ELSEVIER

Computer Networks 35 (2001) 411–428

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

DEAPspace – Transient ad hoc networking of pervasive devices

Reto Hermann [*], Dirk Husemann, Michael Moser, Michael Nidd, Christian Rohner, Andreas Schade

IBM Research Division, Zurich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland

Abstract

The rapid spreading of mobile computerized devices marks the beginning of a new computing paradigm characterized by ad hoc networking and spontaneous interaction, taking place transparently to the human user. The DEAPspace project described in this paper aims at providing a framework for interconnecting pervasive devices over a wireless medium. It supports the development of new proximity-based collective distributed applications. In this paper we discuss the motivation of the project and describe possible new application scenarios from which requirements for both the supporting framework and the underlying wireless medium are derived. Central research issues such as a new push-model-based approach to fast and resource efficient service discovery, and encoding and match-making of compact service descriptions, are elaborated in more detail. The paper also describes some related work and concludes with a critical review of existing wireless communication technologies, followed by a description of the current state of our project and an outlook on future research directions. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Transient ad-hoc networking; Proximity based networking; Service advertising; Service discovery; Distributed service discovery; Push model; Peer-to-peer communication; Service description; Multi-device applications; Bluetooth; IEEE 802.11; IEEE 802.15; HomeCast OpenProtocol

1. A world of connected devices

Most of the things we deal with in our day-to-day life are invisible from a computer science point of view – they either do not even contain a microprocessor or they do not offer an interface to the outside world, let alone communicate with other devices. Nevertheless, we already are surrounded by a host of devices containing a microprocessor of some sort. Moreover, we increasingly encounter devices that not only contain an embedded controller or system but in addition provide an interface for exchanging data with other devices. It does not take a lot of imagination to come to the conclusion that we are on the road to a world of devices that are not only controlled by a microprocessor but also are capable of exchanging information with other devices through some kind of interface:

- already we carry with us GSM phones, personal digital assistants (PDAs), digital watches, and other wearable devices;
- very few office environments survive without a computer, printer, or network;
- modern cars increasingly rely on computers not only to control the operation of the car itself but

[*] Corresponding author. Tel.: +41-1-724-8573.
E-mail address: hud@zurich.ibm.com (R. Hermann).

also to provide additional services such as navigation, communication, and entertainment;
- in our homes we use computerized devices to control our home appliances, to entertain us, and to patrol our property;
- and sensors that provide data about ourselves and our environment pop up everywhere.

Concurrently to the microprocessor pervasion, wireless radio communication technologies (such as Bluetooth or IEEE 802.11) become available that are first steps towards environments in which communication is no longer wire-bound and each device can cheaply communicate with other devices over some form of radio technology. The combination of information appliances with these wireless networking technologies enables an entire range of new applications and usage scenarios.

The DEAPspace project aims at providing a framework for interconnecting pervasive devices over wireless communication technologies. Using this framework, devices that are in proximity to each other can form ad hoc transient alliances to create new types of collective distributed applications. To accomplish this goal, the following questions have to be answered. How can we find out about the information available and the services that devices offer? What kind of applications become possible? Can we provide a framework of building blocks for developing such applications? A particular focus of our work is on *proximity-based networks* and *mobile nodes*; in this context we need to find out about available services in a fast manner as communication relationships are potentially of rather short duration.

In the following sections we present a set of application scenarios that capture the scope of DEAPspace and, based on these scenarios, describe the requirements, challenges, and goals of our research and briefly discuss how our work compares with existing, related research projects. The main part of our paper describes the underlying service model and our approach to service description, followed by a presentation and discussion of the DEAPspace service announcement and discovery protocol. After a discussion of the existing wireless communication technologies and their suitability for transient ad hoc networking, we conclude the paper by describing the current state of our work as well as future research directions.

2. Application scenarios

As a basis for discussion we present three application scenarios. The first scenario illustrates a multidevice application where several devices render one virtual application. The second scenario is centered around a personal information management (PIM) application; here we use DEAPspace technology to hide data synchronization. The third and last scenario illustrates a context-aware application.

The first scenario describes a *multidevice application.* Alice wears a digital wristwatch, carries an ultra-modern PDA containing not only the usual PDA applications (such as date book, address book), but also an attached MP3 player with a headset including a microphone. In her backpack she carries her GSM phone. With current technology, when a phone call comes in, her GSM phone will start ringing, and all Alice can do is either take down her backpack, open it, look for the phone, check the phone to see who is calling, and decide whether to take the call, or she can just ignore the call and hope that whoever is calling will leave a message on the answering service provided by her network operator.

With DEAPspace technology, her GSM phone discovers her digital wristwatch and notices that it offers an alarm service, a limited display service, and an input service (the wristwatch has two buttons). Instead of ringing in the backpack, the GSM phone invokes the alarm service of the digital wristwatch, then the display service of the watch to display the number of the calling party, and the input service of the watch to wait for Alice to accept the call. Alice now merely has to glance at her watch to find out who is calling and then press one of the two buttons of the wristwatch to either accept the call or ignore it. As the phone call is from Bob, her manager, she decides to accept the call. The GSM phone in the meantime has found out that her MP3 player offers a voice output and input service; as soon as Alice has accepted the phone call through her wristwatch, the GSM phone connects with the MP3 player, and Alice is talking to Bob.

The second interesting scenario describes a *hidden personal information management* application. PIM refers to a set of applications such as e-mail, address book, calendar, to-do or task lists, shopping lists, user preference profiles, which are typically jointly hosted on organizers, PDAs, laptops, and/or desktop computers. For the end-user, synchronization of the information across all devices becomes increasingly cumbersome and a task in its own right. Wireless radio-based communication technology can remedy the situation because it can make the synchronization implicit, thus "hiding" it from the user. The following scenario illustrates this transparency.

Bob has just left his home and is driving to work when his phone indicates with a beep the arrival of a message. His wife asks him to buy some flowers and bring them to dinner at his mother-in-law's place tonight. Whereas Bob might well forget the task after a day's worth of work, he rests assured that he will be reminded because his phone has transferred the appointment message to both his PDA and his wrist watch. In the office he switches on his desktop computer and starts working through a full inbox of e-mails. Suddenly, a pop-up window alerts him that there is a conflicting calendar entry. His secretary has scheduled the video broadcast of the CEO's annual kickoff speech and it happens to collide with his wife's request to shop the flowers. As Bob is well versed in the "We've had an outstanding year, but this does not mean that ..." speeches, he decides to skip that speech and instead take care of the flowers as promised. Later that afternoon, while he is engaged in a lively discussion in one of his colleague's offices, the alarm on his wrist sounds, reminding him to leave for the flower shop.

The third scenario involves those situations where we could profit from context-related information and describes a *context-aware application*. Already today, information kiosks supply visitors of exhibitions, museums, entertainment parks, malls, stations, airports, etc. with location-dependent information. Whereas now visitors must walk up to these kiosks to collect the information on paper, radio-based communication technology allows information kiosks to propagate information to the person's information appliance. As the kiosk detects the information appliance within radio range, it tries to obtain user-specific context information (age, gender, origin, etc.) and pushes the tailored information to the appliance. The arrival information in the appliance may then issue an alert to the user or, alternately, the information may simply be cached in the appliance with some adequate time-to-live. Let us look at an airport scenario in more detail.

Alice is driving to the airport to leave for a business trip when her car information system alerts her of a traffic jam on the route to airport. Unfortunately she cannot take another route and, thus, there is nothing she can do to avoid being held up. Alarmingly late she finally arrives at the airport car park, leaves her car and rushes to the elevator. While she is traveling to the departure floor, the airport passenger information system reads out the ticket details from her e-wallet. While still in the elevator, Alice's attention is drawn to her wristwatch by its alarm indicating the arrival of a message. As she hurries towards the departure hall, she reads on display of her watch that she is to proceed directly to gate A37. Simultaneously the security guards at the entrance to terminal A are alerted to her late arrival. As she walks up to the checkpoint, they give her expedited treatment. Bypassing the queue, she passes the checkpoint and proceeds to the gate where the hostess is already waiting for her. Minutes later she is seated in the plane, breathless and thinking that perhaps taking the subway would not have been such a bad idea after all.

3. Requirements, challenges, and goals

Using these scenarios we can extract a set of features that define the scope of our DEAPspace work. A key feature is that applications can span multiple information appliances; that is, they are *distributed*. Moreover, we target a wide range of appliances: from small devices (such as a wristwatch) to large devices (such as the elevator in the airport scenario); from devices designed to accomplish a single function particularly well to devices that combine multiple functions in one box – DEAPspace intends to be *pervasive.*

Another key feature illustrated in the previous examples is *hidden computing*: the consumer is unaware of the applications executing. Applications engage with one another triggered by certain events such as the presence of a certain device in an environment. Applications may alert the user once a result has been obtained, or cache the result for later retrieval by the user. Related to hidden computing is another feature shared by our example applications, namely, *spontaneous interaction*. Interaction with a peer device is triggered by the discovery of the device and its services. Next, appliances should be able to engage in application transactions irrespective of whether they have met before; that is, we expect an *ad hoc* style of communication and application interaction. Tying in with the ad hoc requirement is the desire to have true *peer-to-peer communication* without requiring a master to be present (nor should a master election process have to take place).

Then, *proximity-bound computing* and *transient communication relationships* are key features exemplified by our scenarios. The limited radio range of the wireless communication technology employed by the appliances is used to establish spatial context (i.e., proximity). As we are dealing with mobile users and mobile devices, communication relationships will be of short duration determined by the range of the wireless communication technologies and the (relative) velocities with which devices are moving through this range. Relative mobility of devices can make this period arbitrarily small.

Also, applications will involve various media types, possibly including audio, images and even video data; thus, we need to be able to deal with *multimedia data.*

As the transient communication relationship feature is frequently overlooked when evaluating wireless communication technology let us illustrate this point in more detail. Assuming a mobile scenario with relative device velocity v and a wireless range r, the maximum time available for the application transactions to complete called interaction window t_W, is given as $t_W = 2r/v$. Table 1 illustrates the duration of the interaction windows for various mobility situations.

The execution of an application comprises a number of activities, each adding to the total execution time t_E required. These activities include discovering a peer device or an existing network, creating or joining a network, discovering (all) peer application entities (i.e., service discovery), connection establishment (including parameter negotiation), transaction execution, and finally disconnection.

Clearly, the following equation must hold:

$$t_E < t_W = 2r/v.$$

The primary goal of the DEAPspace research project is the development of suitable algorithms and formats (where necessary) and the implementation of a ready-to-use framework for building applications. Research issues that we have identified include prompt service discovery without a central service discovery server, compact and extensible service description, security in pervasive ad hoc networks, configuration and management of a large number of devices, new application models such as multidevice applications, and also wireless communication technology. To arrive at an experimental platform as soon as possible, we concentrated in a first step on service discovery, service description, and wireless communication technology.

Table 1
Interaction window t_W for various mobility situations

Mobility situation		Wireless range		
Description	Velocity	10 m	20 m	30 m
Person moving slowly	1 m/s	20 s	40 s	60 s
Person moving quickly	4 m/s	5 s	10 s	15 s
Two persons crossing slowly	2 m/s	10 s	20 s	30 s
Car moving slowly	50 km/h	1.44 s	2.88 s	4.32 s
Two cars crossing slowly	100 km/h	0.72 s	1.44 s	2.16 s

4. DEAPspace service model

The DEAPspace service usage model is role based. An entity providing a service that can be utilized by other requesting entities acts as a provider. Conversely, the entity requesting the provision of a service is called a requester. To provide its service, a provider in turn can act as a requester making use of other services.

DEAPspace forms a distributed system; that is, requesters and providers can live on physically separate hosting devices. In terms of service usage, however, the location of an entity is transparent to its partners. A service provided on the same device as the requester would be used as if it were a remote service.

Regardless of its particular role, an entity conceptually provides at most two interfaces, an input and an output interface. Each of these interfaces requires or emits data in particular formats. An entity (e.g., a microphone) that only provides an output interface acts as a data source from the point of view of the DEAPspace system. Conversely, a data sink is an entity that provides only an input interface (e.g., a speaker or a beamer). [1]

An entity acting as a provider accepts data in a set of different formats at the provider interface. Conversely, a requester can emit a set of different data formats through its requester interface. A provider *satisfies* a requester if the provider interface *matches* the requester interface. Two interfaces match if their data format sets are not disjoint; that is, there exists at least one data format that can be used to connect the two interfaces.

As depicted in Fig. 1, provision of a service is triggered by sending data in a supported format f from the requester interface i_r of a requesting entity to the provider interface i_p of a satisfying provider.

Communication between a requester interface and a provider interface can be bi-directional. This is the case if the data format in use is coarse grained and refers to a protocol such as FTP or SMTP consisting of different primitives.

[1] Sources and sinks can offer management interfaces through which their normal operation can be controlled. These management interfaces are of course input interfaces.

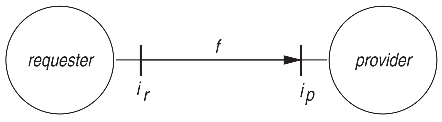


Fig. 1. Communicating DEAPspace entities.

The roles that entities in DEAPspace can assume are similar to those found in other models of distributed systems such as RM-ODP [11], CORBA [13] or DCOM [2]. The main difference is that the DEAPspace service usage scheme also allows streaming of data, and is therefore more general than the request-based mechanism of remote operations in object-oriented distributed systems.

Note that with DEAPspace the communication medium chosen to announce a service need not necessarily be the same as the communication medium used to invoke a service; for example, a device can choose to announce a service using a radio-based technology, while service invocation can only occur via infrared.

5. Compact service description

Service advertising and selection depend directly on the service description format. This is important in two aspects:

1. On the service provider side, the service description format determines the precision for services advertisements.
2. On the requester side, the service description format determines the selection capabilities for choosing a matching service offer.

When a provider advertises its services, descriptions of the services are encoded for distribution to potential users. A requester receives an encoded description of an available service, decodes it, and can then decide to use the service offering based on its description.

Service descriptions have to meet different, to some extent even contradictory requirements. On the one hand, the requirement for precise service advertising and matchmaking, for instance, advocates complex service description formats. On the other hand, there is a requirement for efficient encoding and decoding of service descriptions that

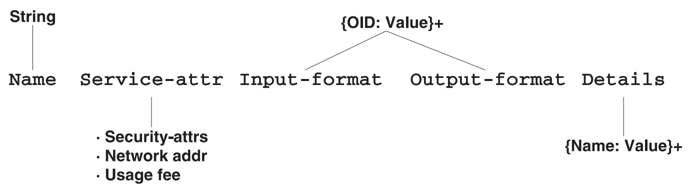


Fig. 2. DEAPspace service description format.

suggests using simpler data formats. The following requirements have been identified as being important to the DEAPspace platform:

- Service description formats must accommodate all information necessary to select suitable services allowing a fine-grained selection.
- The encoding scheme must generate small messages efficiently to preserve bandwidth.
- The coding of service descriptions must be simple and efficient to keep the size of the encoding and decoding modules (codecs) small.

DEAPspace assumes a very basic model of service usage in which services provide and/or accept data for processing. Therefore parts of a DEAPspace service description represent the service input and output in terms of acceptable and produced data formats, respectively. This general model allows both stream-based and message-based service deployment.

Fig. 2 illustrates the general idea behind the DEAPspace service description format. Each service has a short identifier (`Name`), a set of service attributes (`Service-attr`), an optional set of acceptable input formats (`Input-format`), an optional set of output formats (`Output-format`), and also an optional detailed information part (`Details`).

The DEAPspace service description format differentiates between mandatory and optional service attributes. While the mandatory attributes (e.g., service address) ensure that all advertised services can be utilized, the optional part (e.g., quality-of-service characteristics) allows service properties to be further qualified.

As no assumptions can be made as to which service types will be available in a DEAPspace system, the DEAPspace services description format can be used to describe arbitrary services. Format types are represented in a hierarchical form in which specific subtypes are derived from common supertypes. For efficient encoding and decoding, some standard format types are already known to the DEAPspace system. At any node, this type tree can be extended with application-specific derivatives.

5.1. Service description data structure

DEAPspace services are specified as a data hierarchy. The root node of this hierarchy is the `DSService` class. As depicted in the listing below, instances of this class contain the name of the service, a `ServiceAttribute` object, two `FormatType` objects, and an array of `NameValuePair` objects – the `Details` part of Fig. 2. [2]

```
DSService    ::= Name
                 ServiceAttribute
                 InputFormat
                 OutputFormat
                 NameValuePair*.
InputFormat  ::= FormatType.
OutputFormat ::= FormatType.
```

The `ServiceAttribute` object represents mandatory descriptive information about the service, such as some security-related information using the `SecurityAttribute` class, the fee (`Fee`) for using the service, and the service address (`DSNetworkAddress`). As a DEAPspace system can communicate using different communication protocols, a DEAPspace address is represented by

[2] Note that the extended Backus–Naur form (EBNF) listings of the service description format shown in this section are not complete.

an abstract class `DSNetworkAddress`. From this class all concrete network address description classes are derived, such as `IPNetworkAddress` for the IP protocol or `BtNetworkAddress` for Bluetooth addresses.

8 `ServiceAttribute ::= Security AttributeFeeDSNetworkAddress.`

The two `FormatType` objects in the `DSService` class describe the input and the output format of a DEAPspace service. The description of services acting as data sources in a DEAPspace system contains a null value for the input format type indicating that they do not accept any input from other DEAPspace services. Conversely, services that act as data sinks for the DEAPspace system carry a null output format type in their service description, indicating that the service does not produce any output that is useful to other DEAPspace services.

Format types are represented in a hierarchical manner, ranging from abstract format types to more specific ones:

9 `FormatType ::= DSoidCompFormatType* .`

DEAPspace provides an identification scheme for format types that is based on qualified node names in a tree. Each format type is associated with an identifier component that is unique relative to the supertype. A particular format type is identified by concatenating all identifiers from the tree's root along the path to the type node. This scheme is a generalization of the two-level-MIMEs, similar to the concept of object identifiers (OIDs) known from ASN.1 [9].

Each format type level is associated with a `DSOIDComp` object that describes an entity similar to an OID component. This OID component object either represents a standard format type (standardized within the DEAPspace framework) or the name of a proprietary extension possibly known only to some recipients of the service description object.

`FormatType` descriptions allow the inclusion of a list of next-level format type descriptions in addition to the `DSOIDComp` object. Standard format type classes (e.g., `ImageFormat`, `ApplicationFormat`, `SoundFormat`) have been defined as refinements of the abstract `FormatType` class. Thus, specific format types can be specified by appropriately arranging instances of those standard classes; for example, we represent the MIME type *application/postscript* through an `ApplicationFormat` object that contains an `AppPostscriptFormat` object in its list of subformats. Specific properties of the concrete format type are described on the corresponding level (e.g., the postscript version and release number as well as color or monochrome postscript). The advantage of this approach is that a series of format type specifiers can be folded into one.

The list of `NameValuePair` objects in a service description represents optional information used to differentiate further services. DEAPspace does not impose a particular structure on this kind of information – it is merely a sequence of name value pairs (both as byte arrays) with an associated type indicator enabling type-specific comparison operations. A recipient will usually have some expectation as to what the optional service description part conveys.

5.2. Coding service descriptions

The way encoding in general works is straightforward. The service description hierarchy is traversed and the information found in any of the nodes transformed into a serialized representation. Decoding is the reverse operation, in which the original data hierarchy is reconstructed from the serialized form.

As the encoder can use application-specific format types, the decoder may receive information that it is unable to process. For these cases, DEAPspace provides *opaque decoding*, which allows unknown format types to be read. While the data cannot be used locally, it can be passed on "as is" to another entity that may understand it. The decoding procedure skips the unknown format type, and re-synchronizes with the byte stream afterwards.

DEAPspace services can accept or emit multiple, in most cases similar format types. Normal encoding of those service descriptions would result in carrying redundant information: The top-level format type would be the same, merely the sublevel format would differ. To avoid this redundancy DEAPspace encoding allows sharing of top-level format types where only the sublevels differ, and accomplishes better compression than methods that traverse a service description as prescribed by the type hierarchy.

We have investigated five of the most common encoding methods: ASN.1 BER and DER encoding [10], XML [19], WBXML [18], Java serialization, and XDR [15]. None of them shows good results when viewed in terms of our optimization targets, codec size and message size. ASN.1 generates the shortest message, yet entails the largest codec module size. XML contains much overhead compared to payload data, and optimum WBXML is difficult to generate, resulting in large codecs. Using Java serialization we arrive at small codecs but also at the largest service description messages in our evaluation. XDR is a rather poor compromise as neither its code size nor its message size is particularly good. As a benchmark exercise we therefore decided to develop a DEAPspace-specific data-driven encoding and decoding scheme. The main reason for this is that all common encoding methods do not allow sharing format type information at the highest possible level. An encoding procedure sensitive to the structure of the data to be encoded will therefore always produce better results.

The DEAPspace encoding scheme is characterized by the following features:

- All data types are byte-aligned; encoded data (with the exception of booleans) will only take up as much space as absolutely necessary.
- Service descriptions are hierarchically structured data types that share information at the highest possible level.
- `FormatType` data are encoded in a post-order fashion: first those parts that differ, then those that are shared; for each level specific end markers are used.
- The decoding algorithm skips unknown data format types, and resynchronizes with the next known upper-level format type using the end marker corresponding to the item to be skipped.

6. DEAPspace service discovery

Two assumptions characterize the bulk of existing service discovery protocols: that some stable node exists on which a central server can be installed, and that low round-trip times can usually be expected. Both of these assumptions are no longer valid in our target environment, thus a new protocol is needed.

A centralized system, such as DHCP [5] or Jini [17], provides a simple and robust mechanism for service lookup (once the server has been located). In wired networks, it is reasonable to expect that services like a DHCP server are provided by the same organization that maintains the physical equipment; ad hoc groups cannot be expected to provide such infrastructure. An ad hoc group may be formed by any two (or more) devices that come into proximity with one another, therefore guaranteeing at least one server in such an arbitrary group requires that almost all candidate devices be running that server. In addition to the increased overhead resulting from this practice, such routine deployment of servers means that groups would regularly be formed with more than one server as a member. In this situation, maintaining a centralized approach would require servers to exchange state, and to elect which server is to represent the discovery protocol in each ad hoc group every time the network reconfigures itself. If devices can be members of more than one network, the scenario becomes increasingly complicated.

The second assumption that we must give up is the expectation of low round-trip times. The consequence of this assumption has been reactive rather than pro-active systems as these conserve bandwidth. However, if in a decentralized system the members do not pro-actively cache information about their environment, every service request must be transmitted to all group members. Broadcasts in wireless networks are often scheduled to allow power-saving modules to wake up, and therefore offer slow response time. Furthermore, because reliable broadcast is difficult, requests must be repeated, further increasing response time. In this environment, the delay problems outweigh the advantages of reactive systems, and a pro-active solution is preferable.

Pro-active discovery necessarily involves some spontaneous actions on the part of the devices involved. Scheduling of these actions can broadly be categorized into *regular* and *slotted* advertisements. Examples of systems with regular advertisements include Appletalk [1] and HIPERLAN [6], in which devices send periodic beacons scheduled independently by each device. An example of slotted advertisements is 802.11 [8] in which devices contend for each occurrence of a well-defined beacon slot. The similarity between these categories is that they both require a new member of the group to be present for at least as long as the maximum time between broadcasts for all other member devices before the new device has a complete view of the available services. This requirement means that if devices are to have a view of their environment no more than 10 s out of date, then all devices must broadcast their service offerings at least once every 10 s. When the information being shared are only addresses, as with 802.11, or is on a mostly static, high-bandwidth network, like Appletalk, this trade-off between timeliness and bandwidth is not a problem; in wireless, transient ad hoc networks, it poses a problem. Because regular advertisements put an unbounded demand on broadcast packets, and slotted advertisements cause the expected time for discovery to grow linearly with the number of devices present, a hybrid solution is necessary.

6.1. DEAPspace algorithm

We have established that in some situations each device should build a timely picture, or "world view," of the services offered in its area. One example of where this is helpful is a group of four devices, each offering some set of services. What each device would ideally do is to make a reliable broadcast to the other three, constructing a current world view for each device. Aside from the complexity of reliable broadcast under normal conditions, this approach carries with it the risk that an individual device might not even be certain which other devices are present to receive its advertisement, or when that set changes.

To solve this, we propose that devices broadcast their entire world view. If all listening devices incorporate new elements found in each broadcast into their own views, then occasionally missing a broadcast will not cause significant problems for any particular device because the next broadcast by any device that did not miss the first one will repeat (and update) the same information. In this way, we obtain reliability through continued repetition. Also, as each device revises the contents of the world view before repeating it, the information undergoes a constant slow alteration. It is the awareness of time that separates this algorithm from typical gossiping approaches. Information must not only be shared, but also be timely.

The following is a general outline of device behavior in a DEAPspace network. Embellished letters are used to connect a rule to its use in the sample implementation of Fig. 3:

1. Each node maintains a list of service descriptions.
2. Nodes participate in the advertising of services by a broadcast mechanism.
3. A service is described by a service description that includes at least
   - a time-to-live,
   - the address of the node offering the service.
4. The broadcast mechanism works as follows:
   - Broadcasts are scheduled to occur within regularly recurring time windows, with one node broadcasting in each window. $\Leftarrow \mathbb{A}$
   - The broadcasting node is determined by a random back-off mechanism. $\Leftarrow \mathbb{B}$
   - Nodes that see one of their service descriptions about to time out increase their chance to broad-cast by choosing a shorter random back-off time. $\Leftarrow \mathbb{C}$
   - Before broadcasting its list, a node re-initializes the time-to-live value of its local services. $\Leftarrow \mathbb{D}$
5. Each node processes a newly received service list in the following way:
   - It combines the received list of (remote) services with the set of its own (local) services. $\Leftarrow \mathbb{E}$
   - It updates the time-to-live values of the remote services. $\Leftarrow \mathbb{F}$

In order to give a sample implementation, some configuration information must be decided. The

```
advertise(A) {
    time tout ←getTimeout($X$)
    loop(forever) {
        B ←read(tout)
        if(timed out) {
            foreach a ∈ A                                $\Leftarrow \mathbb{D}$
                if(a is my service)
                    a.expiry ←NormalExpiry
            broadcast(A)
            tout ←getTimeout($X$)
        } else {
            Interval I ←update(A,B)
            tout ←getTimeout(I)                          $\Leftarrow \mathbb{A}$
        }
    }
}

read(t) {
    blocking read from network with timeout t
}
```

```
Interval update(A,B) {                                   $\Leftarrow \mathbb{E}$
    foreach b ∈ B {
        if(b is not my service) {
            if($\exists_{a \in A}$ b.id = a.id) {
                if(b.expiry > a.expiry)
                    a.expiry ←b.expiry                   $\Leftarrow \mathbb{F}$
            } else {
                insert b into A
            }
        }
    }
    if($\exists_s s$ is my service and $s \notin B$) return $X'$      $\Leftarrow \mathbb{C}$
    foreach b ∈ B {
        if(b is my service and b.expiry < minExpiry)
            return X'
    }
    return $X$
}
getTimeout(I) {                                          $\Leftarrow \mathbb{B}$
    pick random value on interval I
}
```

Fig. 3. Sample pseudo-code implementation of the DEAPspace algorithm.

following values are used in Fig. 3 and in later analysis:

- Time-to-live values are reset to a maximum value NormalExpiry.
- Normal broadcast timeouts are taken from range X.
- Short broadcast timeouts, used when a device sees one of its own services missing or about to expire, are taken from range X' strictly less than X. In this case, "about to expire" is defined as having a time-to-live value less than MinExpiry.

Ranges rather than constant values are used for timeouts because the receipt of an advertisement resets the transmit timer in every device, therefore without a random jitter in the setting of the timer all devices would broadcast together. Aside from causing possible problems at the physical level, this would defeat the goal of the algorithm: reduction in the number of broadcasts from each individual device.

If this approach seems reminiscent of pro-active route discovery protocols for ad hoc networks, then it is because of the similar goals: route discovery is the special case of service discovery in which all services are adjacency tables. Not surprisingly, for reasons similar to those used as design points for DEAPspace, the route discovery protocols associated with many ad hoc networking specifications also use pro-active systems. Using the slotted and regular categories defined above, we will now compare the performance of alternative solutions with that of DEAPspace for a similar payload.

6.2. Timeliness

For a fair comparison of timeliness, it is important to use configurations that cause comparable network traffic. Thus, timeliness comparisons will be made under conditions where the DEAPspace algorithm sends one broadcast (typically having n entries) about once for every n broadcasts (having one entry each) of the regular algorithm. There is some overhead associated with sending a broadcast packet, at the least a packet header containing a broadcast address, and we can approximate this overhead as being about the same as that caused by the timestamp required for each entry in DEAPspace, justifying this equivalence. Let us now measure network load in terms of advertisements per second, where one advertisement describes one device's capabilities.

As we assume that service discovery is implemented above an existing protocol, we expect packets to be delivered correctly or not at all (i.e., no bit-errors will occur). This, combined with the above load assumption, means the offered load will be about the same for regular, slotted, and DEAPspace discovery [12].

The time for acquisition (discovery) of available services is improved with DEAPspace because more information is being transmitted when one device broadcasts its world view than when each device broadcasts its own information. The additional information transmitted is the feedback to the member devices about what is known about them by at least one other member of the network.

For both slotted and regular broadcast classes of advertisement schemes, a period T can easily be established for the expected time between advertisement repetitions. Suppose we want to tune parameters to perform well with 10% (uncorrelated) packet loss probability and groups of up to six devices. It is now informative to consider what happens when the sixth device encounters an existing group of five others. For the sake of simplicity, only a regular scheme is shown here. In general, regular schemes offer faster discovery than slotted ones, so this is a reasonable comparison.

Some quick math shows that the probability that all five correctly receive the first broadcast is $(1-0.1)^5 = 59\%$, by the second broadcast, it is $(1-(0.1)^2)^5 = 95\%$, and after three broadcasts, it is 99.5%. In the other direction, the new device will learn about all five existing devices with the same probability, except it will usually take slightly longer for all devices to have had a turn at broadcasting.

Now consider DEAPspace under the same conditions. Let us define the newly arriving device as A, the group of five devices already in steady state as G, and particular members of G as G_i. Notationally, let us refer to the average values chosen from X and X' by a single device as $\overline{X}$ and $\overline{X'}$, respectively, and the average lowest values chosen from X and X' out of five choices as $\tilde{X}$ and $\tilde{X}'$, respectively. For simplicity, assume the range of X $(X_{\max} - X_{\min})$ to be less than $X'_{\min}$, so a late transmission will always occur before a new round starts.

Initially, both A and G broadcast periodically, with independent periods. Once they have come within range of each other, either A or some G_i will

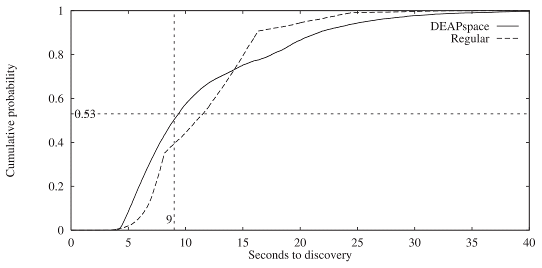


Fig. 4. Probability $P\{E\} = P\{$*all devices have an accurate world view*$\}$ as a function of time for the DEAPspace and the regular algorithm.

transmit first. Under ideal conditions without packet loss, one will transmit, the other will receive the transmission, choose a timeout from X', transmit next, and that transmission will also be received, resulting in all devices having an accurate world view. With 10% packet loss, this will succeed in only about $0.9 \times 0.9^5 = 53\%$ of the cases and can be expected to take about $\frac{1}{3}\overline{X} + \overline{X'}$ time. [3]

This behavior is confirmed by the simulation results shown in Fig. 4, which compares the probabilities of event $E = \{$*all devices have an accurate world view*$\}$ as a function of time for the regular and the DEAPspace algorithm. The results have been obtained with 10,000 simulation runs assuming 10% packet loss. The parameters for the DEAPspace algorithm were chosen as $X = [12, 15]$, $X' = [4, 5]$, NormalExpiry $= 115$ s, and MinExpiry $= 15$ s, resulting in average time between advertisements of 8.16 s. The advertisement period of the regular algorithm was chosen equal to this average of 8.16 s to make the comparison fair in terms of network load. Observe the good agreement of the simulated DEAPspace curve and the analytically derived intersection of ordinate $\frac{1}{3}\overline{X} + \overline{X'} = 9$ and abscissa $P(E) = 0.53$. Moreover, the DEAPspace algorithm achieves the E at about the same rate as the regular algorithm.

To analyze the remaining 47% of the cases, let us consider first some properties of the environment under consideration:

[3] The expected minimum for two random variables uniformly distributed on [0,1] is $\frac{1}{3}$, so the expected time until either A or G broadcasts is about $\frac{1}{3}\overline{X}$ (actually slightly less because the expected period for G is $\tilde{X} < \overline{X}$).

- At least one member of G will virtually always ($1 - 0.1^5 = 99.999$) hear a broadcast from A (assuming that loss is independent), but all members of G will hear any particular broadcast from A only about $0.9^5 = 59\%$ of the time.
- In general, if a device does not receive a transmission, it will be the next to transmit because it will not reset its timer. (The exceptions to this are infrequent, so we ignore them for now.)
- During time $X_{\max}$ following a transmission by any device, all devices will have either sent or received another transmission.

As long as A fails to receive the broadcasts from G, it will continue to broadcast its own local list with a period of $\overline{X}$. Each of these will trigger at least one member of G to choose a timeout from X', and send its own list. Some G_i that did not receive the broadcast from A might broadcast earlier, but some member of G will certainly broadcast its list within time $X'_{\max}$ of the broadcast from A. This implies that the DEAPspace algorithm allows G to discover A at about the same rate as the regular algorithm does (about one try per round) but allows A to discover all of G faster than the regular algorithm does, because the rounds automatically shrink while the environment is changing. This behavior is illustrated in Fig. 5, which compares the probabilities of the events $E_1 = G$ *has discovered* A and $E_2 = A$ *has discovered* G as a function of time for the regular and the DEAPspace algorithm. The simulation parameters are identical to those given for Fig. 4.

The discovery behavior can be considerably improved if the underlying physical layer can provide a trigger indicating that a device has "joined" a network (e.g., when it has synchronized with the spreading sequence of a spread-spectrum system and the MAC layer has been enabled). Fig. 6 shows the behavior of the algorithms with and without trigger. The regular scheme benefits from the shorter time for G to discover A, but has exactly the same time for A to discover G, because G does not respond to the new arrival. In contrast, with the DEAPspace algorithm, G will recognize the arrival of A, and respond by switching to shorter timeouts.

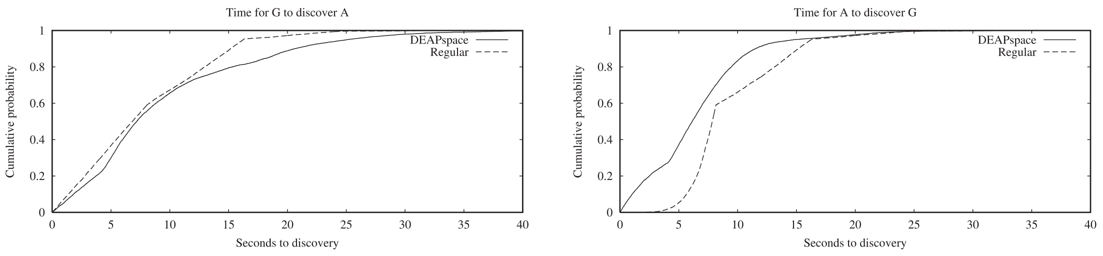


Fig. 5. Probabilities $P\{E_1\} = \{G\ \textit{has discovered}\ A\}$ and $P\{E_2\} = \{A\ \textit{has discovered}\ G\}$ as a function of time for the DEAPspace and the regular algorithm.

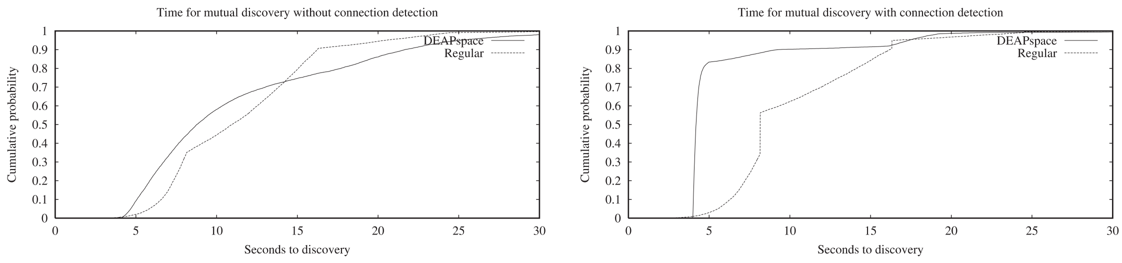


Fig. 6. Probability $P\{E\} = P\{\textit{all devices have an accurate world view}\}$ as a function of time for DEAPspace algorithm and regular algorithm with and without triggering on connection detection from physical layer.

7. Wireless communication platforms

DEAPspace applications are networked applications executing across two or more nodes connected via wireless communication links. In Section 2 we identified a set of characteristics of DEAPspace applications that have profound implications on the underlying wireless communications infrastructure.

As our research focus has been on networking protocols and algorithms rather than the design of a new wireless technology, we have investigated a number of candidate technologies for their suitability with the requirements of DEAPspace.

7.1. Requirements

As DEAPspace applications are spontaneous and ad hoc in nature (in contrast to wireless LANs, where infrastructure networks are the common case), we require a *distributed* MAC *scheme.* Centralized MAC schemes are not well suited for ad hoc environments with mobility, for instance, take two wireless networks each with its own "master" that initially are out of range: as they move within range of one another, one of them must be "dissolved" and its member nodes moved to the other network.

Given the pervasiveness of information appliances, the density of nodes in a wireless network can be very high. Depending on the environment and the wireless range, the number of devices that can potentially interact with one another at any given instant can easily exceed 100 (imagine public areas such as airports, train/subway stations, stadiums, etc.). Thus, the MAC should possibly support on the order of a *few hundred high-rate channels or simultaneous users per band;* furthermore, all nodes should experience a *fair and gradual degradation in throughput* with increasing number of nodes.

Another issue is the *time* it takes *to detect a network and attach to it.* The transient nature of DEAPspace applications, combined with the unawareness principle underlying hidden computing, limits the time available for application-related transactions. Basically, the interaction window shown in Table 1 sets the upper time limit on the window available not only for detecting a wireless network and joining it, but also for discovering peer application entities and executing application transactions. Thus, we require minimal network detection/attach time combined with high-data rates. A further requirement is the capability to broadcast at the physical layer as this helps implement efficient and prompt service discovery mechanisms. Tempting as it may be, we cannot change the relative velocity v of the user (i.e., slow her down to increase the interaction window) as this would violate the unawareness principle. However, the interaction window t_W increases linearly with the wireless range r, which suggests making the range larger. Yet, this implies a higher transmit power, which adversely affects the battery lifetime of the information appliance and blurs the location context, which in turn reduces the precision of location-sensitive semantics. Ideally, the wireless range should be adjustable via some physical layer management interface.

The ad hoc nature of DEAPspace applications demands that any two nodes can interact without prior configuration of details, such as the physical address of devices or shared secrets enabling security mechanisms, about the remote node. Wireless communication must be possible without any prior configuration of device addresses. Link-level security should be defined, but its use should be optional and/or negotiable. This would meet the requirements of ad hoc application scenarios as well as "canned" applications between a user's private devices (laptop, cellular phone, PDA).

The end-user should not need to align devices to enable wireless communication between them. The wireless range should be omni-directional and not line-of-sight.

With proximity-bound service provision, the physical separation between client and server node may be a useful criterium to accept a service or arbitrate between otherwise equivalent service offerings. The wireless subsystem should be capable of estimating the relative distance to neighboring devices and offer a way to access this measure via a physical layer management interface.

The support of audio and/or video applications requires quality-of-service guarantees (throughput, latency, jitter).

Besides these DEAPspace-specific requirements, there is a set of requirements common to environments with low-cost, battery-powered appliances. In particular, the technology should exhibit a high level of VLSI integration, ideally resulting in a single chip implementation of the entire subsystem. Power consumption must be kept at a minimum to increase battery lifetime. The cost to add wireless connectivity must be incremental to the total cost of typical consumer-oriented end-user appliances. The technology must be deployable in license-free frequency bands available around the globe.

Below we shall take a look at currently available technologies and evaluate them in the context of the requirements described above.

Bluetooth wireless technology is an industry standard for small-form factor, low-cost, short-range radio links between mobile PCs, mobile phones and other portable devices. It is developed by the Bluetooth Special Interest Group, an industry group consisting of 3Com, Ericsson, IBM, Intel, Lucent, Microsoft, Motorola, Nokia and Toshiba.

Although Bluetooth looks like an ideal candidate for the wireless infrastructure of DEAPspace it does not stand up to scrutiny vis-a-vis several of our requirements. The reason for this is that, initially, Bluetooth was designed to be a "cable-replacement technology", which has profoundly influenced the design of its physical and media access control layers.

At the physical layer, Bluetooth is a point-to-multipoint technology using a centralized access control scheme with a master controlling the time-division duplex traffic in the so-called *piconet*. The number of nodes that can be active concurrently is limited to eight (including the master). This number can virtually be increased via time-sharing by *parking* and *unparking* slaves. Obviously, this creates overhead at the link management layer. A device that actively seeks a connection to some other device not part of its piconet becomes by definition the master of a new piconet. Either it has to give up membership of the original piconet or it can remain a (parked) slave in the original piconet and toggle between its two roles. Support of this scatternet situation renders the link management task quite complex in the general case. The corresponding issues still have to be sorted out and therefore link management policies for scatternet situations are missing from release 1.0 of the Bluetooth specification. Bluetooth has no true broadcast mechanism. Broadcasts only work among members of a piconet. In effect, the broadcast is a multidrop communication from the master to its slaves. A direct broadcast from a slave to its master and all other slaves is currently not possible. Discovery of devices within radio range relies on the so-called *inquiry* procedure. Depending on the number of active channels, inquiry can take up to 30 s (at least 10 s). A connection can be established only after the address of the Bluetooth device has been learned via inquiry.

In conclusion, Bluetooth is ill suited for the peer-to-peer, spontaneous and transient nature of DEAPspace applications.

A second candidate is a technology based on the IEEE 802.11 standards. The 802.11 working group for wireless local area networks (WLAN) has successfully produced a single standard for the medium access control sublayer and several standards for the physical layer. IEEE-802.11-compliant products have been available from a number of vendors for some time now and, since the introduction of the 11 Mbps version WLAN, has increasingly become a viable alternative to Ethernet and token-ring LANs.

IEEE 802.11 is clearly not a technology designed for short-range (<10 m) networking between information appliances. For that, its cost, range and power consumption are too high. However, having a distributed media access control mechanism, it is well suited for ad hoc networking and supports virtually an unlimited number of nodes in the same network with up to 20 nodes talking concurrently. It does support broadcasts and the time to detect and join an existing network is low. But it lacks support for isochronous traffic. Some care is required when establishing a network to preclude ending up with the situation that two devices each establish their own network and will never see one another. Apart from chosing a common SSID (service set identifier) no configuration is required. Interoperability is only possible between 802.11 nodes sharing at least one physical layer implementation. In particular, frequency-hopping spread spectrum (FHSS) and direct-sequence spread spectrum (DSSS) devices cannot interoperate.

Alation Systems, Inc., specializes in developing technology for the wireless home network. Their *HomeCast Open Protocol* (HOP) is a proprietary technology with similar design points as Bluetooth. The technology ships in the Wireless HomeFree products from Diamond Multimedia.

HOP exhibits a prompt device discovery of the order of less than 2 s and provides physical layer broadcast capabilities. It uses a CSMA/CA host-based software MAC scheme and a FHSS low-power radio in the 2.4 GHz ISM band. The nominal range is 150 feet. The aggregate data rate of 1 Mbps is rather low and limits the number of nodes that can be active in a network to around 10. There is no support for isochronous traffic.

IEEE 802.15 is a working group (WG) of the IEEE 802 LAN/MAN Standards Committee and develops wireless personal area network (WPAN) standards for short-distance wireless networks. The 802.15 WG consists of multiple task groups (TG). The IEEE P802.15 TG1 is deriving a WPAN standard based on the Bluetooth v1.0 specification. Unless the standard developed deviates considerably from its base, the same objections as for Bluetooth apply. The IEEE P802.15 High-Rate Study Group (HRSG) for WPANs is a Study Group (SG) with the objective of developing a physical layer (PHY) and MAC layer specification for high rate, low-complexity, low-power consumption wireless connectivity.

The HRSG is still at an early stage, and it will be several years until a final standard will be adopted and complying products will become available. The HRSG is currently soliciting inputs for applications and technical proposals. The application scenarios submitted will be used to define of a comprehensive set of technical criteria. The technical submitted proposals will then be evaluated (and refined) in terms of these technical criteria.

7.2. Summary of wireless communication technology

The development of short-range, low-power, low-cost wireless technology suitable for connecting information appliances still is a relatively young discipline. So far, Bluetooth is the only available technology with a clear focus on this application area. Unfortunately, if falls short in terms of spontaneous, transient, peer-to-peer, networking of mobile information appliances, as required for DEAPspace. Alternatively, 802.11 might possibly meet the bill. Cost for 802.11 components can be expected to decrease further in the near future, which leaves the high-power consumption as the main obstacle to its deployment in hand-held, battery-powered appliances. Finally, the standards developed by the 802.15 HRSG (and the respective technologies) offer the opportunity to address the above issues and correct the deficiencies identified with existing technologies.

Owing to the lack of an ideal technology at present, we have chosen 802.11 WLAN components as a substitute for our DEAPspace prototyping work. Overall this choice offers the best compromise for our set of requirements.

8. Related research work

The core of our work revolves around prompt and efficient service discovery in transient ad hoc networks of pervasive devices. Service discovery is not a radical new topic. Several projects have addressed the problem of discovering or locating resources and services in a network. Best known approaches are probably Sun's Jini [16,17] and the Microsoft-driven Universal Plug and Play (UPNP) project [4], others are the IETF's Service Lookup Protocol (SLP) [14], the Salutation consortium [3], and HP's JetSend [7]. Of these approaches, Jini probably is the most-advanced project currently, followed by UPNP.

Considering its origins, it is not surprising that Jini depends on Java and Java's remote method invocation (RMI) concept. Jini defines services in terms of Java interfaces and, thus, uses an application program interface (API) contract model. Service discovery relies on Jini's Lookup service (LUS), where a device wanting to share a service with its environment registers this service with a LUS in its environment by submitting Java code to the LUS. A device wanting to use a service offered by another device has to contact the LUS and query it for the desired Java interface, then it has to download the posted code to access the service. To find an LUS, device either use a well-known address or inquire for an LUS through IP multicast. With Jini, access to devices that do not support a Java virtual machine occurs through a *proxy* that knows how to talk to the device and acts on its behalf.

In contrast to Jini, the Microsoft-driven Universal Plug and Play project is specified independently of the underlying platforms and languages, as UPNP uses a protocol contract model. Consequently, devices that desire to interact with other devices do not need to download code but instead need logic that implements a given protocol. Services are represented through *profiles* that define the contract between client and service. Service discovery is taken care of by the Simple Service Discovery Protocol (SSDP): clients request a service either through IP multicast or by querying an SSDP proxy. In the case of a multicast query any device providing the requested service needs to respond to the request; in the case of a proxied environment the proxy will respond. Also, in a proxied environment each device offering services needs to register itself with the SSDP proxy. Clients can find out more about a service by requesting a service *schema* from the service-providing device, which is a structured data description that contains more information about a service.

Both Jini and UPNP employ a *pull-model* for service discovery: a client interested in a certain service either needs to query its environment or a designated, central service to find a service provider. Compared with Jini, UPNP offers greater flexibility as it requires no LUS to be available and instead can work in truly peer-to-peer fashion. While Jini's API-based contract model is convenient and easy to work within Java-based environments, UPNP's protocol-based contract model allows a much larger range of devices and platforms. Also, Jini's use of downloadable code introduces code quality, security, and quality-of-service problems. Of the two approaches, however, Jini is currently widely used, whereas UPNP has yet to make its appearance in code form.

Similar to Jini and UPNP, Salutation uses a central service lookup server, called *Salutation Manager*. In contrast to Jini and UPNP, Salutation requires neither a specific platform (such as Java in Jini's case) nor a specific network technology (such as HTTP or UDP in UPNP's case). The IETF SLP is similar in scope and operation to UPNP as it also uses a central directory agent to find out about and locate services.

In the context of transient ad hoc networking Jini, UPNP and the other discovery protocols suffer from their use of a pull-model-based service discovery: by the time a device has issued its service request and received an answer, it might not be able to invoke the service as it has left the range of the service providing environment. Jini in addition introduces a considerable overhead with its dependence on Java and sole use of Java RMI.

9. Summary and outlook

Ad hoc networking and spontaneous interaction between small and mobile devices over wireless connections will be the key features of new pervasive applications which we can expect to become real over the next few years. The DEAPspace project presented in this paper is positioned as a first step towards this goal. Starting from a description of new application scenarios, we have derived the most important requirements to be met by a framework supporting the development of pervasive applications. This paper has explained the most fundamental concepts of DEAPspace: its service model, compact and efficient service description and matchmaking, and a new push-based service-discovery algorithm. The discussion of wireless communication technology currently being developed and related research has shown that transient as-hod networking is a relatively young discipline in which more work remains to be done.

9.1. Current Status

A framework based on the DEAPspace concepts as described in this paper has been implemented. This framework allows the development of distributed applications based on ad hoc transient networking whose components run on devices connected via a broadcast medium. The primary implementation was done in Java; for devices which cannot run a virtual machine, a subset of the framework has been implemented in C.

The framework enables deploying different wired or wireless communication technologies. To accomplish transparency from the concrete medium, the framework provides abstract interfaces shielding an application from the specifics of the concrete network. Currently, we provide code to run DEAPspace applications over a simulated network as well as TCP/IP and 802.3 MAC.

Several demonstration applications are running, among them an application enabling downloadable GUIs for devices which do no permit direct management access. Here, a controller responds to management requests using Wireless Markup Language (WML) decks. The requesting entity is a WML browser visualizing the controller responses and supporting GUI-based interaction.

9.2. Outlook

Important and challenging areas of further work are: security in ad hoc transient networking, location-aware and multidevice applications as well as their configuration and management.

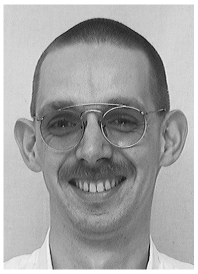
The aspect of security in transient networks constitutes a field of research where very little has been done so far. In particular, access control and, to some extent, information flow control are the most-needed mechanisms. An open issue in this context is how to introduce, describe, and qualify different levels of trust.

Transient ad hoc networking and spontaneous interaction enable the development of new programming models and distributed applications. In terms of the behavior of these applications, relayed and choired scenarios can be distinguished. Relayed applications sequentially utilize services as they become available and store temporary results. In contrast to that, choired applications require the presence and interaction of multiple services at the same time.

References

[1] Apple Computer Inc., Inside Macintosh: Networking, 1994.
[2] N. Brown, C. Kindel, Distributed component object model protocol – DCOM/1.0, Internet Draft. Technical report, Microsoft Corporation, 1996.
[3] Salutation Consortium, Salutation architecture specification, http://www.salutation.org/specordr.htm, June 1999 (cited 20 March 2000).
[4] Microsoft Corporation, Universal Plug and Play: Background, http://www.upnp.org/resources/UPnPbkgnd.htm, 1999 (cited 17 March 2000).
[5] R. Droms, Dynamic host configuration protocol, Technical Report RFC 1541, Bucknell University, October 1993.
[6] ETSI, High Performance Radio Local Area Network (HIPERLAN); Type 1; Functional Specification, October 1996.
[7] Hewlett Packard, What is hp jetsend?, http://www.hp.jetsend.com:80/products/hp/overview.html, 2000 (cited 20 March 2000).
[8] IEEE Computer Society, LAN MAN Standards Committee, Wireless LAN Medium Access, June 1997.
[9] International Telecommunication Union, ITU-T Recommendation X.208: Open Systems Interconnection – Model and Notation – Specification of Abstract Syntax Notation One (ASN.1), 1988.
[10] International Telecommunication Union, ITU-T Recommendation X.690: Information Technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), December 1997.
[11] International Standardization Organization, ISO/IEC IS 10746-3 Reference model for open distributed processing – part 3: Architecture, Technical report, ISO/IEC JTC1/SC21/WG7, March 1995.
[12] Michael Nidd, Service discovery in DEAPspace, Technical Report RZ6906, IBM Research, Zurich Research Laboratory, January 2000.
[13] Object Management Group, The common object request broker architecture: Architecture and specification, Revision 2.0. Technical report, Object Management Group, 1995.
[14] Charles Perkin, Slp white paper, http://playground.sun.com/srvloc/slp_white_paper.html, May 1997 (cited 20 March 2000).
[15] R. Srinivasan, XDR: External Data Representation Standard, IETF RFC 1832, August 1995.
[16] Sun Microsystems Inc., Jini connection technology, http://www.sun.com/jini/ (cited 28 February 2000).
[17] Sun Microsystems Inc., Jini Architectural Overview, 1999.
[18] Wireless Application Protocol Forum, Wireless Application Protocol, Binary XML Content Format Specification, version 1.1 edition, 1999.
[19] World Wide Web Consortium, Extensible Markup Language (XML), February 1998.

Reto Hermann obtained the Diploma in Electrical Engineering at the Swiss Federal Institute of Technology, Zurich, in 1983. Subsequently, he joined the IBM Research Division, Zurich Research Laboratory, as research staff member. From 1983 to 1992 he worked in the area of signal processing for digital magnetic recording systems. Since 1993 his research focus has moved to computer science where he has done work in mobile computing and smart card technology. His present technical interests are pervasive computing, wireless personal area networking and mobile internet technologies. He is a member of the IEEE and IEEE Communications Society.



Dirk Husemann joined IBM's Research Division in 1996 and has since then been working on pervasive computing projects. Currently he is involved in the DEAPspace project as well as in datacasting over digital audio broadcast channels. He received both an MS (Diploma, 1991) and a Ph.D. (Dr.-Ing., 1995) in Computer Science from the University of Erlangen-Nürnberg, Germany. Husemann is a member of the IEEE, IEEE Computer, USENIX, TUG, and the German GI.

Michael Moser received both his Diploma in Electrical Engineering and his Ph.D. in Technical Sciences from the ETH Zurich, Switzerland. He is a research staff member with the IBM Zurich Research Laboratory, where he is currently involved in various pervasive computing projects. His research interests include distributed and intermittently connected systems, advanced user interfaces for portable devices as well as program design observation and execution monitoring tools.

Michael Nidd is currently writing his Ph.D. dissertation on the topic of service discovery in transient ad hoc wireless networks, associated jointly with Institut Eurécom and the Swiss Federal Institute of Technology (EPFL), and working at the IBM Research Zurich Laboratory. He holds BMath and MMath degrees in Computer Science from the University of Waterloo.

Christian Rohner is working towards his Ph.D. in Computer Science from the Swiss Federal Institute of Technology (ETH) Zürich, from where he also received a Diploma in Electrical Engineering in 1997. He is doing his work at the IBM Research Zürich Laboratory. His research interests include security and communication issues in ad hoc networking.

Andreas Schade joined the IBM Zurich Research Laboratory in 1994 where he currently is part of the pervasive computing group. His research interests include distributed systems, applications and their management. He holds a diploma degree (Dipl.-Inf., 1994) and a doctoral degree (Dr. rer. nat., 1998) in Computer Science, both from Humboldt-University Berlin, Germany.