware provides fault tolerant service composition. Let $P_s(n_j)$ be the success probability of the j^{th} device in the composite service, that is, the j^{th} device in the composition completes its corresponding service and transfers the result to the next device in the composite service. The entire composition succeeds when all of the involved devices succeed. Hence success probability of composite service C_i is given by:

$$P_s(C_i) = \prod_{j=1}^{k} P_s(n_j) \tag{1}$$

where k is the number of devices in the composition. The service composer sorts all possible compositions in descending order of the preference and selects the best path. Since success probability of the best path might be less than 1, the service composer executes more composition paths in parallel. Let r be the number of independent composition paths executed. The probability $P_s(C)$ that at least one composition will succeed is given by:

$$P_s(C) = 1 - \prod_{i=1}^{r} 1 - P_s(C_i) \tag{2}$$

The middleware selects best r paths such that $P_s(C)$ is sufficiently high and the r paths do not have any device in common. Thus fault tolerance is achieved by replicating service executions.

We now consider the case where the r paths are not distinct, that is, a device is used in more than one composition. Suppose there is a request for composing service $A \rightarrow D$. Let S^a_{AB} denote that service S_{AB} is hosted on device n_a, S^b_{BC} denote that S_{BC} is hosted on n_b and service S^c_{CD} and S^d_{CD} denote that S_{CD} is hosted on two devices: n_c and n_d. Assume that $S^a_{AB} \rightarrow S^b_{BC} \rightarrow S^c_{CD}$ and $S^a_{AB} \rightarrow S^b_{BC} \rightarrow S^d_{CD}$ are the only two possible compositions for the request $A \rightarrow D$. Since only S_{CD} is replicated on two devices, a failure of n_a or n_b will result in failure of both the paths. The replicated composite service fails if either n_a fails during execution of S_{AB} or n_b fails during execution of S_{BC} (provided n_a has succeeded) or both n_c and n_d fail (provided n_a and n_b have succeeded). Success probability of the replicated composite service is given by:

$$P_s(C) = P_s(S^a_{AB}).P_s(S^b_{BC}).\{1 - \left(1 - P_s(S^c_{CD})\right)\left(1 - P_s(S^d_{CD})\right)\}$$

Hence to know the probability that at least one service composition will succeed, first we create a combined graph of the set of compositions. If there are any parallel execution sequences in the graph, the probability of success over those parallel portions is computed using (2). The parallel portions can then be replaced by a single dummy service whose success probability is equal to that computed for the parallel portion. The graph is thus reduced to a single chain (without any parallelism) of service executions. For this single chain, (1) is used.

VII. Simulation Studies

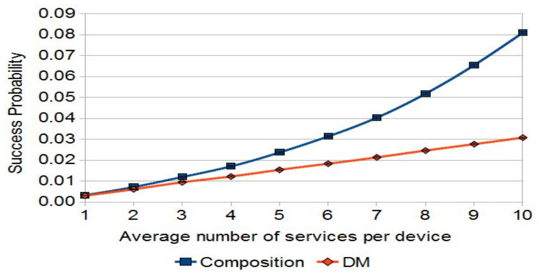
To study service composition in OppNets, we use the Imote/Cambridge/Haggle [14] real world trace. This dataset includes Bluetooth sightings made by 12 students carrying iMotes for six days. In addition, to further validate our approach, we also use Random Waypoint (RWP) mobility model. Modifications proposed in [15] are used to ensure that the simulations are in steady state. The simulation consists of variable number of nodes moving in an area of 300m x 300m. We consider RWP to vary the number of nodes and study the effect of changes in cardinality on service completion. Details of number of nodes is provided along with each simulation. The average speed is 10m per minute and average communication range of devices is 10m. Each node comes in contact with at most one other node at any point in time. Time required to execute a service on a node is exponentially distributed and has average value of 30 seconds. Note that, although RWP is not representative of all kinds of opportunistic networks, it is typically used to reproduce social mobility patterns where users belong to the same social community [16]. Confidence intervals have been computed with 95% confidence level. In several plots, the confidence intervals are extremely small and hence are not visible.

We consider the task model similar to that used in [10]. There are 26 types of input/output formats, $A, B, C, ..., Z$. Hence there can be a maximum of 650 services. Each device hosts a randomly chosen set of services. For example, to host an average of 5 services per device with 10 devices, the following steps are performed: (1) To ensure that there are on average 5 services per device, a *God* program generates 10 numbers such that the average is 5. Each device gets exactly one of these numbers. (2) Each device will host exactly the number, say H_j, of services as provided by the *God* program. (3) For each device the following steps are repeated till the device selects H_j services: For each service: a random number between 0 and 1 (inclusive) is generated. If this random value is less than $(H_j/650)$, the corresponding device hosts the service. There might be multiple service providers for a particular service. When two nodes opportunistically come in contact and remain in contact of sufficiently long time, the exchange of service graphs is performed successfully. The nodes then update their corresponding composite service graphs. Each node periodically generates random service requests. The average time interval between generation of two service requests on any particular device was 1 minute. When a client node wishes to convert data from one format to another, it first checks if such a conversion is possible locally. If not, the service graph is traversed to find a path from input format to output format.

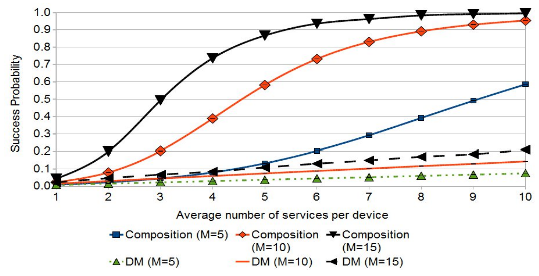
A. Success Probability

We compare MLC approach to a Direct-Match (DM) approach where a single service that provides the required conversion is used. In DM, services are not linked into a composition. Hence if there are few services available in the network, a data format transformation request might not succeed.

Figure 4 shows the probability of finding a transformation between the requested formats. The number of devices M in the network is chosen from $\{2, 5, 10, 15\}$. The simulation for RWP is divided into two phases. In the first phase, devices spread their composite service graphs to other devices in the network. The first phase is executed till each device receives complete knowledge of all services in the network. Hence the first phase requires variable amount of time to complete. The second phase is started only after completion of the first phase. In the second phase, devices generate service execution requests. If a service is not available anywhere in the network, the device on which the request was generated performs service composition. The second phase is executed for 40000 seconds after the end of first phase. In MLC, as the number of services provided by each node increases, the probability of finding a composite service increases exponentially, whereas that in DM increases linearly. The probability of successfully finding a composite service converges



(a) $M = 2$



(b) $M = 5, 10, 15$

Figure 4. Probability of successfully finding a transformation between the requested formats (RWP)

to 1 faster as the number of devices in the system is increased. For DM, the probability of finding the required service also increases with the number of devices in the system.

B. Length of Composition

Figure 5 shows the probabilities that the service composition will be of a particular length. Since DM always uses 1 service, the trend for DM is not shown. Each device hosted an average of 10 services.

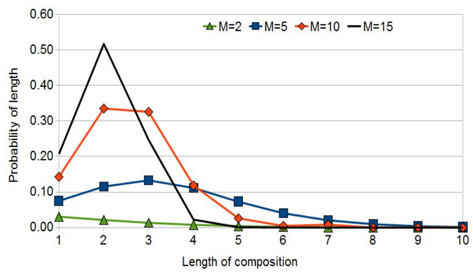


Figure 5. Probability of length of composition (RWP)

The number of devices M in the network is chosen from $\{2, 5, 10, 15\}$. We can see that as the number of devices increases, the probability of finding shorter length composition increases. For all the trends, the summation of probabilities of each length gives the success probability, whose value is same as that in corresponding subfigures of Figure 4.

MLC selects compositions that have the minimum number of services and does not consider the intercontact times between nodes. Hence, in an opportunistic network, MLC might not provide minimum delay in executing the composite service. As the number of services per device increases, initially the success probability is less than 1. During this, the length of composition shows an increasing trend. When success probability reaches 1, length of composition starts decreasing with higher number of services per device.

C. *Time and number of devices required to complete service execution*

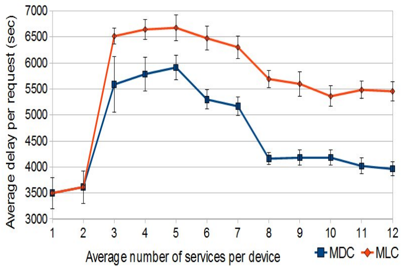
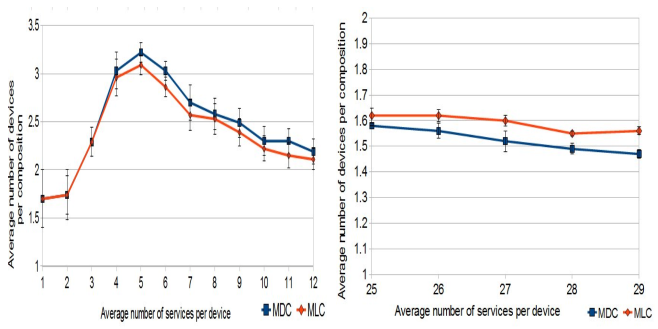


Figure 6. Average delay in completing the composite service (RWP)



(a) 1 to 12 number of services per device (b) 25 to 29 number of services per device

Figure 7. Average number of devices in composition (RWP)

For Figures 6 and 7, $M = 10$. Each reading is an average of 50 requests, shown with 95% confidence interval. Only those cases where either composite services or direct match were found are considered. Figure 6 shows the average delay per request. Since MLC selects paths that have minimum composition length, it has a higher average delay. Figure 7 shows the number of devices used in order to satisfy the request. MLC and MDC perform composition whereas DM does not. Thus DM always uses at most one device to execute the service, whereas MLC and MDC might require multiple devices. Hence the trend for DM is not shown. Figure 7 shows a non-intuitive result that even though MDC reduces the average delay per request, the number of devices used in the composite service is almost same as that in MLC. When the number of services per device increases, MDC tends to reduce the delay due to intercontact time by selecting more services on a device. Though this increases the number of services in the composite service, the number of devices selected by MDC tends to be almost same as than that selected by MLC.

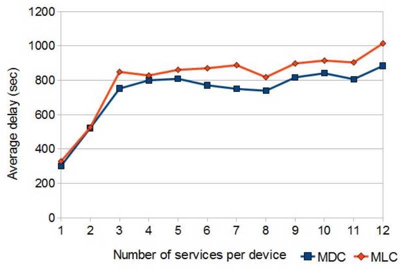


Figure 8. Average delay in completing the composite service (using Haggle dataset)

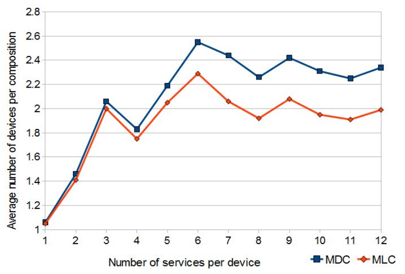


Figure 9. Average number of devices in composition (using Haggle)

Figure 8 shows the average delay in executing a composite service. Figure 9 shows the average number of devices involved in each composition. The devices use the Haggle [14] real world trace to simulate the mobility. This dataset includes Bluetooth sightings made by 12 students carrying iMotes for six days. Since the trace includes timestamps of Bluetooth sightings, establishing connections between devices at the corresponding times ensures that the mobility pattern is followed correctly. From Figures 6 and 8 we see that MDC has lower average delay overhead than MLC. From Figures 7 and 9, we see that MLC and MDC require almost equal number of devices per composition.

D. *Fault Tolerance*

Figure 10 shows the success probability of service executions with and without using fault tolerance. There were 10 devices in the network and on average 5 services per device. For replication, we need more than one service composition paths between a given input and required output. From Figure 4(b), we see that the probability of finding at least one service composition

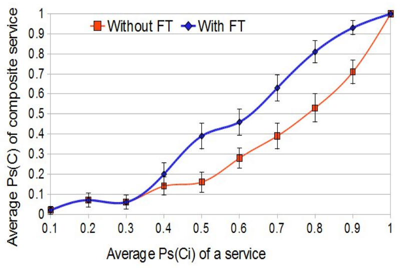


Figure 10. Probability of success with and without fault tolerance (RWP)

is around 0.6. Hence the probability of finding more than one path will be less than 0.6. Hence in Figure 10, when the success probability of individual service is less, the two trends have almost equal values. When the success probability of individual service increases, the probability of success with replication is higher. When the success probability of individual services becomes 1, there are no failures, and hence the success probabilities reach 1.

VIII. Conclusion and future work

In this paper, we have developed a modular middleware architecture for opportunistic computing. The paper also presents service composition in opportunistic networks. Since the devices and service executions are prone to failure, the middleware performs fault tolerant service composition by the means of replication. The work presented in the paper demonstrates that application oriented service composition can be effectively performed in opportunistic environments. The concepts developed in this paper can be employed in such CPS pervasive applications as e-health, traffic management, environment monitoring and sustainability, and others. In the future we will investigate incorporation of context information into the service composition algorithm.

References

[1] L. Pelusi, A. Passarella, and M. Conti, “Opportunistic networking: data forwarding in disconnected mobile ad hoc networks,” *IEEE Communications Magazine*, vol. 44, no. 11, pp. 134–141, Nov. 2006.

[2] M. Conti and M. Kumar, “Opportunities in opportunistic computing,” *Computer*, vol. 43, no. 1, pp. 42–50, Jan. 2010.

[3] M. Conti, S. Das, C. Bisdikian, M. Kumar, L. Ni, A. Passarella, G. Roussos, G. Trster, G. Tsudik, and F. Zambonelli, “Looking ahead in pervasive computing: Challenges and opportunities in the era of cyberphysical convergence,” *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 2–21, February 2012.

[4] J. Ott and D. Kutscher, “Bundling the web: Http over dtn,” in *Workshop on Networking in Public Transport (WNEPT)*, Aug. 2006.

[5] P. Hui, J. Crowcroft, and E. Yoneki, “Bubble rap: social-based forwarding in delay tolerant networks,” in *International Symposium on Mobile Ad Hoc Networking and Computing*, 2008, pp. 241–250.

[6] C. Boldrini, M. Conti, and A. Passarella, “Exploiting users social relations to forward data in opportunistic networks: The hibop solution,” *Pervasive and Mobile Computing*, vol. 4, no. 5, pp. 633–657, October 2008.

[7] A. Passarella, M. Conti, E. Borgia, and M. Kumar, “Performance evaluation of service execution in opportunistic computing,” in *ACM MSWiM*, October 2010, pp. 17–21.

[8] A. Passarella, M. Kumar, M. Conti, and E. Borgia, “Minimum-delay service provisioning in opportunistic networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1267–1275, August 2011.

[9] U. Sadiq, M. Kumar, A. Passarella, and M. Conti, “Modeling and simulation of service composition in opportunistic networks,” in *ACM MSWiM*, October 2011.

[10] S. Kalasapur, M. Kumar, and B. A. Shirazi, “Dynamic service composition in pervasive computing,” *Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 907–918, July 2007.

[11] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, “Service composition for mobile environments,” *Mobile Networks and Applications*, vol. 10, no. 4, pp. 435–451, Aug. 2005.

[12] WiFi-Alliance, “Miracast,” *http://www.wi-fi.org/media/press-releases/wi-fi-alliance-launch-wi-fi-certified-miracast-deliver-display-applications*, May 2012.

[13] S. Tamhane and M. Kumar, “Token based algorithm for supporting mutual exclusion in opportunistic networks,” in *International Workshop on Mobile Opportunistic Networking (MobiOpp)*, Feb 2010.

[14] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, “CRAWDAD trace cambridge/haggle/imote/cambridge (v. 2006-01-31),” Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote/cambridge, Jan. 2006.

[15] W. Navidi, T. Camp, and N. Bauer, “Improving the accuracy of random waypoint simulations through steady-state initialization,” in *15th International Conference on Modeling and Simulation*, March 2004, pp. 319–326.

[16] C. Boldrini, M. Conti, and A. Passarella, “Users mobility models for opportunistic networks: the role of physical locations,” in *In Proc. IEEE Wireless Rural and Emergency Communications Conference (WRECOM 2007)*, October 2007, pp. 1–6.