Supporting Smart Applications in Multihop Ad-hoc Networks

- The GecGo Middleware -

Peter Sturm, Hannes Frey, Daniel Gšrgen, and Johannes Lehnert

University of Trier, 54289 Trier, Germany
{sturm,frey,goergen,lehnert}@syssoft.uni-trier.de

Abstract. The goal of the GecGo middleware is to provide all the services required by self-organizing distributed applications running on multihop ad-hoc networks. Because of the frequent as well as unreliable and anonymous communication between accidental neighbors observed in these networks, applications have to adapt continuously to changes in the mobile environment and the GecGo middleware offers the required tight coupling. Additionally, GecGo addresses specifically the issue of *en passant* communication, where moving neighbor devices may interact only for short periods of time. In this paper, the architecture and basic concepts of the GecGo middleware are discussed and a prototype implementation of GecGo using the Microsoft Windows CE 4.2 .NET operating system for mobile devices and the .NET Compact Framework is presented. The paper also addresses issues on application programming and example application prototypes for these mobile networks.

1 Introduction

The emerging capabilities of modern mobile devices with respect to CPU power, wireless communication facilities, and battery capacity are the foundation of future multihop ad-hoc networks. The frequent as well as unreliable and anonymous communication between accidental neighbors observed in these mobile networks makes their successful deployment a challenging task. With the absence of any reliable backbone network, all mobile devices have to participate altruistically in a distributed execution environment with some kind of epidemic message delivery. Self-organization is the most promising design principle in order to manage these networks successfully and efficiently. As a consequence, any decision of a mobile device must be based on local as well as on current neighborhood knowledge and common goals must be achieved by means of synergy.

Any fundamental communication pattern in such a network exhibits an *en passant* characteristic. Two devices are within communication range for a short period of time and while they pass each other, they might cooperate and exchange certain data. In most cases, being within communication range with a given device is purely accidental and the probability to meet this device again in the near future is fairly low. During this en passant communication, applications and middleware must agree fast on which entities should change the hosting device in order to get closer to their final destination. The required decisions depend on a number of factors, among others the importance of the moving entity, the size of the entity compared to an estimation of the remaining interaction period, and the future direction of the neighbor with respect to the final destination.

These stringent conditions for distributed applications in multihop ad-hoc networks aggravate the need for the continuous adaption to a dynamically changing environment. Smartness is the key to success and only by utilizing knowledge-based mechanisms in a self-organizing system structure, application components on mobile devices can adapt successfully. As a consequence, this requires a very tight coupling between the mobile applications and the middleware. Many high-level mechanisms that are common in traditional system software and middleware that trade transparency vs. performance are therefore inadequate.

The goal of the GecGo middleware (Geographic Gizmos) is to offer this tight interaction with application components and to provide all the necessary services required by self-organizing systems running on multihop ad-hoc networks. In the next section, the fundamental concepts and the basic functionality of the GecGo middleware are introduced. The prototype implementation of the GecGo middleware using the Microsoft Windows CE 4.2 .NET operation system and the .NET Compact Framework is discussed briefly in section 3. In section 4, the basic structure of smart applications for multihop ad-how networks is introduced and examples of prototype applications are given. The paper ends with an overview on related work and a conclusion.

2 Concepts of the GecGo Middleware

The conceptional structure of the middleware and its four basic abstractions are depicted in figure 1. Any mobile or stationary device participating in the GecGo runtime environment is represented by a `DeviceGizmo` and the code of GecGo applications is derived from the base class `CodeGizmo`. Every code has its residence in form of a device. Depending on the distributed execution model, this residence remains fixed or it might change over time (mobile agents). For application code with a fixed residence, GecGo provides the abstraction of mobile state `(StateGizmo)` that might change the hosting device instead of the code. Since end-to-end messages between devices may remain on a device for a longer period of time in case no suitable neighbor is found, they also exhibit a more state-like nature. As a consequence, messages are represented in GecGo as special cases of a `StateGizmo`.

The fourth abstraction is defined by the `VenueGizmo` which ties a logical place resp. event to a well-defined set of geographic coordinates and time slots, e.g. a several week long lecture on distributed systems with changing rooms and time slots. `VenueGizmos` are virtual in the sense, that they bear no computational resources per se. Instead they rely on the devices that are within a given distance from the venue center. Entities with a venue as their destination will first try to reach a device at the venue. As long as they have no other destination, they will try to remain at the venue possibly by changing the hosting device.

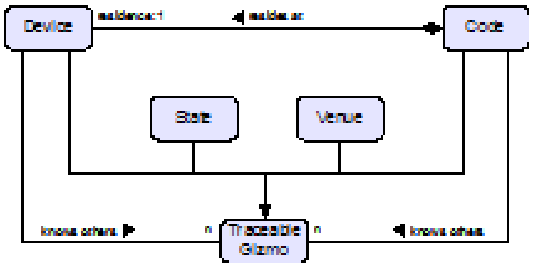

Fig. 1. Main GecGo Abstraction

All major abstractions in GecGo are derived from a fundamental data type `TraceableGizmo` (see figure 2). Any subtype of this class is traceable in time and space by means of a `GeoTrace`. These traces keep accounts on events in the past, they reflect the present situation, and they store estimates about future events. The actual information stored in the trace of a gizmo is defined by its type and consists of `a set of so-called Gepots` (pieces of time and geographic data). Also the depth and the level of detail of the `GeoTrace` depends on resource considerations and the actual type of gizmo. For example, `DeviceGizmos` keep track about where they have been in the past, at what time as well as why and they may also store information about previous neighbor devices. The present informs about the current position of the device and the actual neighborhood. The future trace might contain estimates where the device will be in the future, e.g. students will be in certain future lectures with a high probability.

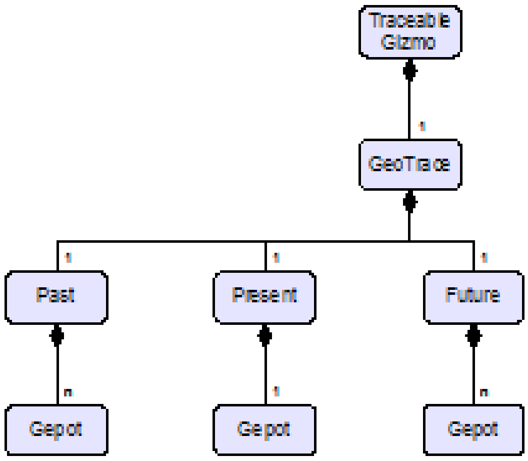

Fig. 2. Basic type "Traceable Gizmo"

Traces of `StateGizmos` will be more resource-limited. They will store at least the final destination as part of the future trace. `VenueGizmos` are even more restricted, since they represent only virtual entities within the GecGo environment. As such, the trace of the venue is identical to the time schedule of the event associated with this venue. Additional data that might be important to run the venue must be stored by the hosting devices that are currently within the vicinity of the venue center.

All devices are required to update their traces continuously over time. With the goal to keep the number of Gepots in the past to a reasonable minimum, the information stored in the present of the trace will be shifted into the past, e.g. when a mobile device starts moving again. The traces of devices are also the primary source for changes in the traces of other currently hosted state and code gizmos.

From a conceptual point of view, the main function of the GecGo middleware enables traceable gizmos to move towards new destinations. The most common type of movement allows for mobile state to reach a given mobile device or to get into the vicinity of a certain venue, e.g. to implement the marketplace communication pattern for multihop ad-hoc networks as presented in [4]. Covered by the concepts of GecGo are also the movement of mobile devices to reach a given venue (the special case of a navigation system, of course with the physical help of the human device owner) and movement of mobile code between devices as a means to implement mobile agent systems.

3 NET Implementation of GecGo

A first prototype version of GecGo has been implemented using the Microsoft Windows CE .NET 4.2 operation system and the .NET Compact Framework. The basic architecture of the GecGo middleware consists of two gizmo management domains (see also figure 3):

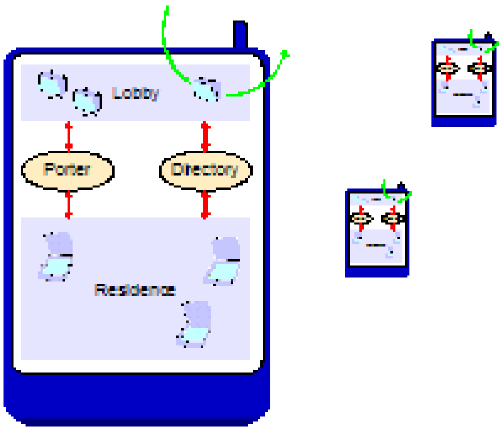

Fig. 3. GecGo Device Architecture

- the `Lobby` for all the gizmos that are in transit and haven't reached their final destination yet
- the `Residence`, with gizmos that are intended to stay at this device for a longer period of time

A central directory service keeps track on any changes in both management domains. Gizmos in the lobby and the residence may query for the existence of certain gizmos and they may register a delegate to be informed about specific events. For this purpose, every gizmo has an application defined unique name that serves as the key for the directory service. Possible events are: (a) arrival of a new gizmo with a specified type in the lobby of a device, (b) departure of a gizmo from the lobby, or (c) movement of gizmos between the lobby and the residence.

The directory itself has a hierarchical structure with leaves at the gizmo level. Applications may query the directory with wildcards to locate the required information. Most of the attributes of a gizmo entry are application-specific. Events of the .NET framework may be used to implement asynchronous notifications between different gizmos. The same mechanism is used to implement the aforementioned events that are provided by the directory service itself. For example, if a gizmo inside the residence wants to be notified upon the arrival of gizmos of a given type `T` in the lobby, it simply registers a delegate with the event `/Lobby/T/<Enter>` and the middleware will call back each time such a gizmo enters the device.

Movement of gizmos from the lobby to the residence and vice versa will be performed with the aid of the porter service. Primarily, the porter is responsible for securing the identity of incoming gizmos and for providing the resources requested by the entity.

A central decision in mobile ad-hoc networks addresses the issues on mobile code vs. mobile state. Mobile agents are an interesting technology for wireless and mobile networks with far reaching implications on system security and code integrity. The GecGo middleware covers mobile agents in its architecture by accepting a changing residence for mobile code. This functionality is currently not part of the .NET implementation of the GecGo middleware platform. Besides technical reasons, this decision is primarily driven by a number of unsolved problems with respect to the limited resources on a mobile device, the larger amount of data required to move `mobile code including its execution state transparently from one device to another, and the need to authenticate and secure code execution.`

Instead of mobile code, the GecGo middleware actually offers so-called *mobile state*, which requires the application components to cooperate non-transparently in packaging and unpacking execution state into and from mobile state gizmos. The middleware offers several functions and services to ease this task for the application code. In contrast to mobile code, applications must be installed explicitly by the user of a device, before state gizmos for a given application can be received and processed on their final destination. Of course no application code must be installed on devices that are only intermediate hosts for state gizmos.

4 Supporting Smart Applications

Relying solely on the information local to the mobile device and on the limited support of the dynamically changing neighborhood is a challenge for every distributed application executed in an ad-hoc environment. As a consequence, the application depends much stronger on the information and services a middleware for these networks can offer. And the application has to adapt continuously to the changes in the environment. Conversely, the middleware services can only perform their tasks in close cooperation with all the application components residing on a device. As a consequence, a very tight coupling is required between middleware and application as well as among the applications themselves. Detailed reasons for this are, among others:

- The decision, which gizmos to move while being in contact with another device strongly depends on the applications. And since this information will change over time in structure and content, it must be managed by the application components themselves and accessed by the middleware services on demand.
- Performance issues force the middleware to cooperate very closely in the physical movement of gizmos from one device to the other during the short interaction periods. Only the application components can successfully define a minimal state gizmo to be transmitted.
- The altruism required among the mobile applications enforces various mechanisms to balance the resources on the device in coexistence with the traditional applications and to schedule the access to i/o devices (especially the wireless network access).

This first prototype version of the GecGo middleware platform is currently used to implement several example applications, to gain experience with the abstractions provided by the middleware and to improve the platform architecture and functionality. We started with the development of a simple e-learning application: a peer-to-peer quiz for students to assist in the preparation of examinations. The basic idea is to enable participating students to issue interesting examination questions. These questions are propagated by the GecGo middleware to the corresponding venue that has been assigned to the specific course. Participants interested in examination questions for a given course will issue a request that too will be propagated to the corresponding venue where it remains for some period of time to collect new items. This collection of new questions will be realized by means of additions to the initial mobile state gizmo. Eventually, the request will move back to the sending owner and any results will be presented to the user. Additionally, the application enables students to rate and to order a set of questions from a didactical point of view. Rates and orders are again sent to the venue to be accessible to other participants.

The implementation of additional mobile applications for ad-hoc networks using GecGo is planned for the near future: a mobile auction system and a self-organizing electronic rideboard in an university environment [Frey H, Lehnert J. K, and Sturm P. “Frey H, Gšrgen D, Lehnert J. K, and Sturm P. “”]. These applications have been investigated already on a simulated basis [Lehnert J. K, Gšrgen D, Frey H, and

Sturm P. “Frey H, Gšrgen D, Lehnert J. K, and Sturm P. “”] and as prototypes running on a java-based middleware called SELMA [Gšrgen D, Lehnert J. K, Frey H, and Sturm P. “”, the predecessor of GecGo.

5 Related Work

Traditional middleware systems such as CORBA, Microsoft DCOM or Java RMI are not suitable for mobile ad-hoc networks because they rely on central infrastructure like naming services and assume the reachability of all network nodes. These assumptions cannot be matched by mobile multihop ad-hoc networks. Additionally, traditional middleware approaches are too heavyweight for mobile devices. Many adaptions have been made to apply them in mobile settings such as OpenCORBA [**Error! Reference source not found.**] or NextGenerationMiddleware [**Error! Reference source not found.**]. These extensions provide mechanisms for context awareness, but cover mainly infrastructure networks and one-hop mobile communications.

An increasing number of middleware systems is developed specifically for mobile ad-hoc networks. XMIDDLE [**Error! Reference source not found.**] allows the sharing of XML documents between mobile nodes. Lime [**Error! Reference source not found.**] and L2imbo [11] are based on the idea of tuple-spaces [12], which they share between neighbored nodes. But due to the coupling of nodes, these approaches are not well-suited for highly mobile multihop ad-hoc networks. MESHMdl [13] employs the idea of tuple-spaces as well, but avoids coupling of nodes by using mobile agents, which communicate with each other using the local tuple-space of the agent platform. Proem [14] provides a peer-to-peer computing platform for mobile ad-hoc networks. STEAM [15] limits the delivery of events to geographic regions around the sender which is similar to the geographically bound communication at marketplaces. STEAM provides no long distance communication, it is only possible to receive events over a distance of a few hops.

Mobile agent frameworks exist in numerous variations, Aglets [16] or MARS [17] may serve as examples. These frameworks were designed for fixed networks and thus the above mentioned problems of traditional middleware approaches apply to them as well. The SWAT infrastructure [18] provides a secure platform for mobile agents in mobile ad-hoc networks. This infrastructure requires a permanent link-based routing connection between all hosts and thus limits the ad-hoc network to a few hops and it is therefore not applicable to en passant communication pattern.

6 Conclusions

The specific nature of multihop ad-hoc networks enforces a tight coupling between the middleware and any mobile application. The sole dependence on information local to the mobile device leads to new programming and execution models, that favor self-organization and adaption to a continuously changing environment. Only smart applications survive in such a rough environment. The specific architecture of the GecGo middleware as presented in this paper is trying to address these issues by supporting mobile application components and by providing flexible interaction mechanisms between entities on a single device as well as entities on mobile devices that are within communication range for short period of times.

7 References

[1] Frey H, Lehnert J. K, and Sturm P. "*Ubibay: An auction system for mobile multihop ad-hoc networks*," Workshop on Ad hoc Communications and Collaboration in Ubiquitous Computing Environments (AdHocCCUCE'02), New Orleans, Louisiana, USA, 2002

[2] Gšrgen D, Frey H, Lehnert J, and Sturm P, "*Marketplaces as communication patterns in mobile ad-hoc networks*,'' *in* Kommunikation in Verteilten Systemen (KiVS), Leipzig, German, 2003

[3] Lehnert J. K, Gšrgen D, Frey H, and Sturm P. "*A Scalable Workbench for Implementing and Evaluating Distributed Applications in Mobile Ad Hoc Networks*," Western Simulation MultiConference WMC'04, San Diego, California, USA , 2004

[4] Gšrgen D, Lehnert J. K, Frey H, and Sturm P. "*SELMA: A Middleware Platform for Self-Organzing Distributed Applications in Mobile Multihop Ad-hoc Networks*;" Western Simulation MultiConference WMC'04, San Diego, California, USA, 2004

[5] Frey H, Gšrgen D, Lehnert J. K, and Sturm P. "*Auctions in mobile multihop ad-hoc networks following the marketplace communication pattern*," submitted to 6th International Conference on Enterprise Information Systems ICEIS'04, Porto, Portugal, 2004

[6] Frey H, Gšrgen D, Lehnert J. K, and Sturm P. "*A Java-based uniform workbench for simulating and executing distributed mobile applications*," FIDJI 2003 International Workshop on Scientific Engineering of Distributed Java Applications, Luxembourg, Luxembourg, 2003 (to appear in Springer LNCS)

[7] Ledoux T. "*OpenCorba: A reactive open broker*," Springer LNCS, Volume 1616, pp. 197ff, 1999

[8] Blair G. S, Coulson G, Robin P, and Papathomas M. "*An architecture for next generation middleware*," in Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer-Verlag, London, UK, 1998

[9] Zachariadis S, Capra L, Mascolo C, and Emmerich W. "*XMIDDLE: Information sharing middleware for a mobile environment*," in ACM Proc. Int. Conf. Software Engineering (ICSE02). Demo Presentation, Orlando, Forida, USA, 2002

[10] Picco G. P, Murphy A. L, and Roman G.-C. "*LIME: Linda meets mobility*," in International Conference on Software Engineering, pp. 368-377, 1999

[11] Davies N, Friday A, Wade S. P, and Blair G. S. "*L2imbo: A distributed systems platform for mobile computing*," ACM Mobile Networks and Applications (MONET) - Special Issue on Protocols and Software Paradigms of Mobile Networks, Volume 3, pp. 143-156, Aug. 1998

[12] Ahuja S, Carriero N, and Gelernter D. "*Linda and friends*," IEEE Computer, Volume 19, pp. 26-34, Aug. 1986.

[13] Herrmann K,. "*MESHMdl - A Middleware for Self-Organization in Ad hoc Networks*," in Proceedings of the 1st International Workshop on Mobile Distributed Computing (MDC'03), 2003

[14] Kortuem G. "*Proem: a middleware platform for mobile peer-to-peer computing*," ACM SIGMOBILE Mobile Computing and Communications Review, Volume 6, Number 4, pp. 62-64, 2002

[15] Meier R and Cahill V. “*STEAM: Event-based middleware for wireless ad hoc networks*,” in 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW '02), Vienna, Austria, 2002

[16] Lange D and Oshima M. “*Programming and Deploying Java Mobile Agents with Aglets*,” Addison-Wesley, 1998

[17] Cabri G, Leonardi L, and Zambonelli F. “*MARS: A programmable coordination architecture for mobile agents*,” IEEE Internet Computing, Volume 4, Numer 4, pp. 26-35, 2000

[18] Sultanik E, Artz D, Anderson G, Kam M, Regli W, Peysakhov M, Sevy J, Belov N, Morizio N, and Mroczkowski A. “*Secure mobile agents on ad hoc wireless networks*," in The 15th Innovative Applications of Articial Intelligence Conference, American Association for Articial Intelligence, 2003

This work is funded in parts by the german science foundation DFG as part of the Schwerpunktprogramm SPP1140 and by the Microsoft Research Embedded Systems RFP 2003.