An Autonomous Middleware Model for Essential Services in Distributed Mobile Applications

Marcio E.F. Maia, Lincoln S. Rocha,
Paulo Henrique M. Maia, and Rossana M.C. Andrade

Group of Computer Networks, Software Engineering and Systems
Federal University of Ceara. Av. Mister Hull, s/n - Campus do Pici, Bloco 942-A,
CEP: 60455-760, Fortaleza, CE, Brasil
{marcio,lincoln,paulomaia,rossana}@great.ufc.br

Abstract. The evolution and popularization of mobile devices and wireless networks give rise to the creation of a new interaction paradigm, where the devices cooperate to execute short tasks. In this scenario, the problem of how to handle environment changes, which may increase the complexity of distributed mobile applications management and maintenance, needs to be addressed. This paper presents an autonomous and evolutionary model to permit a prompt adaptation of essential services (i.e. message exchange, service description service discovery, service coordination, mobility support and security) to context changes. To validate it, a mathematical model describing the time complexity to diffuse an efficient implementation of an essential service (strategy) taking into account the number of devices is proposed. Finally, the diffusion approach is implemented in a simulator to reason about its impact on the overall efficiency of the essential services and, consequently, the performance of the application.

Keywords: Mobile Middleware, SOA, Autonomic Computing.

1 Introduction

The popularization of mobile devices and wireless communication technologies has influenced the architectural design of networks and applications. Networks evolved from centralized, static and cable-based to mobile and wireless [1]. In addition, the communication between mobile applications that use different network interfaces simultaneously has become common.

Although it increases flexibility and robustness, it can make the development and management of these applications more complicated. The core of this problem is that application-level protocols are designed considering a limited subset of information from the executing environment. While protocols designed to architectures like Wi-Fi and cellular networks may use centralized approaches, protocols developed to ad-hoc networks are fundamentally decentralized [3].

Throughout this paper, application-level protocols will be called *essential services*. They are key services used to create service-oriented mobile applications, such as message exchange, mobility support, service description, service discovery, service coordination and security. For instance, a chat application may require that message exchange, service description and discovery are available and a museum guide running on a PDA might also need service coordination.

To implement essential services, application developers can rely on abstractions provided by middlewares [2]. Here, these implementations are called *strategies*. For example, remote method invocation and tuple-based are two strategies for implementing message exchange. The strategy to be used in the application is defined at design time and coded using middleware libraries. Although it may reduce development time and complexity, it may also produce a rigid design that is unable to cope efficiently with changes in the execution environment.

Changes in the execution environment occur in two levels: network and application. An example of the former is the use of a different network interface or variation in the communication latency. Changes in the latter may be the unavailability of a centralized server or the use of a different language to describe services. The consequences of binding the application and middleware at design-time may vary from loss in performance to application crashing.

This paper presents *AMESMA*, an autonomous and evolutionary middleware model for essential services in distributed mobile applications. It promptly selects the best strategy (most efficient) for an essential service according to changes in the environment. Here, efficiency is described by a quality of service (QoS) descriptor, which depends on the application and essential service. For instance, service discovery efficiency can be measured by the number of services discovered or the average time to discover one service. Thus, by autonomously choosing the most efficient strategies, the model tries to improve the overall performance of the essential services. It also permits new strategies to be incorporated on-the-fly, with a minimum effort to maintain the essential services.

The benefits of this approach are threefold: firstly, strategies for an essential service can be replaced more easily, according to their efficiency. Secondly, an autonomous layer, placed between the applications and the strategies, rapidly defines which strategy should be used when changes in the environment are detected. Thirdly, instances of the model running on different devices cooperate to define efficient strategies. As devices interact and the efficient strategies are identified, these strategies are diffused throughout the network. After a short period of time, the strategies being used should converge to one or a small set of efficient strategies, increasing the overall performance of the essential services.

AMESMA was evaluated according to an analytical model that described the time to diffuse one strategy to all devices. That time is affected by the probability of diffusing an strategy (fanout factor). A higher fanout factor means a faster diffusion, but at a higher cost. The analytical model described how to adapt this factor, minimizing the cost on the network with an acceptable diffusion time. The second evaluation investigated in a simulator how the overall performance of the essential services varied as efficient strategies are diffused.

The rest of this paper is organized as follows. Section 2 gives a brief overview on the background of essential services. Section 3 details the AMESMA model and an analytical model used to describe the strategy diffusion, while section 4 presents some simulation results. Section 5 compares our approach to some related work and, finally, section 6 presents the conclusions and future work.

2 Essential Services for Mobile Applications

Service orientation (SOA) permits applications to be decomposed in atomic parts called services [6], facilitating their deployment and management. SOA-based applications require mechanisms to describe, find, access and compose these parts in a secure, fault-tolerant and context-aware manner. Here, these mechanisms are called essential services. Due to space restrictions, this paper only describes service discovery strategies, since this is the essential service implemented to validate the proposed model.

2.1 Service Discovery

Service discovery allows services to be discovered and accessed when available, permitting the creation of loosely-coupled and robust applications, since service information is accessed at runtime. Service discovery strategies can be divided according to how service information is published and how it is stored [3].

There are two strategies to publish a service: push-based, where information is published when the service becomes available, and pull-based, which publish information when requested. The former has a lower discovery latency and a higher cost on the network. The latter has a higher latency and a lower cost.

How service information is stored refers to the number and location of service registries and can be implemented by three strategies: centralized, totally distributed and hybrid. Centralized strategies publishes information in a single registry. It is easy to implement and less resilient to failures. In totally distributed strategies, every device is a registry, which increase failure resilience. In hybrid strategies, few devices act as service registries. This strategy is harder to implement than the other two strategies, but has a lower cost on the network than the distributed strategy and is more resilient to failures than the centralized strategy.

3 AMESMA

AMESMA is an autonomous and evolutionary middleware model to increase the performance of mobile applications by autonomously monitoring and searching for more efficient strategies. The main characteristic of the model is try-and-error, from evolutionary computing, in which new solutions are generated and tried in the environment and efficient solutions out live inefficient ones. The goal is to allow a rapid adaptation of the essential services to momentary conditions of the execution environment, along with minimizing coupling between application and middleware, facilitating the maintenance of the essential services.

Different strategies for an essential service are monitored at runtime and dynamically chosen based on information collected from the application layer. To use an essential service, an application informs its *quality of service (QoS) descriptor*, a particular metric relevant to that application. In a service discovery essential service, the QoS descriptor may be the number of services discovered or the average number of sent messages necessary to discover one service.

When new strategies are discovered in a nearby device, the efficiency level of the remote strategies is compared to the efficiency level of the equivalent local strategies. More efficient remote strategies are deployed locally. Strategy monitoring and substitution is performed transparently to the applications. When an application requires the execution of an essential service, it invokes an interface associated to that service. The actual strategy used is defined by the model using the QoS descriptor informed by the application and the efficiency values of the strategies. The most efficiency strategy is used to invoke this essential service.

3.1 Middleware Architecture

Figure 1 divides the architecture of the middleware model into essential service interfaces, which are packed with the application at compile-time, and middleware core, responsible for managing the essential services, i.e., the strategy selection and evolution. The middleware core is deployed in each device and is divided into QoS and context management and Middleware management. We discuss theses modules of the architecture in more details below.

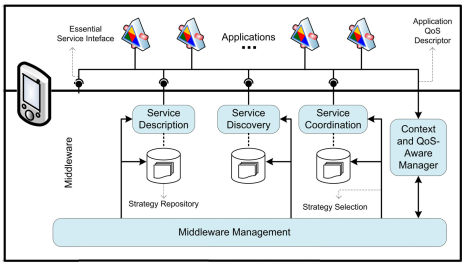


Fig. 1. Middleware architecture

Essential Services Interfaces. Applications invoke the essential services using an interface for each service, compiled with the application. Strategy selection is performed by the middleware management module based on QoS descriptor and context information received from the QoS and context module.

This separation between interface and execution is possible if different strategies are accessed equally. Considering the service discovery essential service, regardless whether the strategies and centralized, distributed or hybrid, they have similar service discovery messages. The fields in this message can be summurized in 1) ID, that uniquely identifies a device/application; 2) available or required resources; 3) service description, that contains information about a service (*Pull* model) or search parameters (*Push* model); 4) Time to Live, number of times a message can be retransmitted and 5) amount of time message remains valid.

Different strategies from devices distributed in the network communicate using the service discovery message. Amongst its fields, the service description is application-dependent. Instead of standardizing service description to guarantee interoperability, it should be defined by each application to promote flexibility.

QoS and Context Manager Module. This module manages QoS requirements from the application. When an application invokes an essential service, it informs its QoS descriptor for that service. That descriptor helps to define which strategy should be picked from a local strategy repository and to compare two strategies to define the most efficient.

This module stores information about strategy efficiency in different contexts. Upon context change, it informs the middleware management module of this new context. Based on context information received from the QoS and context manager, the middleware management module decides whether or not to change the strategy being used.

Context information may be acquired by different sources. It can be accessed using sensors presented in the device itself or from context provision services present in the environment.

Middleware Management Module. Autonomy and evolution are implemented in the middleware management module. It autonomously monitors, detects and diffuses efficient strategies and receives essential service invocation. Upon receiving an invocation, the middleware management module access the QoS and context management module to define which strategy is the most efficient at that specific moment. Applications only invoke an interface for an essential service and are unaware of which strategy will be used. Additionally, evolution is accomplished by diffusing efficient strategies and minimizing the presence of inefficient ones.

3.2 Middleware Internals

The mechanisms executed by the middleware are: essential service invocation and strategy diffusion. To invoke an essential service, an application informs its QoS descriptor, its ID and the parameters of the required essential service. Since this invocation is unblocking, the application can continue its execution and access the *result* later, using an object returned to the application. That invocation triggers an event to the middleware management module, which defines the correct strategy to be executed using the QoS descriptor and context information, accesses the strategy repository and executes the selected strategy.

The QoS descriptor is also important in a distributed execution of a strategy. For instance, a service discovery based on a distributed strategy is executed by numerous devices throughout the network. The service discovery message carries the QoS descriptor, which is used to select the correct strategy. This approach allows all devices to struggle to provide a higher efficiency according to the required QoS metric.

Simultaneously to the essential service invocation, the middleware management module periodically contacts a subset of its neighbors searching for more efficient strategies. If efficient strategies are found, a compatibility test is performed to define whether that remote strategy can be deployed locally.

The number of neighbors chosen to compare the strategies is called *fanout factor*. A lower fanout factor means that strategies will be diffused slower, with a lower cost on the network. On the contrary, a higher fanout factor means faster diffusion and higher cost on the network. Therefore, it is relevant to understand how the fanout factor impacts on the strategy diffusion time and cost on the network.

3.3 Strategy Diffusion Analytical Model

The dynamic nature of mobile networks requests quick adaptation of essential services to network conditions. Hence, it is important to understand how long diffusing one strategy takes and how it impacts on the overall efficiency of the essential service. This analytical model is based on a paper published by Groenevelt [13], which analyzes the mean time to diffuse a message to an specific node. However, our goal is to understand when all nodes receive the message. Furthermore, our model allows nodes to leave the network, creating a birth-death model.

Suppose there are N nodes in the network, moving independently according to the random way point mobility model [26], with a limited connection range. Each node has its own strategy $s_i, i \in E$, where E is the set of strategies. Moreover, initially only one node has the most efficient strategy, called s_e. Whenever that node contacts another node, that strategy is diffused. This approach is called *contact-and-infect* or *epidemic* diffusion [25].

Strategy Diffusion Delay. Assume that the system is in state k whenever there are k nodes using strategy s_e. The system starts at state 1 and reaches an absorbing state N. Figure 2 shows a Markov Chain for the states the system can enter. Once the system enters state i, it never returns to state $i-1$, since there is no strategy more efficient than s_e and this model assumes that all nodes do not leave the network during the diffusion process.

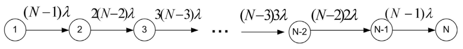


Fig. 2. Strategy diffusion Markov chain

Instead of diffusing the strategies to all nodes it meets, one node chooses a subset of these neighbors to compare their strategies. The fanout factor v represents the percentage of neighbors chosen to compare their strategies.

That approach tries to minimize the number of messages being exchanged without compromising the diffusion time.

Suppose there are i nodes with strategy s_e, then it leaves state i to $i+1$ at a rate $\lambda i(N-i)v$ (I), where λ represents the meeting rate and dependends on the average node speed, mobility pattern, area and antenna range. That meeting rate follows an exponential probability distribution [13]. If S_i represents the total time the process spends in state i, then $S_i = \frac{1}{\lambda i(N-i)v}$.

Let T_S be the time the process reaches state N (diffusion time) , then $E[T_S] = \sum_{i=1}^{N-1} S_i = \sum_{i=1}^{N-1} \frac{1}{\lambda i(N-i)v} = \frac{2}{\lambda v N}[\log(N-1)+\gamma+o(\frac{1}{N-1})]$, where γ is the Euller constant.

Birth and Death Model. The previous model assumed that nodes do not leave or arrive in the network. This restriction is now relaxed by considering that one node leaves the network at a rate μ and another node enters the network at the same rate. It keeps the number of nodes constant but varies the number of nodes using the efficient strategy.

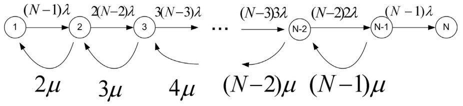


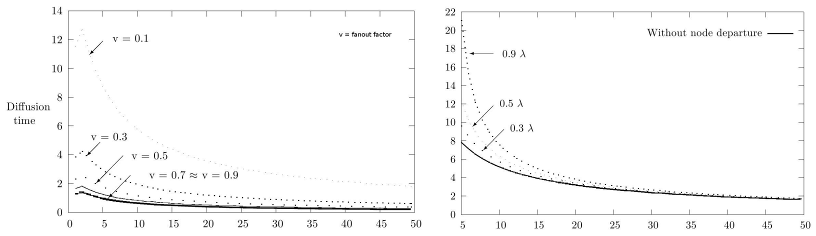
Fig. 3. Strategy diffusion Markov chain with departure

Figure 3 shows the strategy diffusion Markov chain with node departure. The process leaves state i to $i+1$ when an efficient strategy is diffused at a rate $i(N-i)\lambda v$ and from state $i+1$ to i when an node with an efficient strategy leaves the network at a rate $i\mu$. It is important to highlight that this analysis assumes that $\lambda v > \mu$, or otherwise the strategies would not be diffused. From that, the diffusion time expectation is $E[T_J] = \frac{2}{\lambda v N - \mu}[\log(N-1)+\gamma+o(\frac{1}{N-1})]$.

Graphics. The main objective is to understand how the number of neighbors chosen to compare the strategies impact on the strategy diffusion time. Figure 4 shows the strategy diffusion delay for both scenarios, without departure on Figure 4a, and with departure on Figure 4b.

In the graphic without departure, when the number of nodes is less than 20 nodes, a fanout factor of 0.5 has a similar delay as if all neighbors were chosen. When the number of nodes increases, even a fanout of 0.3 behaves similarly as the fanout factor 1.

In Figure 4b, the node departure was considered, and the fanout factor chosen was 0.3. This value was chosen to analyze the impact of the node departure rate, and it presents a good trade-off between diffusion time and cost. When the number of nodes is smaller than 20 nodes, the diffusion time increases considerably and the fanout factor must be increased. However, as the number of nodes raises, the departure rate gradually loses its influence.



(a) (b)

Fig. 4. (a) Strategy delay without departure varying the number of nodes. (b) With departure varying the number of nodes.

The fanout factor may be adapted based on the number of neighbors known at a given moment. According to [15], if d_{avg} is an estimate for the average number of neighbors and n the total number of nodes, then $d_{avg} = \pi \ln n$. Thus, using this estimate and the number of 20 nodes from Figure 4b as an adaptation parameter, the fanout factor can be adapted as follows:

$$v = \begin{cases} 0.5 \text{ If } d_{avg} < 9, \\ 0.3 \text{ Otherwise} \end{cases}$$

This estimate makes assumptions about the node density and mobility pattern. Since the goal is to obtain an approximation to guide the adaptation process instead of knowing the exact number of nodes, and the maximum and minimum number of neighbors is of the same magnitude as the average number of neighbors [15], this estimate gives an acceptable adaptation guide.

4 Simulation Results

AESPmob was implemented in the Jist/Swans [16] network simulator to verify how the strategy diffusion mechanism impacts on the overall performance and cost on the network. Simulation parameters are shown in Tables 1 and 2.

Table 1. Simulation parameters

Parameter	Value
Area	1000m x 1000m
Simulation time	400s
Transmission radius	100m
Mobility pattern	Random Way Point
Minimum velocity	3 km/h
Maximum velocity	5 km/h
Pause time	0s

Table 2. Model parameters

Parameter	Value
Number of devices	20 - 100
Fanout factor	0.2 - 0.8
Strategy verification period	10s
Density of services	10%
Service requisition rate	10 every 15s
Number of simulation	500

The simulation aims to analyze how the overall performance behaves as the strategies are diffused, varying the number of devices and the fanout factor. The number of nodes ranged between 20 and 100, and the fanout factor from 0.2 to 0.8. Each scenario was simulated 500 times, with the average values shown.

Three important concepts about the simulation must be understood. The first one is the service discovery strategies that were implemented. In this work, four strategies have been implemented: a Flooding-based, a probabilistic gossip-based [18], a reliable push-based, called RAPID [19], and a hybrid strategy [3].

The second one is the QoS metric used to evaluate the efficiency of the strategies: 1) *number of services discovered*, where strategies that find more services are diffused, 2) *ratio between number of services discovered and number of messages sent*, where a higher ratio defines the strategies that are diffused and 3) *average latency time*, where a lower latency means a higher efficiency.

The third one is how the efficiency information is collected. The more precise this information is, the higher the probability that more efficient strategies are diffused. The first approach called *local view* decides which strategy is more efficient based only on information locally collected. The second approach is the *cooperation view* and uses a combination of information locally collected with information obtained from devices nearby. While the third approach is called *global view* and the decision about which strategies are more efficient is based on information shared by all devices.

4.1 Overall Efficiency

The overall efficiency of an essential service is the sum of the efficiencies of all strategies. For instance, the overall efficiency of a service discovery essential service is the total number of services discovered in a period of time by all devices. The individual efficiency is the number of services discovered by one strategy in the same period.

The Y-axis in Figure 5 shows how the overall efficiency of the service discovery essential service varies during the execution as the strategies are diffused. The X-axis represents the simulation time.

The global view has a higher chance of making correct decisions. This was confirmed by simulation, since the global view was the most efficient in all scenarios simulated. However, distributed global information is hard to collect and update. Moreover, it usually introduces an overhead that very often prohibits its use. The cooperation view is not as efficient as the global view, but has a considerable lower impact. Decisions based on local information are less reliable, but it has the lowest cost.

Figure 5a and 5b shows the overall efficiency varying as the strategies are diffused according to the QoS metric called number of services found. The overall efficiency of the global view approach increases until the execution time of approximately 160 seconds, when the most efficient strategy is totally diffused and all devices are using the same strategy. After that, the overall efficiency is constant.

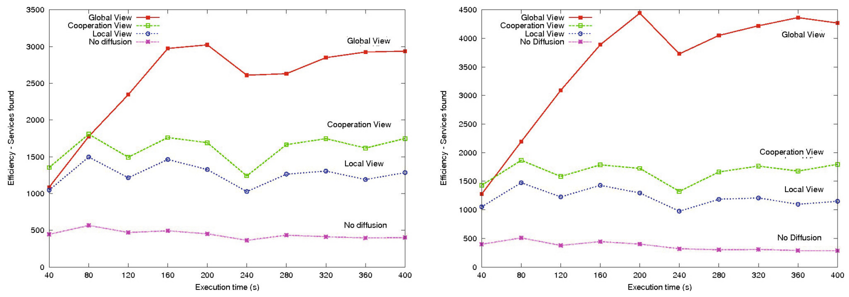
The cooperation and local view increase the efficiency at a lower rate. It happened because they are not as reliable as the global view, and inefficient strategies were diffused. However, despite inefficient strategies may eventually be diffused, the strategies that performed better are diffused after all. These two approaches presented a better performance that the scenario without diffusion.

Figure 6 shows how the ratio between the number of services discovered and messages sent varies as strategies are diffused. All approaches with diffusion presented an efficiency gain. The difference between the Global View and Cooperation View is smaller. This happened because strategies based on flooding and gossiping send more messages than the other two strategies. It makes the value of the ratio to decrease and facilitates the identification of the efficient strategies, which are the other two.

Figure 7 shows the QoS metric that considers the average time to discover one service. As the simulation happens, services requests sent at the beginning of the simulation arrive, which makes the average time to discover one service to increase. The no diffusion approach initially behaves more efficiently, but as time passes, the other three approaches with strategy diffusion behave more efficiently. This happened because initially, even using the global View approach, inefficient strategies were diffused. After a while, the average time started to become constant, or decrease in the case of the global view. This means that the time to discover one service is actually decreasing, since services requests sent at the beginning are still arriving (higher discover time), but the average time is being kept constant, or decreasing a little, which shows again that the overall efficiency is indeed increasing (smaller time to discover one service).

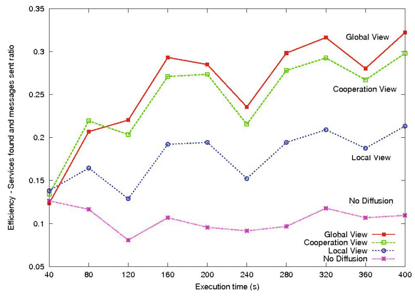
4.2 Diffusion of Efficient Strategies

Initially, the four strategies were randomly assigned to the nodes according to an uniform probability distribution. During simulation, the number of strategies used may increase or decrease according to their efficiency. The goal is to maximize the number of efficient strategies and to minimize inefficient ones.

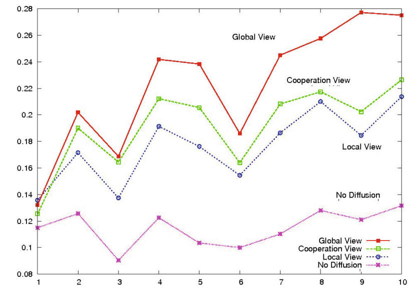


(a) 20 nodes and a fanout factor of 0.4 (b) 60 nodes and a fanout factor of 0.2

Fig. 5. Number of services discovered during the simulation

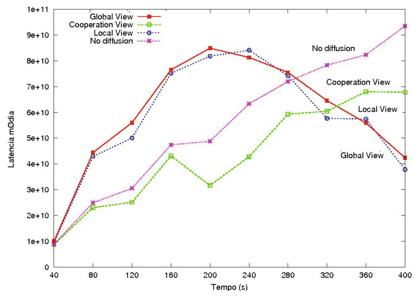


(a) 80 nodes and fanout factor of 0.6

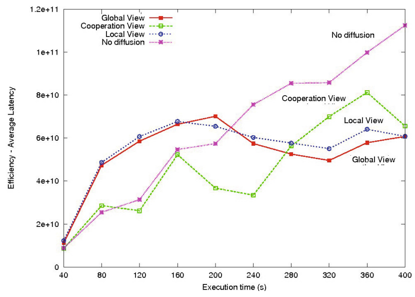


(b) 60 nodes and fanout factor of 0.8

Fig. 6. Ratio between the number of services discovered and messages sent

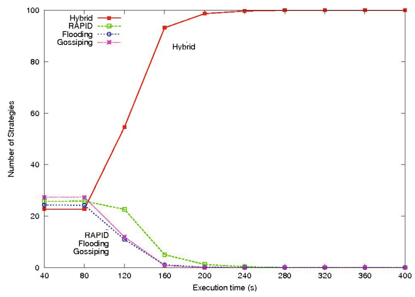


(a) 40 nodes and fanout factor of 0.2

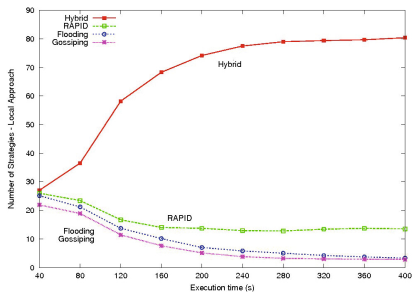


(b) 60 nodes and fanout factor of 0.6

Fig. 7. Average discovery time



(a) Global View



(b) Local View

Fig. 8. Number of each strategy for 100 nodes and a fanout factor of 0.4

Figure 8 shows the number of each strategy during simulation, using the number of services found to guide the diffusion. Figure 8a shows how the number of each strategy varies using the global view and Figure 8b presents the local view. The Hybrid strategy has a better performance than the other strategies, and it is diffused. The global view (Figure 8a) diffuses this strategy faster than the Local view (Figure 8b). Approximately at 160 seconds, all nodes are using the Hybrid strategy in the global view approach and all other strategies are eliminated. While in the local view, the Hybrid strategy is not totally diffused, and there are a few nodes left using the other strategies at the end of simulation.

Analyzing Figure 5, it is interesting to verify that the global view presents a peak in performance at approximately 160 seconds. This is the same time that Figure 8a shows that the Hybrid strategy was diffused and all devices were using the most efficient strategy. Analyzing the local view in Figure 5, the number of services discovered increases until 160 seconds, where it remains constant. Figure 8b shows the number of the Hybrid strategy tending to remain constant after 160 seconds. At the end, strategies that are not the most efficient are still present.

5 Related Work

AdHocWS [22] allows service migration and allocation according to QoS policies defined by the system administrator. Additionally, service migration and usage occurs according to benefit functions. These functions consider environment conditions to decide which service instance is more appropriate at that moment. The difference is that while the AdHocWS considers application services, AMESMA is concerned about the efficiency and evolution of essential services.

Kramer and Maggee [23] propose an architectural model composed of three layers for self-managed systems: 1) goal management considers high-level goal specification and system state to generate a plan; 2) change management adapts the system architecture in response to change in the plans and 3) component control, formed by the system components providing state information to the higher layers. This model proposed by Kramer and Maggee can be used to implement the middleware management layer of AMESMA.

SLACER [24] self-organizes its nodes into an artificial high-cooperative P2P social network. SLACER has the same mechanism of service evaluation and evolution. However, while it is concentrated in P2P services, the interest of AMESMA is on essential mobile services. AMESMA also permits application QoS description and service execution according to Qos restrictions.

6 Conclusion and Future Work

This paper presented AMESMA, an autonomous and evolutionary model to adapt essential services on mobile applications. It consists of a middleware layer where different strategies are selected according to their efficiency, increasing the probability that the most efficient strategies are used by the applications. Our major contribution is the improvement in the overall performance of mobile applications. In addition, our model eases the maintenance and evolution of the essential services, since new strategies can be automatically found and deployed.

A mathematical model investigated the time it takes to diffuse one strategy. That model showed how to minimize the cost on the network inserted by

AMESMA without compromising the diffusion time, by adapting the fanout factor based solely on information collected locally by the devices.

Simulation results indicated that decisions based on global view would lead to an efficiency gain when compared to cooperation and local view. It also showed that any form of diffusion improves the efficiency when compared to the no diffusion approach. Although global view proved to be more efficient, it is impractical, or at least inefficient, in dynamic mobile applications due to the cost associated to gathering and updating information. The goal is to investigate more efficient forms of collecting information locally and regionally.

The problem addressed here is complex and needs to be thoroughly investigated. Validation is performed by means of simulations, where results showed that the general efficiency of the essential services has been improved, giving us confidence that similar results will be obtained when the model is implemented in real devices. We have started the development process to produce a concrete solution of our model. Moreover, we envisage other essential services being analyzed and how to improve the local and cooperation view, while keeping the cost at an acceptable level. Furthermore, we intend to scrutinize existing QoS description models to decide how these models can be used in our approach.

References

1. Chiani, M.: Wireless technologies. In: Bellavista, P., Conrradi, A. (eds.) The Handbook of Mobile Middleware, ch. 3, pp. 52–73. Prentice Hall (2006)
2. Jaroucheh, Z., Liu, X., Smith, S.: A perspective on middleware-oriented context-aware pervasive systems. In: Ahamed, S.I., Bertino, E., Chang, C.K., Getov, V., Liu, L., Ming, H., Subramanyan, R. (eds.) COMPSAC (2), pp. 249–254. IEEE Computer Society (2009)
3. Engelstad, P.E., Zheng, Y., Koodli, R., Perkins, C.E.: Service discovery architectures for on-demand ad hoc networks. International Journal of Ad Hoc and Sensor Wireless Networks 2(1), 27–58 (2006)
4. Viana, W., Andrade, R.M.C.: Xmobile: A mb-uid environment for semi-automatic generation of adaptive applications for mobile devices. J. Syst. Softw. 81(3), 382–394 (2008)
5. Maia, M.E., Rocha, L.S., Andrade, R.M.: Requirements and challenges for building service-oriented pervasive middleware. In: ICPS 2009: Proceedings of the 2009 International Conference on Pervasive Services, pp. 93–102. ACM, New York (2009)
6. Erl, T.: Service-Oriented Architecture : Concepts, Technology, and Design. Prentice Hall PTR (August 2005), `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&amp;path=ASIN/0131858580`
7. Coulouris, G.F., Dollimore, J.: Distributed systems: concepts and design, 4th edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2005)
8. Schade, S., Sahlmann, A., Lutz, M., Probst, F., Kuhn, W.: Comparing Approaches for Semantic Service Description and Matchmaking. In: Meersman, R. (ed.) OTM 2004, Part II. LNCS, vol. 3291, pp. 1062–1079. Springer, Heidelberg (2004)
9. Andrade, R.M.C., Logrippo, L.: Morar: A pattern language for mobility and radio resource management. In: Dragos Manusecu, J.N., Volter, M. (eds.) Pattern Language of Program Design 5, ch. 10, pp. 213–256. Addison-Wesley (2006)

10. Liu, C., Peng, Y., Chen, J.: Web services description ontology-based service discovery model. In: WI 2006: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, pp. 633–636. IEEE Computer Society, Washington, DC (2006)
11. IBM, An architectural blueprint for autonomic computing (2005)
12. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, ch. Introduction, pp. 1–14. Springer (2003)
13. Groenevelt, R., Nain, P., Koole, G.: The message delay in mobile ad hoc networks. Perform. Eval. 62, 210–228 (2005)
14. Feller, W.: An Introduction to Probability Theory and Its Applications, vol. 1. Wiley (January 1968)
15. Bar-Yossef, Z., Friedman, R., Kliot, G.: Rawms - random walk based lightweight membership service for wireless ad hoc networks. ACM Trans. Comput. Syst. 26(2), 1–66 (2008)
16. Cornell, U.: Jist/swans java in simulation time/scalable wireless ad hoc network simulator (2008), `http://jist.ece.cornell.edu/`
17. De Meyer, K., Bishop, J.M., Nasuto, S.J.: Small-World Effects in Lattice Stochastic Diffusion Search. In: Dorronsoro, J.R. (ed.) ICANN 2002. LNCS, vol. 2415, pp. 147–152. Springer, Heidelberg (2002)
18. Khelil, A., Marrón, P.J., Becker, C., Rothermel, K.: Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks. Ad Hoc Netw. 5(5), 531–546 (2007)
19. Drabkin, V., Friedman, R., Kliot, G., Segal, M.: Rapid: Reliable probabilistic dissemination in wireless ad-hoc networks. In: 26th IEEE International Symposium on Reliable Distributed Systems, SRDS 2007, pp. 13–22 (October 2007)
20. Liu, J., Issarny, V.: Qos-aware service location in mobile ad hoc networks. In: Proceedings of 2004 IEEE International Conference on Mobile Data Management, pp. 224–235 (2004)
21. Rellermeyer, J.S., Alonso, G.: Concierge: a service platform for resource-constrained devices. SIGOPS Oper. Syst. Rev. 41(3), 245–258 (2007)
22. Liu, J., Issarny, V.: Qos-aware service location in mobile ad hoc networks. In: Proceedings of 2004 IEEE International Conference on Mobile Data Management, pp. 224–235 (2004)
23. Kramer, J., Magee, J.: A rigorous architectural approach to adaptive software engineering. Journal of Computer Science and Technology 24, 183–188 (2009)
24. Hales, D., Arteconi, S.: Slacer: A self-organizing protocol for coordination in peer-to-peer networks. IEEE Intelligent Systems 21(2), 29–35 (2006)
25. Bailey, N.: The Mathematical Theory of Infectious Diseases and its Applications. Griffin, London (1975)
26. Tracy Camp, V.D., Boleng, J.: A survey of mobility models for ad hoc network research. Wireless Communications and Mobile Computing 2(5), 483–502 (2002)