An InfoSpace Paradigm for Local and Ad hoc Peer-to-Peer Communication

J. Brehm, G. Brancovici, C. Müller-Schloer, T. Smaoui, S. Voigt

Institute of Systems Engineering – System- and Computer Architecture, Appelstraße 4,
30167 Hannover, Germany
{brehm, brancovi, cms, smaoui, voigt}@sra.uni-hannover.de

R. Welge

Dpt. of Automation Technology, University of Applied Sciences of NE Lower Saxony
Volgershall 1, 21339 Lüneburg, Germany
welge@fhnon.de

Abstract. A key feature of ubiquitous handheld devices will be their simple usability in mobile and highly dynamic ad-hoc peer-to-peer environments. With LINDA, JAVASPACES and TSPACES, shared memory-based communication concepts have been realized on the system level with corresponding APIs. This article proposes to consequently use the same simple communication paradigm also on the user interface level. In this paper we examine existing basic technologies, analyze typical application scenarios and the underlying communication patterns. We discuss in detail, how these patterns can be realized with the InfoSpace mechanism which offers to each user a private local space and a view on a common space shared between all participants. Finally we show an architecture which realizes the InfoSpace infrastructure and report on first experience with a JXTA-based solution.

Introduction

The direct interaction between people is helped if not dominated by context information. Clues given by body language like facial expression, gestures, voice melody etc., as well as by the common knowledge of present time and location make otherwise cryptic messages intelligible. Handheld computers so far have very limited sensory equipment. Today, basic context information can be made available with relatively little effort [1]. Examples are location (point in the 3D space), position (vector in the 3D space), time and date, present user, temperature etc. With little effort, also the proximity of one device to another can be detected. Here, the limited range of Bluetooth [2] connections is an advantage.

We make use especially of the latter possibility and will exploit the related context information. Two (or more) devices which find themselves in the same physical space (the extent of which is defined by the radio range) are candidates for local information exchange. They can enter a common information space, an InfoSpace, – provided their respective users choose to do so.

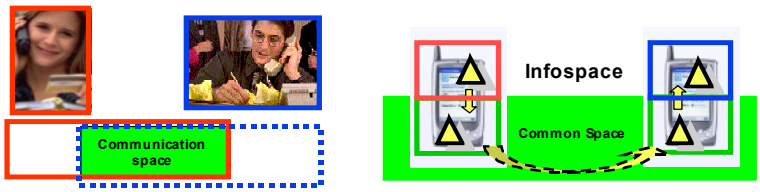


Fig. 1. Human and ubiquitous communication spaces

The mind set of two humans communicating to each other can be described in the same terms: The existence of a physical channel – e.g. audibility plus eye contact – establishes a common information space. Every communication partner is allowed to drop information into this common space, usually guided by synchronization rules (s. fig. 1). Additional partners can be accepted to this space or excluded from it. A user interface which stays as close as possible to these traditional and well accepted patterns has good chances to get adopted.

Projects dealing with information sharing and exchange in the local area have been realized. The 7DS Project [3] of the University of North Carolina is a peer-to-peer system providing distributed information retrieval among (wireless) hosts in the neighborhood. It is based on HTTP request/answer, the GUI for the information retrieval is implemented by a web browser. Each peer stores data in a separate cache. The DataSpace project of Winlab at Rugers is a further example of interesting work concerned with forming information spaces related to physical objects. The spaces (the so-called datacubes) are of different scales. Locality and distance in DataSpace are not defined by the logical structure of the underlying network but rather by geographic space [4]. In this paper we will make clear how our project differs from the above mentioned ones in the underlying architecture as well as in the user interface concept.

This project builds on the JINI, JXTA and tuple space technologies. Tuple spaces are a basic shared memory paradigm proposed by D. Gelernter [5]. JINI and JXTA provide basic functionalities to advertise and discover services. They are alternatives in this project because they provide similar functionality.

JINI [6,8] is a network technology which permits various devices to build an ad hoc network. The devices can offer services over JINI for other devices. JINI has the following characteristics:

- Ad hoc networking: JINI builds connections without any knowledge about the underlying physical network. Hence JINI is prepared for dynamic networks.
- Service Infrastructure: All devices can use and offer services. A typical example is a printer service.
- Connecting heterogeneous systems: JINI is based on JAVA. All devices which support JAVA can participate in a JINI community.
- Leasing: Leasing is an especially important feature in JINI. All Objects in JINI have a validity period. This supports the idea of ad hoc networks and dynamic networking.

JINI was first released in 1999 by Sun. Sun released JXTA two years later. The JXTA project [7,9] started as a research project to address peer-to-peer[1] communication. This project has its roots in programs like Napster and Gnutella which connect particularly in a peer-to-peer mode instead of using connections based on the client-server model. The main target of JXTA is to form a platform which offers all the functionality to work in a peer-to-peer mode. JXTA intends to solve problems like:

- Interoperability: JXTA is designed to enable peers to provide various peer-to-peer services to locate each other and to communicate with each other.
- Platform independence: JXTA does not depend on the programming language, transport protocols and deployment platforms. This is reached with a fortified use of XML.
- Ubiquity: The goal of JXTA is to make any device accessible and allow it to communicate.

The main difference between both architectures is that JXTA only defines a set of protocols, expressed in XML. So it is only defined how to communicate and there is need of equal platforms. JINI uses Java objects to communicate, so the architecture assumes that Java Byte code can be interpreted anywhere. Therefore Jini is protocol independent but is tied to the Java platform.

Further on JINI uses a centralized service location broker. In a JINI community there must be at least one Lookup Service. It is responsible for finding and searching services. In JXTA entities can find other entities without a centralized service.

Each of these architectures works fine as a middleware in an ad hoc community. Due to its basically decentralized architecture and the smaller footprint we have decided in favor of JXTA.

The envisaged InfoSpaces system sketched in fig 1 requires a complex infrastructure which will be described in this article. Chapter 2 will discuss a few application scenarios where InfoSpaces can be used. In chapter 3 we describe the basic interaction patterns necessary for a local peer-to-peer communication between individuals and groups. We will show that a tuple space is well suited for asynchronous communication. The following chapter 4 describes the basic concepts and protocols of our InfoSpace architecture, chapter 5 a mapping between the interaction patterns and some application scenarios and chapter 6 the architecture of the underlying infrastructure. In chapter 7 we report results of a JXTA-based pilot implementation. The final chapter 8 summarizes the article and gives an outlook on future work.

Application Scenarios

The following examples serve illustrative purposes. With their help, we will be able to check the suitability of the InfoSpace paradigm for real world applications. Actually, many more application scenarios exist. We will come back to more scenarios in chapter 5 after we will have discussed the mechanism in detail.

[1] For clarity we would like to stress that the term „peer-to-peer“ in conjunction with InfoSpaces is used to refer to (local) network technologies based on equal peers cooperating with each other for performing a certain task. The term does therefore in our context not mean any of those networks (Gnutella, KaZaa…) which are mostly used for file sharing across the internet.

Ad-hoc Communication

1. A visitor of a computer fair walks from one exhibition area to the next with her handheld device turned on. Thereby she traverses the related potential InfoSpaces offered by the respective exhibitors. If she decides to accept one of the offers she enters the InfoSpace and starts to download information or to upload enquiries.
2. Two visitors meet in a crowd and want to exchange their business cards. Out of the many InfoSpaces offered around them, they choose a private one for themselves and exchange the desired information.

Interactive Lecture

For an interactive lecture, we assume that all students are physically present in the same room or lecture hall. They are all equipped with notebook computers or tablets, connected to a preferably wireless communication network. Defined by the physical range of wireless communication of their notebook computers, the group members are (physically) eligible for becoming members of the InfoSpace. The teacher might want to restrict access to a smaller (logical) group of members of the InfoSpace. The notebooks provide to each accepted member of the group a common view which can be used for information exchange in a very intuitive manner.

For the presentation, the instructor uses “transparencies” which carry the backbone information of the lecture. There will probably be also some kind of multimedia enhancement like incremental build-up of the transparencies, animations, videos and sounds or illustrative simulations. Two kinds of annotations will be used: The instructor will want to emphasize, add or correct the information on the transparency. The student wants to have copies of these remarks and, in addition, wants to add his own annotations.

A practise phase should interrupt the presentation every 20 – 30 minutes. This can be done by assigning short task assignments to the students. A few multiple choice questions could assist the students in a self test. Also a short simulation carried out by the student is possible. An automatic answering and correction mode keeps this phase from becoming too time consuming.

The students can ask written questions to the instructor anonymously during the presentation or in special Q&A periods. The inhibition of many students to ask possibly silly questions in the public can be overcome this way. On the other hand, there should be a possibility for the instructor to trace questions back to the originator in order to prevent misuse.

The test phase is similar to the practise pattern but requires a collective or individual back channel. An anonymous back channel allows for an aggregate statistical evaluation of the answers which might lead to an adaptation of the presentation speed, stepping back or skipping of parts of the lecture. Individualized task assignments and answers make possible the assignment of micro credit points to the student.

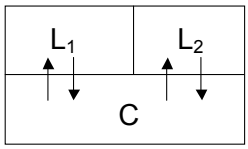
Project Team

The objective of a project team is learning by doing in a co-operative team environment. An example is the group-oriented development of a computer program. A specific interaction pattern could be the one promoted by "Extreme Programming" [10] proponents.

In a first step, a project team has to be instructed about the task assigned to them. A common workspace is necessary which reflects and co-ordinates changes made by the participants. Notebooks involved will form an ad-hoc community. In addition to the shared development target (the program), a common discussion channel or space has to be provided. In case of a team physically present in one location, the natural voice channel will serve this purpose. Otherwise, a conference-call set-up is necessary which requires means for synchronous communication.

Infokiosk

An infokiosk is the electronic equivalent of a bulletin board. It is realized through large wall-mounted displays, possibly with touch-sensitive surface. In the simplest case, the user can interact with the information kiosk via fingertip and browser. If the user is equipped with a portable device (PDA, tablet, notebook) she can set up a conversation between her portable device and the infokiosk. This conversation probably will involve some identification of the user. Additional context information like the location of the infokiosk (in front of the lecture hall or in the student restaurant) can help to pre-select the information offered. This interaction might lead to the download of selected information to the portable device or, vice versa, to the upload (or posting) of information to the bulletin board.

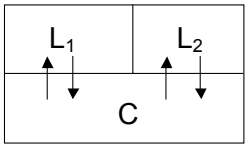


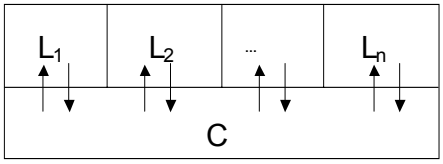
Interaction Patterns

We claim that the tuple space paradigm which essentially is a globally shared memory for tuples or – in our case – objects, is sufficient to handle all interaction patterns (I-patterns) typical for peer-to-peer-communication in a local environment. In the following we will analyze such interaction patterns in more detail. The basic operations available to the users are *move* and *copy*. Every user i has a local view L_i holding symbols (i.e. references to objects) stored within his own device, and a common view C shared with all users of the same InfoSpace. The total of the two local spaces L_1 and L_2 and the common space C is one logical Infospace. Each device i sees its device space consisting of the local space L_i and a view of the common space C.

I-pattern A

The simplest interaction pattern is a symmetric peer-to-peer-communication between two equal partners. Both partners want to move or copy objects between their respective L-view and C. There is no restriction on the type of objects or the sequence of copies or moves. An example of this type of interaction A(1:1) is the encounter of 2 persons (with their InfoSpace-compatible handheld devices). An InfoSpace is built up with two users, both have write and read access to the common space C. They might use the InfoSpace e.g. to exchange their business cards.





The I-pattern A can be generalized from the 1:1 relationship to an n:n-pattern. n users participate in the same InfoSpace, all of them may write into C or read from it. As in the 1:1 case, there are no restrictions on the type of objects or the sequence of operations.

An example for A(n:n) is a discussion group with free information exchange. Every participant offers and consumes information at the same time. This arrangement requires a certain discipline and trust between the partners in order to avoid overloading of C or e.g. offensive postings. Hence, there has to be an acceptance procedure to become a member of this InfoSpace.

I-pattern B (Broadcast)

I-pattern B restricts the access of users to the common space C. It allows one of the users to post information while all the others can only consume it. This I-pattern constitutes a broadcast situation. It generally makes sense only in a 1:n relationship: B(1:n). An example for I-pattern B(1:n) would be the menu in the students' restaurant which is published by the restaurant owner in a dedicated InfoSpace. It is not desirable to let users (students) use this I-Space for their personal postings.

I-pattern G (Gathering)

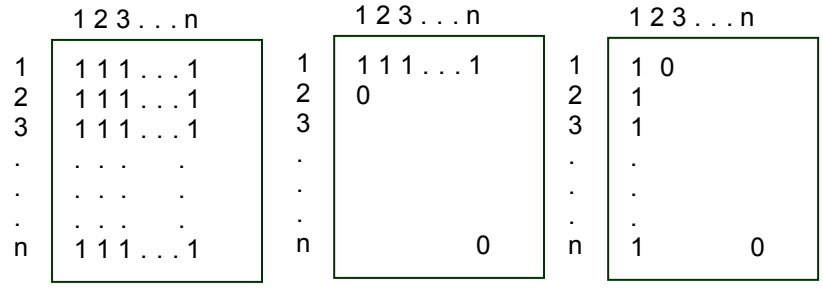
I-pattern G is the inversion of pattern B. Here we have n writers to the common space and (usually) just 1 reader: G(n:1). This type of InfoSpace can be used to collect and later evaluate information from a group of users. An example could be questionnaires filled in by students or answered multiple choice questions. The instructor plays the role of the collector.

Interaction Matrix

	A	B	G
U_1	RW	W	R
U_2	RW	R	W
U_3	RW	R	W
.			
:			
U_n	RW	R	W

If we denote the right to write into C with W, and the right to read from C with R, the 3 I-patterns can be shown as a table which assigns rights to the users U_1, U_2 … U_n. Alternatively (and more generally) we can describe the I-patterns in terms of a directed relationship graph or its equivalent interaction matrix M. The users of the InfoSpace are listed as rows and columns of the matrix M. Matrix element $m_{ij} = 1$ if there is a write relationship from user i to user j, otherwise 0.

The matrices for the I-patterns A, B and G take following form:

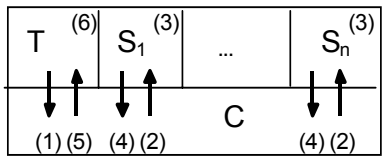


I-pattern A(n:n) I-pattern B(1:n) I-pattern G(n:1)

Complex Interaction Patterns

Obviously the above interaction matrices are just special cases. A general interaction pattern would be represented by an arbitrary distribution of 0's and 1's over the interaction matrix M. Such patterns make sense in special applications and must be supported by the InfoSpace mechanism. We will discuss just one such application as an example taken from the university scenario as sketched in the introduction to this article.

In an interactive lecture, the lecturer will from time to time ask questions to the students in order to get feedback on their progress. The sequence of interactions is the following:



1. Lecturer broadcasts question (same question to all students).
2. Students read the question.
3. Students answer question (local action).
4. Students write answers to C.
5. Lecturer collects answers from C.
6. Lecturer evaluates answers (local action).

Since all participants write and read at some time, we have I-pattern A(n:n). Looking more closely, we have patterns B(1:n) superimposed by pattern C(n:1). This

means that, in addition to a general assignment of R and/or W rights to the common space C, we need a more fine grained protection mechanism on the level of the single users vs. the single objects. This solution is based on an access pattern carried by each object, and a capability assigned to each user. This allows us to place objects with different access patterns into the common space C and check capabilities vs. access patterns at access time.

InfoSpaces

Infospace Paradigm

The properties of all tuple space-based approaches (LINDA, JAVASPACES, TSPACES) make them interesting for the realization of InfoSpaces. As the name indicates, an InfoSpace is a common space to share information. In the context of this paper the expression information is used for data and services synonymously. An InfoSpace mimics the communication environment which is set up spontaneously whenever two persons start to interact. It is required to meet the following goals:

1. Simple usability: Only few functions to provide and consume information have to be made available through a graphical user interface.
2. Interoperability: Automatic ad hoc networking between a wide variety of different devices.
3. Service infrastructure: All devices can start and support services. These services can be used by any other device.
4. Data infrastructure: All devices can provide data to and consume data from the InfoSpace.
5. Context sensitivity: The selection process for the shared information should be context driven.

From the user point of view all elements of an InfoSpace are objects ready to be used (in the case of services) or ready to be consumed (in the case of data). The user should not be bothered with configuring network connections, browsing through large file systems, entering network addresses and similarly annoying chores. In contrast, he should be supported by his device to make life easier which is one of the main issues of ubiquitous computing.

The view of user i on an InfoSpace is divided into two parts, a local space L_i and a common space C. The local space contains all data and services provided by the local device (workstation, PC, PDA, mobile phone, etc.). The common space is dynamic in a sense that its data and services are provided in an ad hoc network built from devices with their respective local spaces. We want to emphasize the strict separation between the local space, which is a private area exclusively for the local user's activities without any possibility for other users to access it or change its content, and the common space, which represents a kind of interface between all users that are free to read and write objects into it (copy in/out, move in/out). The result is a peer-to-peer network connection between two devices (fig. 2) creating an InfoSpace consisting of two local spaces (L_1, L_2) and a common space C.

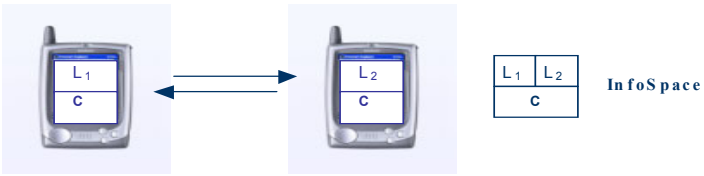


Fig. 2. User views (2 users A and B, 2 devices) on 1 InfoSpace

The common space C is symmetric for both devices and users. The information in the common space can be provided and used by both devices. Any information from a local space can be transferred (e.g. by drag and drop) to the common space. It is possible to specify certain attributes (metadata) for the information provided. Examples of metadata are owner, creation date, type, short description, copy counter, leasing time, number of consumers, etc. If nothing else is specified, the information is valid as long as its provider participates in the InfoSpace. If there are more than two devices (fig. 3), an ad hoc network is created with a distributed management of the InfoSpace.

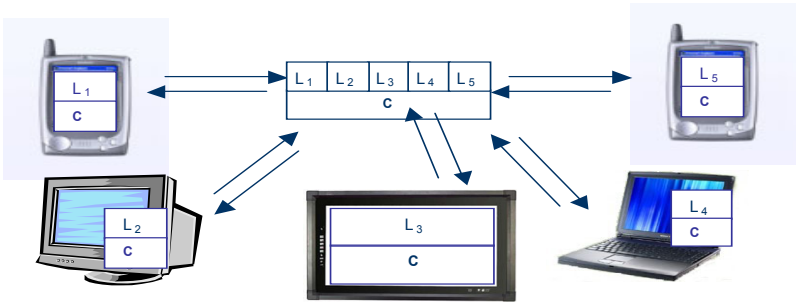


Fig. 3. User views (5 users, 5 devices) on 1 InfoSpace

There are different ways to move information between a local space (L_i) and the common space (C):

1. Manually by the user: Only a few functions to provide and consume information have to be made available through a graphical user interface.
2. Automatically: Each device is aware of the context (e.g. location, time and date, sensor information). Based on these contexts, information can be automatically provided or consumed. Agents are responsible for these tasks on behalf of the user.In any case, information is either copied or moved from L_i to C. The system then updates the common space C to make it consistent between all users. In turn, other users can consume information from C by copy or move operations.

Information Selection

A user has three possibilities to select the information he wishes to see:

1. By accepting one of several offered InfoSpaces he enters a communication relationship with a certain partner or group. The availability of an InfoSpace is determined by the existence of a physical communication channel between the two devices. E.g. in the wireless case the two or more devices must be in their mutual range of radio signals. After he has selected a specific InfoSpace, he sees all the information that is offered via the common space C corresponding to this InfoSpace.
2. L-space selection: A mapping operation from the local storage area (file system) of user i to the local view L_i selects (e.g. by filtering) objects which are of importance in a certain communication context. An example for such a selection is the set of objects belonging to a specific lecture.
3. C-space filtering: In order to avoid a possibly overloaded C space, additional filtering operations can be applied to the common view. Filters use the metadata information of the objects. It is important to note that filtering is an operation strictly local to the receiving device. This avoids centralized computation on the offering side on behalf of the receivers.

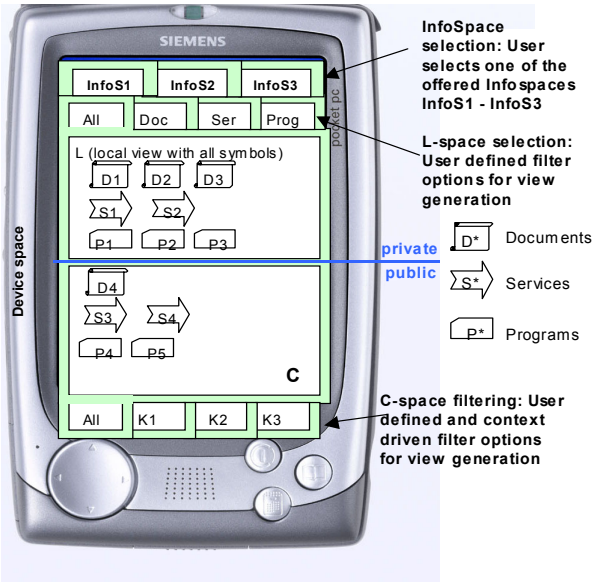


Fig. 4. InfoSpace-capable device with the local view L, the common view C, the InfoSpace selection area (top) and the L-space and C-space filter setting areas

Fig. 4 shows the InfoSpace GUI with the local space L and the common space C. Several symbols representing different objects are shown. A transfer (copy or move, see below) can be carried out by moving the respective symbol. Crossing the L to C borderline constitutes also a transition between the private and the public space. It is guaranteed by the system, that there is no illegal intrusion from a remote device into the private space. All transfers are visible and explicitly initiated by the user. The

agents with the task to scan e.g. the common space for certain desired objects constitute the only exception to this rule. The policy in general is: Everything is forbidden unless it is explicitly allowed.

Hence the user has to carry out four types of operations via his GUI:

1. Selection of one of the offered InfoSpaces (top section of the GUI in fig. 4)
2. Selection of information from the local storage to L_i.
3. Information transfer between L_i and C.
4. Setting filter options for C (lower section of the GUI).

We now have to investigate the actions necessary in the InfoSpace infrastructure in order to keep the system consistent. In particular, we have to discuss the movement and visibility of symbols (references) and the corresponding data transfers.

InfoSpace as an Abstract Communication Mechanism

As already stated, the basic idea of an InfoSpace is to visualize the paradigm of a common globally shared but physically distributed memory on the GUI level and to provide a small number of intuitively simple graphical commands to manipulate information objects in order to make them available to other users or to withdraw them from the common view. In the preceding section we have shown some transfers of objects as seen from the user point of view. At the same time, the InfoSpace can be perceived also as an abstract communication mechanism between applications. The mechanism is then accessed not via (graphical) user commands but through an API. The abstract InfoSpace comes close to the usage other projects like TSPACES and JavaSpaces have made of the tuple space paradigm.

In a strict sense, we have to discriminate between two levels: the abstract InfoSpace and its visualization on the GUI level. The abstract InfoSpace is not visible to the user other than through a GUI. The InfoSpace GUI is a 1:1 visualization of the abstract InfoSpace. Hence, the (human) user is enabled to directly manipulate InfoSpace objects with the help of the GUI. Alternatively, the InfoSpace can also be manipulated (via its API) by application programs. A possible application program can be an agent equipped with a filter or search mask which continually scans the common space C for objects matching its search pattern. In this sense, the InfoSpace GUI resembles the shell of an operating system which allows the human user to access the OS services. The shell also co-exists with other programs which use the operating system via system calls.

Operations and Protocols on InfoSpaces

When two “InfoSpace-capable” devices meet (i.e. a physical connection is established) they will tell their users that an information partner is present and that an InfoSpace can be used if wanted. When both users agree to participate, the InfoSpace consisting of local spaces (L_1, L_2) and common space C is established (compare fig. 5). Now exactly four basic operations are possible: copy out, move out, copy in, move

in. These operations allow the users to share information and services easily by manipulating symbols on the GUI level.

The following copy and move operations are possible if local information from device 1 is to be made accessible for device 2:

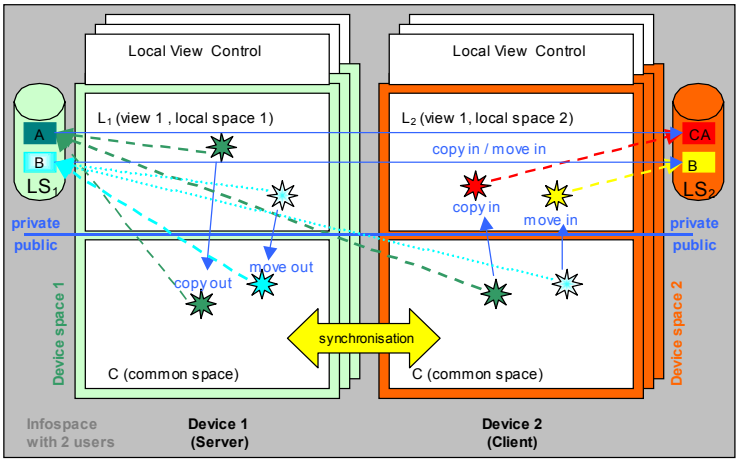


Fig. 5. User views (2 users, 2 devices, 2 device spaces) on 1 InfoSpace, information transfer from LS_1 to LS_2

- Copy to common (copy out): Symbol A is copied from L1 to C. After the copy out operation, symbol A is visible in the common space on both devices. The reference A points to object A in LS1 (Local Storage 1).
- Move to common (move out): Symbol A is moved from L1 to C. After the move out operation, symbol A is visible in the common space on both devices. Symbol A is deleted from L1. A symbol A* (grayed out) remains in L1 pointing to a backup copy A* in LS1. The reference A points to object A in LS1.
- Copy from common (copy in): Symbol A is copied from C to L2, a copy of object A in LS1 is transferred to LS2 (Local Storage 2). The reference from the copied symbol A points to object A in LS2.
- Move from common (move in): Symbol A is moved from C to L2, object A is transferred from LS1 to LS2, symbol A is removed from C. After the "move in" operation, symbol A is only visible on device 2, the reference from symbol A in L2 points to object A in LS2. The "move in" operation is potentially dangerous since it results in the deletion of object A in the local storage of device 1. Therefore a user who moves out an object from his local area to C is warned and a backup copy is generated. Despite its dangerous nature we felt that a consequent move operation including the deletion of the source object is necessary to allow for a mutually exclusive update of objects if this is desired by the user.

  Since this tuple space based approach is symmetric, local information on device 2 can be made accessible for device 1 in the same way.

The four primitives described above are - as already mentioned - basic (i.e. atomic, not resolvable), so that concurrent access to objects of a common space is impossible. Moreover we do not privilege any device by restricting the execution of a certain primitive to certain devices. I.e. every device in the common space has potentially the right to read or write objects from and into the common space (s. also above: Interaction Matrix). Such "device-discrimination" could be very useful for realizing certain application scenarios like the interactive lecture. Such security issues will be subject of future work.

Usage Scenarios and Interaction Patterns

After introducing the i-patterns and the InfoSpace paradigm in the last two chapters, we would like to illustrate how these i-patterns could be matched to concrete usage scenarios we are currently working on.

Usage Scenario 1: Data Exchange over the common space

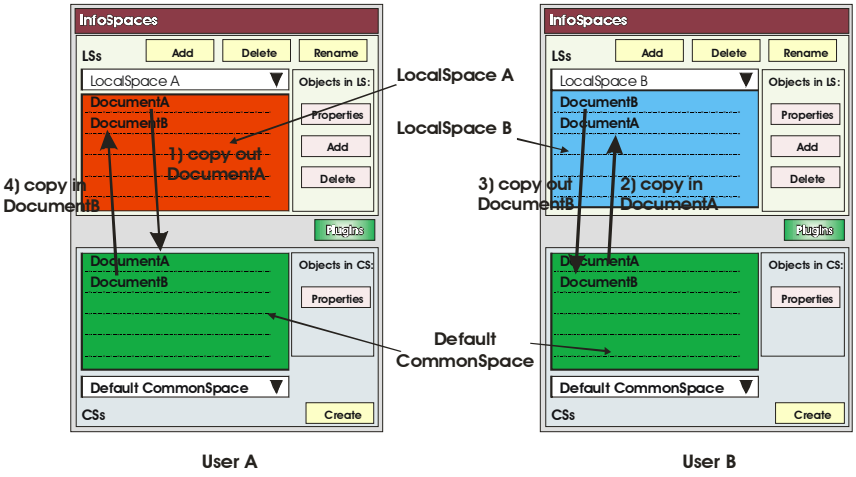


Fig. 6. Object exchange via default common space

This scenario represents the simplest application of the InfoSpaces and in the same time the basis for any other usage. It shows how the InfoSpaces could be used for the spontaneous, intuitive and uncomplicated data (object) exchange between two or more devices in an ad-hoc manner. Suppose two devices running the InfoSpace meet so that the fundamental condition about locality is fulfilled. Without any "extra-effort" (e.g. manual configuration, search...) of the users a (default) common space that comprehends both devices emerges, allowing any user to publish his objects and the others to pick them up. Fig. 6 shows how this could look at the GUI-level.

As - per default - there is no privileged or discriminated entity, we are dealing in this scenario obviously with I-pattern A, where any user can read and write in the common space.

Usage Scenario 2: The Infokiosk

As introduced above, the infokiosk is a sort of electronic information board. One can make use of infokiosks to realize a dynamic calendar of events of a city or a schedule in a train station or provide students on a campus with information about current lectures and the possibility to access their own examination results. The infokiosk scenario in its simplest form can be realized with the help of an InfoSpace application running permanently. The infokiosk offers a specific common space filled with objects that can be consumed by any user. The user is in this case merely a "silent" participant. He has access to the common space and its objects but cannot change its content (i.e. add or remove objects from it). The infokiosk agent, which is a kind of daemon process, will be the only entity that writes into the common space. This corresponds to I-pattern B.

Usage Scenario 3: Match-Maker

The idea behind the match-maker scenario is to allow users in a foreign environment to search for adequate fellows or buddies. For that purpose the InfoSpace is supplied with an additional component (agent) which will be responsible for automating certain operations. Users who have joined a dedicated (not the default) common space, let's call it the buddy-search common space, publish (copy out) their profile into the buddy-search common space. An activated agent on a certain user's device will collect all profiles (copy in) and examine according to some (predefined) criteria which one is relevant. If an interesting profile is found, then the match is successful and other actions are triggered. The latter user and the corresponding profile owner can for instance be brought together in a separate common space where they can exchange private data. In this scenario we have obviously the I-pattern G, where many users write their objects (profiles) in the buddy-search common space and one user – with the help of an agent – collects and evaluates them.

An InfoSpace Architecture

An architecture to realize InfoSpaces is shown in fig. 7. The InfoSpace Middleware controls the InfoSpaces between the devices. It can be addressed via APIs by applications and through GUIs. To be able to connect a variety of devices it is assumed that communication between devices is possible using the TCP/IP protocol transparently regardless of the physical implementation of the communication channel.

The **InfoSpace Middleware** consists of the following modules (compare fig. 7): View Manager (VM), Connection Scanner (CS), Space Director (SD), Profile man-

ager (PM) and Space Manager (SM). The single modules are briefly characterized in the following.

View Manager VM

The View Manager administrates user defined profiles to set up different views to the local device space (L and C). The default profile shows all symbols available for the basic operations (copy in/out, move in/out). To make it more comfortable the user can define local view profiles to filter symbols. The trigger for the filters can be context driven (location, time, etc.) and/or object driven (to differentiate between documents, programs, services).

Connection Scanner CS

The Connection Scanner is necessary for devices with radio links or optical links such as infrared connections. The CS permanently scans for radio/light signals and checks the signal quality. If the transmission quality is adequate a dialog with the device partner is entered to find out whether the device partner is "InfoSpace compatible". If so, an ID-exchange is carried out automatically and the connection to the InfoSpace is established. The CS is also responsible to disconnect devices from the InfoSpaces if the signal quality is poor or to disconnect devices upon explicit user command.

Space Director SD

The Space Director is informed by the CS about the availability of connections. The SD is responsible for the construction and destruction of common spaces between devices. The SD gets active upon request by the CS. Depending on the context and user demands, the SD initiates the construction of one or more common spaces. If only one communication partner is available, the choice is reduced to one common space. If a group of devices meets it is possible to construct one common space for all devices or to construct bilateral common spaces or to construct groups of common spaces. This decision depends on the needed communication and interaction patterns. To be able to make this decision information has to be exchanged. The SDs of the devices offer their local common spaces to all partners. Depending on profiles from the Profile Manager a decision for the construction of the common spaces can be made. This decision can be automatically (context driven) or user driven. If the user has to decide all offered common spaces are displayed. The SD accepts the user's selection.

The SD also negotiates server/client roles with the partner SD. If new connections are available the SD automatically informs the Space Manager. The SD also deletes users from the common space on explicit demand or upon request from the Connection Scanner.

Profile Manager PM

The Profile Manager administrates local profiles of InfoSpaces. These profiles contain acceptance requirements for establishing InfoSpaces. Thus, the SD is enabled to compare local profiles with offered profiles. Profiles for active partner search or passive acceptance can be set by the user.

Space Manager SM

The Space Manager is responsible for the management of objects in the local device space. It can play a server or client role. There is one SM per device space. The SM can access the local space L and the common space C to copy or delete objects. Thus, the control of the basic operations (copy in/out, move in/out) is carried out by

the SM. To correctly perform the basic operations the SM has to communicate with the partner SMs.

The different components described above are realized on the basis of existing class libraries, such as JXTA.

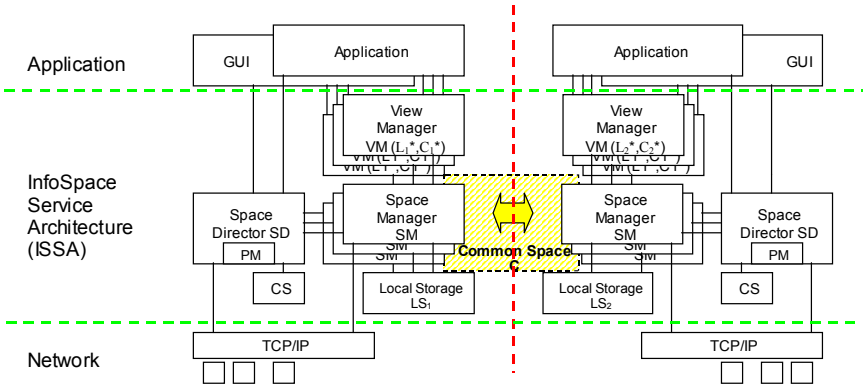


Fig. 7. InfoSpace architecture (CS = Connection Scanner, PM = Profile Manager)

Pilot Implementation using JXTA

Since the release of JXTA, we have been watching the development of the project. Furthermore we have gained some experience by using the platform and its Java reference implementation for developing a demo InfoSpace application. We do not use JXTA as a whole with the complexity of its architecture and its API involved. We rather use specific aspects which serve in particular the ad-hoc and local nature of our InfoSpace Service Architecture (short ISSA). Fig. 8 illustrates the logical architecture of our InfoSpace software. The thickness of every box indicates the importance of the corresponding technology in the model. Notice in particular where JXTA was integrated, namely - apart from the network layer which could be considered as an "external part" of JXTA - at the very bottom level.

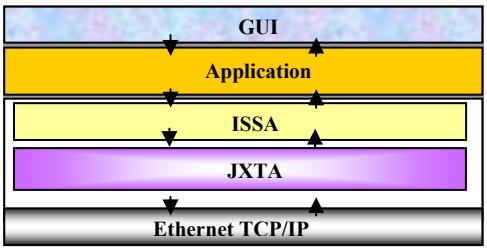


Fig. 8. Architecture of the InfoSpace demo application

Let us describe the architecture bottom-up. The bottom layer depicts the network layer. We have developed the architecture in an Ethernet LAN environment. This is justified by the nature of ISSA, since InfoSpaces will be exclusively deployed between devices that are spatially "close" to each other. This has also a further advantage as far as JXTA is concerned, since JXTA uses different mechanisms for publishing and discovering in LAN and non-LAN (i.e. wide area) environments. Of course those mechanisms designed for LAN are simpler and more reliable. Besides, for this first implementation we have made no assumptions on resources such as computing, storage or network bandwidth in order to concentrate on the JXTA platform and provide a comfortable user interface.

JXTA provides many features which coincide with the ISSA specification. Maybe the most important one is the peer group service. A peer group in JXTA represents a gathering of peers with a common interest. For instance if several peers are interested in a particular service (e.g. content sharing) or in a particular topic (e.g. astronomy) a peer group – given a significant name - is initiated by a peer. Any other peer that can discover this peer group and fulfil its membership policy can apply for membership and enter the group. The idea of membership service represents furthermore an interesting mechanism for building private (password-protected) groups. Due to those properties, it was obvious that the JXTA peer group – in conjunction with other services - could be used to realize the common space. In particular, the fact that peer groups are dynamic and their membership is not stringent is in harmony with our InfoSpace specification. Peer groups can arise ad hoc (default peer group) and vanish autonomously (without a central managing entity) as the last peer leaves the group. Also peers can join and leave a group, without affecting the existence of the group. All these qualities of peer groups are conceived very cleverly by means of advertisements which describe the peer group and which are propagated among peers without the necessity of a coordinating broker service.

Beside the peer group service, we have used extensively the peer discovery service and the corresponding peer discovery protocol (PDP). Both represent the mechanism in JXTA for discovering and publishing resources including peer groups, peers or any other resource "registered" in a JXTA advertisement.

While peer groups merely represent a framework (context) for any kind of activities, other services are responsible for filling the common space with content. Here we have made use of the content management service (CMS). CMS is a JXTA peer group service used for publishing and sharing content on a local peer, demanding a list of shared content from remote peers of the group and retrieving content [7]. Important is

that contents are first advertised to the peer group members in form of content advertisements before any actual data transfer takes place. The content advertisement, a kind of identity of the remote peer content, will be collected by every peer in the group (or InfoSpace) and corresponds therefore to the reference to a remote object in our ISSA architecture (compare fig. 5).

Since ISSA has already been described above in this paper, we will confine our description of the third layer in fig. 8 on trying to map JXTA components to certain blocks of our ISSA architecture as shown in fig 7. As already illustrated, the common space which is a part of the InfoSpace is realized through the JXTA peer group service. The functionality of the space manager, particularly with regard to accessing the local storage, managing objects of the local space and publishing them in the common space is covered by the CMS. The role of the space director responsible for the construction of new InfoSpaces is taken by the JXTA platform itself, which constitutes the infrastructure for invoking any other core or middle layer services.

The last two upper layers: Application and GUI realize some other functionalities of the ISSA like the local space and its object references and the defined four primitives copy in/out and move/in out as defined above.

To sum up, JXTA has proved to be appropriate for implementing our demo application, since it specifies useful mechanisms, services and protocols that deal with some of the fundamental aspects that we have standardized. On the other hand, – as result of studying the JXTA standard[2] and its current implementation - we are not planning to deploy JXTA in its "raw" form as delivered from the open source community for the future work. We rather intend to customize certain protocol and service implementations which on the one hand will suit better our ISSA and on the other hand may lead to saving precious resources (bandwidth, compute power, storage) for an optimal and "really" ubiquitous (anywhere, anytime) use of our InfoSpace.

Summary and Outlook

We have introduced the concept of InfoSpaces as a new and simple interaction paradigm with ubiquitous devices. InfoSpaces consequently follow the original idea of tuple spaces but raise the scheme to the user level. The very simple interaction via drag-and-drop based on copy and move operations makes InfoSpaces an ideal candidate for small footprint ubiquitous devices.

Analyzing a university application environment as an example we have found a small number of basic interaction patterns between two or more users. Typical applications like an interactive lecture can be mapped to sequences of these basic I-patterns. We have also shown that the InfoSpace paradigm quite naturally lends itself to the implementation of these basic patterns.

The InfoSpace project is building on existing technologies. We have analyzed and compared possible candidates for the implementation of InfoSpaces and decided to use JXTA due to its small footprint and adequate functionality.

---

2 JXTA is to be understood as both: a standard and an implementation.

The paper has introduced an architecture to realize the InfoSpaces middleware and has shown in some detail the protocols to handle the information transfer.

As a result we think that a JXTA-based realization of InfoSpaces is quite promising. First implementation work resulted in a pilot application.

While the InfoSpace paradigm is already applicable for realizing some scenarios based on locality and context-awareness, there is, however, still space for improvement in terms of security and the use of the common space for collaborative work (e.g. collaborative editing of objects).

We plan to use the InfoSpace technology in the UbiCampus project which aims to provide all our computer science students with notebooks or other mobile devices. We feel that the broad supply of more or less naked notebooks to the students is rather useless unless they are integrated into an easy-to-use infrastructure with a rich set of services. InfoSpaces has been designed to provide exactly this infrastructure which is missing so far.

Finally we want to point out that the envisaged university environment is just one out of many possible application scenarios. Today's more or less complicated and chaotic interactions between wirelessly connected devices during project meetings, conferences or fairs clearly demonstrate the necessity of generally accepted and simple ad hoc peer-to-peer communication mechanisms.

References

1. J. Hightower, G. Borriello: Location Systems for Ubiquitous Computing, IEEE Computer, August 2001, pp. 57 - 66
2. C. Müller-Schloer, P. Mähönen: The UbiCampus Project: Applying Ubiquitous Computing Technologies in a University Environment, Proc. IDMS 2000, October 2000, University of Twente, Springer, pp. 297-303
3. 7DS: Peer-to-Peer Information Dissemination and Resource Sharing, http://www.cs.unc.edu/~maria/7ds/
4. http://paul.rutgers.edu/~gsamir/dataspace/
5. David Gelernter: Parallel Programming in LINDA, Technical Report 359, Yale University Department of Computer Science, Jan. 1985
6. http://www.jini.org
7. http://www.jxta.org
8. W. Keith Edwards: Core Jini, Prentice Hall, Dezember 2000, 2nd Edition
9. Early Adopter JXTA: Peer-to-peer Computing with Java, Sing Li, Wrox Press Ltd.,Dezember 2001
10. K Beck: „*Extreme Programming Explained"*, Addison-Wesley 2000
11. http://www.almaden.ibm.com/cs/TSpaces/