Peer-to-Peer Coordination Framework (P2PC): Enabler of Mobile Ad-Hoc Networking for Medicine, Business, and Entertainment [1]

(work in progress)

Ekaterina Chtcherbina, Marquart Franz

Siemens AG, Corporate Technology
Software and Engineering Architecture.
ekaterina.chtcherbina@siemens.com, marquart.franz@mchp.siemens.de

Abstract

Growing market of mobile devices with the wireless networking and extended processing capabilities leads to a wider range of new information appliances. A mobile device can support the user in various ways in different situations by exchanging information with other devices. The ability to form ad-hoc groups of communication partners, either in the way of inter-person communication groups, inter-object communication groups or mixed person-object communication groups by using mobile devices requires new concepts for coordination between these devices. Ad-hoc communication and coordination perfectly harmonize with the peer-to-peer concept. In general no infrastructure to reach a central authority is available, so communication and coordination have to be done directly among the participating objects.

The paper gives an overview of the Peer-to-Peer Coordination Framework that provides an infrastructure for ad-hoc applications in areas of medicine, business and nomadic community support. A detailed overview of this framework that is built on the top of JXTA-platform explains the main mechanisms of ad-hoc networking in mobile environment. The Peer-to-Peer Coordination Framework stresses the notion of peer-to-peer. Coordination must be done not only between human beings but also between arbitrary objects. Even non-active objects like posters or shop windows should be able to take part in peer-to-peer coordination.

We would like to discuss the Peer-to-Peer Coordination Framework and to stress perspectives of Peer-to-Peer technology that brings intelligence to mobile devices on one hand and enables ad-hoc networking in the areas of high importance such as health, emergency and business on the other hand.

Introduction

Very often users need more intelligence and more situational behavior from software systems than the software can provide today. This demand is especially critical in medicine and business areas. We can take an example of emergency scenario where time factor is extremely important and where software system must provide an immediate reaction on the situation when somebody needs help. The system must be able to react not only on the user's request for help but also to receive this information from appropriate distributed sources (sensors), excluding the human interaction factor. System also receives information about the location of "patient" from geographical sensors and finds the nearest point of help - the emergency car that requires less time than other emergency cars to react on the request and to come to the patient. In this scenario we see several processes that take place like gathering information about the environment, processing this information, finding available resources that can participate in solving of the problem, and communication of input information needed to resolve the situation to the dynamically found resources. This scenario is quite intuitive but on the other hand it requires certain mechanisms that make this scenario working – among others automated recognition of the demand (situational analysis) and building of an ad-hoc network based on the immediately available participants and resources. In this scenario we can see how ad-hoc features can dramatically improve the ability of the software system to support human beings in various situations.

Analysis of software platforms for medicine, education, business and entertainment (the last but not the least!) leads us to a conclusion that too few aspects of ad-hoc behavior are addressed by the existing software solutions. And the reason is not in the lack of functionality that is offered by those systems but in the systems design,

[1] In Cooperation with University of Linz, Austria

which is normally based on a fixed centralized infrastructure and is restricted to the usage of a prior known resources and facts. Moreover, systems behavior is static in a sense that there is no adaptation to a situation, meaning that the system provides a fixed set of outputs mapped at the design time to a pre-defined and fixed set of input choices, wherever the system is required to react by the active side (known as actor). Let us look to the typical user scenario in the existing software systems – the user initiates the process in the software system, system starts some processes and outputs the result. In the described scenario the user plays an active role while software is a passive tool that helps the user by automating of some processes but still being a passive partner of the user. Of course, initiating of processes can be automated to some extent as well, for example by applying scheduling and event handling mechanisms (system actors). But those systems are inflexible in terms of ability of changing of activity caused by external reasons. In order to enable ad-hoc features of the software system, challenges of receiving and interpretation of environmental information as well as adaptation of the system's behavior based on this information must be addressed. As a result new approaches for design of software systems and new infrastructures must be found.

The main issues addressed by peer-to-peer technology – discovery mechanisms, dynamic building of communities, distributed data exchange and security concerns in the distributed environment - are extremely interesting in the context of discussed topic. As we will see later, peer-to-peer technology enables ad-hoc communication on the application level. Anyway, there are limited mechanisms for gathering of the environmental information in peer-to-peer systems. Although peer-to-peer mechanisms allow for getting information about the resource availability, they do not provide information about external factors such as description of the situation in which software system is located and which can influence the output. With the Peer-to-Peer Coordination Framework (P2PC) we address an issue of context awareness and how context information can be interpreted by the system in order to provide expected functionality adapted to the situation. Therefore, a very interesting combination of pro-activity enabled by peer-to-peer mechanisms together with situational intelligence enabled by integration of context-awareness mechanisms is investigated in this paper. We suggest an approach to the systems architecture design that allows for building an ad-hoc context sensitive applications, using Peer-to-Peer Coordination Framework as a middleware platform. The framework provides a certain level of abstraction that makes ad-hoc processes transparent for the developer.

Before diving into details of P2PC, we would like to give an overview of challenges targeted by ad-hoc systems as well as description of influence of context information on the systems behavior.

Importance of Ad-Hoc Systems

The term "ad-hoc" is used to refer to any kind of dynamic environment, where network is fashioned from "whatever is spontaneous available". An ad-hoc network is the collection of mobile (not necessarily wireless) nodes without the required intervention of a centralized access point or existing infrastructure. The connectivity between the nodes is dynamic and can be broken as the nodes move about the network [CW02].

Very often discussions of benefits of ad-hoc systems in comparison to standard Internet technologies can be found. Among those benefits are cost saving, direct access, native way of communication, and solution for the problem of bottlenecks in the network. At the same time there is a whole set of scenarios and situations where "ad-hoc" features of the system are a must. Let us look at five reasons for ad-hoc systems to exist.

The first reason is "*No infrastructure*". Here we are talking about situations when infrastructure does not exist yet or the connectivity to the infrastructure is temporarily broken. Places like construction sites, warehouses, streets, harbors, or pure natural sites might be not equipped with IT networking infrastructure. People, on one hand, still desire means of communication even if the infrastructure is not available. Somebody in the mountains, for instance, might require help and want to make an emergency call. Today, if there is no infrastructure, people cannot do that. On the other hand, if there were a possibility just to find other people who are somewhere near the person who needs help, it could help a lot in various situations. Also, when the connectivity to the infrastructure is broken for any reasons but interactions are required, the need of finding other nodes in the network without accessing the infrastructure can arise. A centralized approach cannot help in these situations because it is not possible to access the centralized services without an infrastructure.

The second reason is "*Changing environments*". This is not the same as mobility but refers to the ability of integrating in a new situation without changing a location. Referring to the changing environment we can talk, e.g., about a system that should work in the home environment as well as in the office environment providing a relevant behavior according to the environment's self-description. The system in the changing environment should be able to adapt itself to a new environment and use the network resources provided in this environment without requiring a manual change, i.e. a user interaction. A system should be smart enough to obtain the information for the situational behavior from the environment itself.

The third reason is "*Mobility*". Mobility means that participants of the network are changing their location during the usage of the system. When talking about mobility, often scenarios of cars rushing through the streets of a city or many people walking around in public places like shopping centers or stadiums are presented. But mobility can also mean that a device stands on one place for several hours and is then moved to another place to reside there for another couple of hours. Each time a network node moves (or is moved), the entire topology of network changes. Two nodes that were just close together and could easily interact with each other, might be far apart in the next moment requiring some dedicated "routers" in between in order to establish any kind of communication. Or it might happen that a node moves out of the scope of the network and loses the connection completely, while other nodes come into the reach and want to participate. Ad-hoc systems are able to work under these circumstances and even can handle the additional constraints implied by mobile nodes, such as limited bandwidth and power.

The fourth reason is "*No planning*". This is when the behavior or an event in the future cannot be planned or forecasted. To this extent, an ad-hoc system can be regarded as indeterministic. What looks like the edge to chaos at the first glance, turns out to be in fact a particular strength. A system that is able to react to circumstances that have not been anticipated when it was designed is much more flexible and adaptable. An example is finding a service by given parameters that might reside anywhere in the network. As the availability and accessibility of this service can change continuously, the system cannot plan the time and place when this service will be discovered. We should also keep in mind that even centralized systems are not as deterministic as they seem to be. Especially the response times and sequences in concurrency situations depend on so many factors that they cannot be forecasted reliably.

And the fifth reason is "*Instability*". This does not mean instability in the sense of frequent crashes due to errors in the application or operating system software. We want to refer here to system immanent instability by design, usually caused by external forces out of the scope of the system. Examples are rapidly changing conditions for wireless connectivity (e.g. by interferences or shielding) or fault tolerance for massive hardware failures or disasters. If the system is highly unstable, it is worth being ad-hoc in order to be able switch to another working environment if the problem occurs. This might happen also in industrial scenarios where everything must be highly stable. Anyway, instability happens and systems must be able to react to that.

Ad-hoc systems become more and more important because of the growing mobility of the society and as a result of the growing need of "easy-to-use", "always-in-the-pocket" software systems. If we talk today about the ideal software systems in the area of personal communications, business solutions, or home automation, they need to be "simple", "easy to configure", "always available", "stable when necessary" and "secure". This is what ad-hoc networking is designed for.

Personal communication is a technology area where everybody has his or her own experiences and preferences because it is more or less part of every-day life. In the discussion about peer-to-peer networking these experiences tend to push aside an area that is certainly even more important: the intelligence in devices. More and more formerly "dumb" electrical devices carry sophisticated software systems with them that enable them to get in contact with other systems, receive and process orders. To fully exploit the benefits of this intelligence and the value of the network, it is necessary that the nodes of the system communicate with each other providing also their intelligence to the network. The natural use case is plugging them together and let information exchange start. This is what ad-hoc networking means.

Qualitative Changes in Software by Introducing Ad-hoc Features

Having discussed the reasons for ad-hoc systems to exist, we can now formulate the features that differ ad-hoc systems from traditional ones.

Self-Organization: One of the most important features of ad-hoc systems is self-organizing. The self-organization means the ability of the system to change its structure as a function of participants of the system and its environment taken together. It is important to emphasize that participants only cannot create a self-organizing system because the system does not organize itself independent of the environment. In order to form a self-organizing system all participants have to discover and get the information about the surrounding environment. There are two possible mechanisms of implementation of this idea: one is discovery messaging (pull mechanism) and another is an advertisement messaging (push mechanism). Discovery mechanism is used to get the information about the system environment on demand whenever the participant needs this information. Advertisement is used to publish information that reflects changes of the environment without a demand coming from the system. In fact, discovery or advertisement are rarely used alone, but commonly a combination of them is employed in order to optimize the performance of the system. In that case the system provides a caching mechanism for storing advertisements providing the stored information about the environment on demand whenever requested.

The challenge in a search mechanism is to find a right balance between discovery and advertisement messages as well as the appropriate frequency of those messages. In order to optimize the search mechanism in the

wide area network, a look-up service is required. The challenge for setting up such a look-up service is to integrate a right number of providers of the look-up service within the network in order to optimize again finding of information, stability and storage of information. This is an optimization function with three degrees of freedom.

Flexibility: Flexibility is another feature of the ad-hoc systems. Flexibility means an ability of the system to adapt its behavior to a new unknown unpredictable situation. In other words, the system has to be able to provide a situational behavior, e.g. change the behavior based on the changing circumstances. Ad-hoc system should be able

- To support dynamic service composition
- To change the network if the connectivity is broken
- To react on new service supplies and much more.

The main challenge by flexibility is unstable connectivity and the session hand-over in case of change of the network topology or change of the central access point.

Dynamic Reactivity: Taking in consideration the "mobility" and "no planning" aspects of the ad-hoc system, they need to be dynamic. That means they should be able to provide a continuously productive activity or change. If the environment requires change of activity or presentation of the activity in another form, the system must be able to react on this requirement in a productive and efficient way. This dynamic adaptation capability is required for example for service and device adaptation based on context awareness and local information**.**

Challenges are multiple device support as well as obtaining and actively reacting on the up-to-date environment information.

Openness: Ad-hoc systems need to be open. This is a very simple requirement, which is very difficult to implement. "Open" means accessible from all or nearly all sides. Of course, talking about software systems, we cannot require the system to be completely open for other systems (also taking in consideration technical and security issues), but instead we say that "open" means based on open standards. Open standards is a key that allows more devices and applications to participate in the process of self-organization of the network and later on in interactions, this way increasing the value of the network. In order to become open, the system must be not only interoperable with other systems on the protocol level but also to be able to act as a player in the network. That means providing also the basic cross-platform functionality that will enable networking on the application level.

Decentralization: Ad-hoc systems are often fully decentralized like in the example without infrastructure. They might be "often" decentralized but not necessarily always. Ad-hoc means fashioned from whatever is spontaneous available. If a server is available, it can also participate in the ad-hoc networking. That means, sometimes they can be controlled by a central point (user management, total error handling etc) but they might be able to adapt to a situation where no centralized approach is applicable and each node of the network should be a server and a client simultaneously (a so-called servent). In order to provide this capability ad-hoc systems need fully decentralized search and communication mechanisms.

One of the biggest challenges is a multi-hop routing in the network consisting of mobile devices with the power limitation as well as wireless connectivity with the limited bandwidth. Looking at the business side of the problem, there are no billing and payment mechanisms that would allow the ad-hoc network to use the available resources of separate individuals. This is, at least, a major problem for the practical acceptance in many realizations of ad-hoc systems: How can a user be persuaded to let others make use of his resources, i.e. (mis-) use him as a router? Another challenge in the decentralized environment is decentralized security trust model that would allow the participants of the network to initialize trust concept in a distributed manner [Sch01].

Mobility: Taking again the mobility aspect of the ad-hoc networking into consideration, the systems need to be often mobile. That means being physically mobile as well as on the logical level being capable to be productive in the changing environment. On the application level there is a need to support changing network technologies (LAN, WLAN, Bluetooth, GPRS) and addressing in mobile environment. An intelligent support for wireless technologies is also required such as choosing an optimal multi-hop routing or choosing a TTL for a request based on the network topology and available bandwidth.

The mobility aspect requires a "smart" support for wireless technologies. That means analyzing the problems on the transport level and providing support for them on the application level. One of the problems in the wireless standard IEEE802.11 is that the TCP performance goes down due to the limits of the bandwidth. Ad-hoc mode of the same standard has several problems with the multicasting. In order for the system to work efficiently with those wireless technologies, the system must be aware of the problems and react properly. That also can affect frequency of requests sent to the network to get the desired number of successful deliveries.

Context-Awareness in Ad-Hoc Systems

Context-aware computing is the ability of a software system to discover and react to changes in the context in which it is situated. In order to interpret context we need a world model [BCM02], which is an abstraction of a real world and that defines which information is relevant to the description of a certain situations. Therefore, context is a summary of external real-world factors mapped to the set of entity attributes of the abstraction level called the world model of the software system. This set of input attributes is interpreted by the software system and is reflected in the changing of behavior of the system or in the system output. This interaction of the software system with the environment can be described as following:

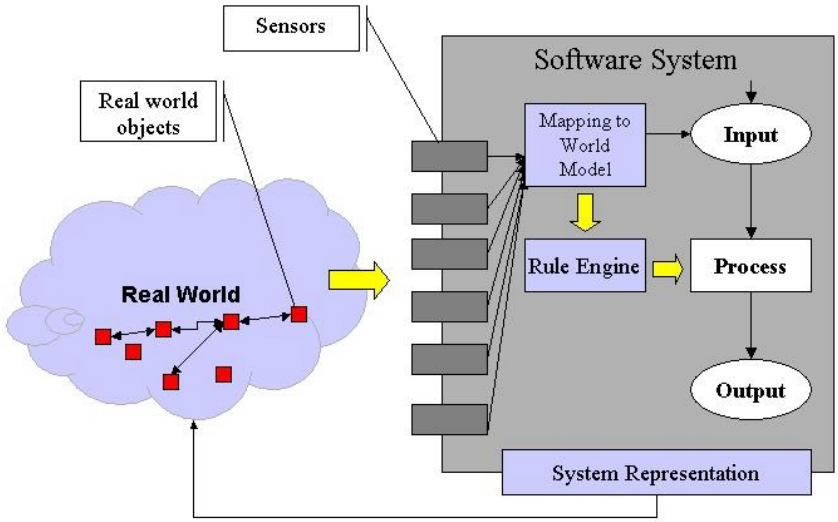

Figure1: Context-Aware Software Systems

Software System participates in the real-world interactions by getting information from the environment as well as changing the environment itself. The information about the environment is gathered by the systems with the help of sensors. The system might be sensing information on the low-level, for example location information, temperature, state, and on the high level, like for example analyzing properties of the surrounding objects and actively finding objects with the required ones. Due to the complexity of relationships between objects and relative nature of object properties in the real world, we need to have a world model that will allow for interpretation of information received from the systems environment. The rule engine of the software system can interpret received information and change system processes. As a result behavior of the system can be changed based on the environmental information. It is important to mention that the software system is not only gathering information but is also actively participating in the relationships with other objects. Therefore, an interface for representation of the software system in the interactions with other objects is required. The software system can generates information that might be actively sensed by other objects.

Based on Schilits [SCH95] "dynamic environment object" definition, as an unspecified collection of data, we also use the term entity to describe an unspecified collection of attributes. "A context information entity is an unspecified collection of context information attributes, possibly describing every aspect of a real world or virtual object".

Attributes encapsulate context information about a specific area, for example location information is described through a LocationAttribute. Attributes are instances of attribute classes, which are defined in a hierarchy, were simple inheritance from other classes, or multiple interface inheritance is possible.

Generally an entity `E` is completely described through the Cartesian Product of its attributes (`A1-An`): `E = A1 ⊗ A2 ⊗ … ⊗ An;`

The description of environment objects through attributes could be compared with the discovery functionality, Schilit defined as one of three major types of functionality. Following information examples could be modeled as attributes for entities:

- Location information (spatial location information, proximity location information)
- Entity Attributes
- Time information
- Information about relationship between different entities:
  - An entity is the *owner* of another entity.
  - A group of entities is *contained* by another entity (could be classified as place)
- Information about the actions an entity is able to perform
  - The entity is able to print, scan, show, send, cipher, … a given data object
- Information about the state of an entity
  - May be on, off, running or *idle* for an object, or alone, busy or sleeping for a person [23]
- Information about personal preferences, predicted out of historic information
  - Historic information for future preference decisions
- Information about well known entities (buddy-lists for example)
  - Grouping information

In the Peer-to-Peer Coordination Framework we have focused on the sensing of high-level information such as entity attributes, which are analyzed by the framework and appropriate actions are initiated based on the characteristics of participants of the ad-hoc network. We will see later how it works.

P2PC: Communication Concepts

With the peer-to-peer Coordination Framework (P2PC Framework) it is easy to develop profile-oriented applications in ad-hoc and mobile scenarios. The most natural way of communication in such mobile ad-hoc scenarios is to communicate directly between the participants. Therefore a peer-to-peer approach is used for direct communication between appliances in this framework. The P2PC framework uses JXTA as the underlying peer-to-peer technology to discover nearby peers and to communicate with them. Project JXTA is an initiative to provide a comprehensive framework for peer-to-peer applications.

If we take ad-hoc systems and their features that have been analyzed in the beginning in one hand, and we take peer-to-peer technology and the functionality that peer-to-peer systems are targeting on in another hand, then we can see that they have very much in common. We would like to show that peer-to-peer platforms such as JXTA might become a middleware for ad-hoc systems that will enable wide range of ad-hoc networking in the future.

Peer-to-Peer networking is an adaptive, self-configuring network, which does not rely on central servers. This is a type of network in which each workstation has in principal equivalent capabilities and responsibilities. Peer-to-peer networking is based on three blocks of basic functionality that enables “equal”-art of communication between nodes in the network: discovery mechanism, messaging and a security trust concept.

Discovery mechanisms: A discovery mechanism (search) allows peers to find other peers and context in the network. Because the centralized look-up service can be unavailable, peer-to-peer systems often provide a fully decentralized search mechanisms (Gnutella [So01], JXTA [Gon01]) or a hybrid semi-decentralized search (JXTA). It depends on a network size and topology (see below).

Peer-to-Peer technology allows for efficient use of resources and robustness in the decentralized environment. It provides mechanisms for finding required devices and resources in the network and to interact with them directly (search and messaging mechanisms). Existing distributed networking technologies apart from Peer-to-Peer (such as DCOM, CORBA, Web Services) currently rely on a priory known directory servers (registry, naming server, UDDI). That requires pre-configuration and limits the flexibility and ability to dynamically react to the change of the environment of the system. Distributed auto-configuration technologies such as JINI also rely on directory servers. Peer-to-peer concept of distributed discovery (plug and play) implemented in UPnP technology [GS00] and in JXTA search mechanism is more suitable for self-organizing networks.

As we have seen, search mechanisms are a key element for efficient and productive self-organization of the ad-hoc systems. The simple case is that only the peers shall be searched that are directly connected to the peer launching the search. This task can be completed in a sequential or even parallel way, depending on the available computing and network resources.

More interesting is the wide area search that comprises peers that cannot be "seen" directly. Several approaches have been realized in existing platforms that either are built on top of the network topology directly or target at matching of profiles or content.

Building on top of the network topology means that the search is related to the degree of decentralization of the network. If there is a central instance where the desired pieces of information are deposited, it can and should be used for the search, for this certainly accelerates the search considerably. In the pure decentralized case, when there is no such server, the search is carried out from one peer to the next or to all that are accessible from it, respectively. This technique is used for example in Gnutella. To avoiding an over-flooding of the network with search requests, each of these requests is assigned a maximal depth, the search horizon. Further optimizations can be done based on the reduction of the depth by introducing connection weights, say according to the number or kind of hops, the round-trip times or similar.

The content-based search follows a different approach. One option there is to designate some peers and save hash-tables about the contents of their environment on them. During a search, only the peers that carry such tables, called the "super peers", need to be queried. This technique is a hybrid solution in the sense that it transfers concepts from the client/server world to a decentralized environment. The disadvantage of hash-tables is that only content with a few attributes can be processed and encoded reasonably. More complex information – as it is common in real-world situations – can hardly be managed this way. An alternative is to avoid searching a peer completely but to rely on it declaring the relevant information on its own. This search technique follows the publisher/subscriber design pattern; each peer published the services it wants to offer to other peers on some sort of "advertisement board". A search algorithm can limit itself to reading these boards and need not waste time searching through the entire peer.

We can also combine both techniques. If all peers describe their offers on "advertisement boards", super-peers can query the services provided by peers in their environment and cache them. In a global view, the search is carried out in a decentralized way from one super-peer to another; in fact a much sparser P2P network than the original one must be combed. Seen locally the search is centralistic because the super-peer plays a server role for the other ones. Such an algorithm is part of the P2P platform JXTA, for example.

The coexistence of different approaches makes clear that there is not such thing like an optimal search strategy. Like in other search problems it depends to a large degree on the application, i.e. on the kind of information that we look for.

Messaging Mechanisms: A messaging mechanism (JXTA, Jabber [Ora01]) has several problems to overcome in the peer-to-peer context. The first one is a multi-hop routing, which is the problem of the mobile environment. The second one is disruptive connectivity and firewalls. Current peer-to-peer platforms provide caching mechanisms (relaying) for overcoming firewalls problem. Caching mechanism allows the system to store information about other peers in the caching-service of the local area network that is used for accessing those peers from another local area network.

The multi-hop routing for the messaging in the ad-hoc environment is a hot topic. There is a need for algorithms that would allow for efficient multi-hop routing, although taking into consideration the mobility aspect and power consumption at the end devices. As long as multi-hop routing is not supported on the transport layer, the network can be built today only based on the connectivity within the transmitting range of the hardware (e.g. 100 m for the wireless LAN). If the node is out of the range, there is no possibility to send a message, although a rule of transitivity could be applied in practice – if node A sees node B and node B sees node C, this implies that node A sees node C. But for that a smart routing algorithm is required. The question is how to send the message avoiding loops and infinite routes? Choosing a node that should be the next node the message is sent to, certain rules should be applied. One of the considerations is to send the message to the node of high potential that is chosen from all available nodes also applying the ad-hoc rules. The node with high potential is just a node that is capable of resending the message. At the same time the route must be cached at each node of the route in order to record the way for further usage like sending the acknowledgement for the message back. But the problem with the one-route algorithms is that the branch can come to end somewhere and no nodes will be available at some point. That means the message won't reach the goal but will be lost in the network. These considerations make clear that multi-hop routing cannot simply follow the techniques that are employed in wire-line networks, but should be based on a multi-way approach that is an optimal one for non-deterministic system, where the behaviour and the state of the system cannot be predicted, so the probability of delivery of the message with the one-way algorithm is very low.

Open Issues: A security trust concept is also provided by peer-to-peer technology, as a concept of groups where all peers are trusted within one group. This group concept (JXTA, Groove [SO00]) allows for solving the problem of decentralized initial trust establishment. Anyway, the trust establishment is still an issue for open networks. The trust of the single participant of the network is based on the experience of the node and gathering the experiences together requires an algorithm for distributed voting.

Flexibility is often supported by easy or no configuration required from the user of the system. And this self-configuration mechanism is also available in the peer-to-peer concept (UPnP, JXTA).

There are still several open issues that need to be solved before the ad-hoc networking will become a reality. As we said, ad-hoc systems require dynamic adaptation capability. This includes location adaptation, service adaptation to a device, taking in consideration context awareness information. These features are not supported by any of the existing Peer-to-Peer platforms.

Also mobility aspect is not fully supported by Peer-to-Peer platforms. The addressing issue has been partly solved in some platforms. For example, in JXTA the peer receives a peer_id, which is an abstraction of the peer, used to address requests to this peer. Peer_id is independent from IP-addresses or any other network-based addressing mechanism. The existing mechanisms of "on the fly" address assignment like DHCP, autoIP, mobile IP also helps to minimize the configuration efforts. But name to address mapping is not really solved because the problem of finding an exact peer is still a problem in peer-to-peer networks. Another problem is a hand-over when changing the network or type of the network – vertical and session hand-over. The peer-to-peer platforms today are not able to handle these situations. If the connectivity changes from WLAN to Bluetooth the system today is still not able to adapt to this change.

P2PC Framework Components

Distributed architectures require a principle decision whether to choose a centralized registry approach like e.g. JINI does (see [FAH02a]) , or to choose a decentralized approach like many peer-to-peer frameworks do. A centralized registry stores information about the individual peers in the network, and peers need to advertise their description to this registry. Coordination in this case is quite simple and is done over the registry. Decentralized approaches presume that information about peers is decentralized and at each moment only a subset of this information is available. There is no dependency on other peers, like there is in client/server environments, on one hand, but on the other hand there is dependency on network topology and network technology. Ad-hoc systems have a decentralized nature, although they might sometimes rely on centralized registries. This led us to a decision to use a fully decentralized approach. We would refer to peer properties as to the peer profile. Profile describes an object using the world model of P2PC. Based on the peers' profiles, coordination of communication between peers is also decentralized. There is no need for fixed pre-defined infrastructure.

In the real word objects are typically the part of more than one "life – domain". When describing an object or user with a profile, it is required to allow the user the definition of several roles, representing these "life – domains". Such a role could be "member of a tennis-club" or "member of the conference". This information not only allows for interpretation of the object role dependent on context, but provides also security mechanisms for object authentication. Thus, objects that have the same role can be grouped together based on the trust of objects with the same role. Besides that, the framework allows multiple applications to be run at once. Each of these applications would typically perform a task for one role. The definition of more than one role allows the user to specify data only for that special role, it is not intended that data specified for the role A must be necessarily used for the role B.

Along with other information, context definitions can be provided in profiles. It is essential that coordination between peers sometimes should only be performed when certain context conditions are fulfilled. For example a user could tell an application that he is interested in different news at work and at home. This can be simply achieved by adding corresponding context definitions in the profile.

Generally, applications built on the top of the framework will be specialized in performing one special task for the peer. For example one application could provide the peer with news it encounters in the nearby, while another application could provide location-based services for this peer. The P2PC framework supports an unlimited number of applications that can run at the same time on top of the framework. To build an application based on this framework only several interfaces must be implemented, the framework will inform the applications when proper users or objects are nearby.

One of the main goals of the framework is to run multiple profile oriented peer-to-peer applications on top of the framework without the need for detailed coordination knowledge. This is achieved through a layer architecture beginning with the hardware-nearest Transport/Sensor Layer responsible for transporting data and sensing nearby devices up to the Profile Layer that notifies applications of nearby devices with profiles that match the interests of the local user. Figure 2 shows the architecture of the peer-to-peer coordination framework.

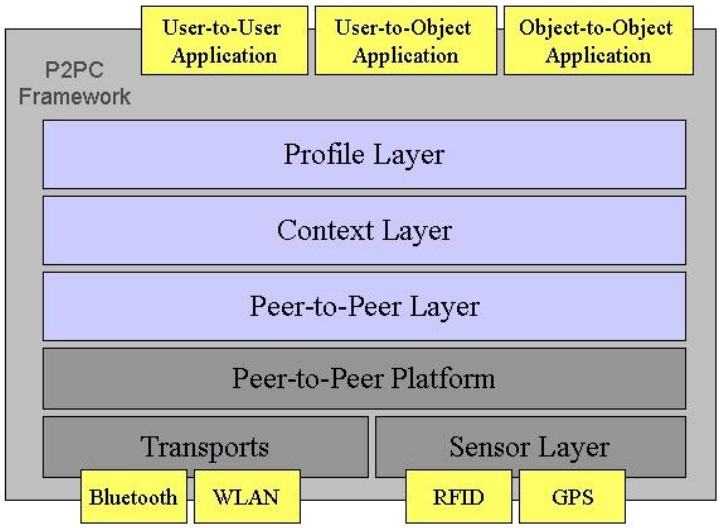

Figure 2: P2PC Framework Architecture

As mentioned above the framework uses JXTA to discover nearby peers and to communicate with them. JXTA itself is transport independent, it can send data over TCP/IP, over Bluetooth or any other communication technology. To discover peers, the framework uses JXTA advertisements, which are, if no sensors are used, periodically sent out. In order to communicate with other peers a JXTA pipe is established after a new peer has been discovered. By using the Transport/Sensor Layer special sensing technology can be used to restrict communication range between peers arbitrary. In general each technology will require its own sensor implementation, which can easily be plugged into the framework. After sensing a nearby device the sensor has the possibility to do some work required for the Peer-to-Peer Layer. When using Bluetooth as communication technology for example it is possible to establish a TCP/IP link over Bluetooth, so a JXTA implementation without Bluetooth - Binding can discover other peers. The sensor then notifies the Peer-to-Peer Layer of the presence of the new device (i.e. a new peer). The Peer-to-Peer Layer would then send out a JXTA advertisement. When receiving an advertisement the Peer-to-Peer Layer may ask the Transport/Sensor Layer, if the peer, which sent out the advertisement is actually in sensor range. The sensor would then response with yes or no and the Peer-to-Peer Layer would propagate a new peer or discard the advertisement. An example for such a "mixed-scenario" has been implemented and tested. Since JXTA does not yet support Bluetooth-Transport (we believe it will), we used Bluetooth as range sensor technology and WLAN (IEEE802.11) as TCP/IP transport for JXTA.

The Peer-to-Peer Layer manages the reachable list of peers and provides generic means for communication to other peers. When a new peer is discovered a JXTA pipe is established and the Profile Layer is notified of the presence of a new peer. After a specified amount of time or after a sensor notifies the Peer-to-Peer Layer that a device is out of range, the Peer-to-Peer Layer will again notify the Profile Layer of this. In addition to this management of reachable peers the Peer-to-Peer Layer allows layers on top to communicate with peers in range.

The Profile Layer fulfills the task of profile-oriented coordination. Profile Layer includes the PPDL (Pervasive Profile Description Language) module and a Profile Matcher module are integrated. The Profile Layer is notified by the Peer-to-Peer Layer when a new peer is in communication and sensor range. The Profile Layer builds a Role Exchange Profile (REP), containing the Roles that are provided by the different applications on top of the framework. The REP is sent to the opposite peer, which will also send its REP. If at least one role of the received REP match, the corresponding profile data is transferred. After receiving a profile it is compared to the local interest profile (the interest filter) using the profile matcher. When the profiles match (i.e. it is interesting enough), the Peer-to-Peer Layer informs all applications defining the corresponding role. Applications on top of the framework can then exchange application specific data between themselves.

Conclusion

This paper represents work in progress that is targeting to create a framework that enables context-aware applications with the ability of user-to-user, user-to-object and object-to-object ad-hoc communications. The concept of P2PC Framework is new and it is very promising for such areas as medicine, business, education and entertainment. Applications built on the top of the framework not only give context sensitive support to the user of the system but also are able to change their processes and adapt to the new situation, whenever it is necessary. This ability makes such applications dynamic in a sense of providing continues activity and flexible in a sense of using of immediate available knowledge and resources.

Apart of providing an implementation of the framework, we suggest a new approach for design of software systems that assume an ad-hoc nature of communication between participants of the network. This is extremely important in some scenarios like emergency scenario that has been described in this paper.

References

[BFS01] A. Buttermann, A. Franz, P. Sties, S. Vogel: "Ad hoc Networking – Technology and Trends", Center for Digital Technology and Management, 2001.

[BCM02] Wolfgang Beer, Volker Christian, Lars Mehrmann "How we are talking about Context Information?": Siemens AG-University of Linz, 2002

[CW02] Ekaterina Chtcherbina, Thomas Wieland In the Beginning was the Peer-to-Peer…, Siemens AG, 2002

[FAH02] Alois Ferscha, Markus Armbruckner, Manfred Hechinger

Peer-to-Peer Coordination Framework: How to write a SILICON Peer-to-Peer Application

[Gon01] L. Gong: "Project JXTA : A Technology Overview", Sun Microsystems, 2001.

[GS00] M. Gitsels, J. Sauter: "Profile-based Service Browsing – A Pattern for Intelligent Service Discovery in Large Networks". In *Proceedings OOPSLA 2000,* Workshop on the Jini pattern language, Minneapolis, October 2000.

[GSG01] M. Gitsels, J. Sauter, S. Goose, G. Schneider: „Enterprise on Air : Universal Mobility for Enterprise Users", UbiComp 2001.

[MGL00] S. Marti, T. Guili, K. Lai, M. Baker: "Mitigating Routing Misbehaviour in Mobile Ad Hoc Networks", Mobicom 2000, Boston, 2000.

[Ora01] A. Oram, Hrsg.: "Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology", O'Reilly, Sebastopol, 2001.

[PSG00] T. Pham, G. Schneider, S. Goose: "A Situated Computing Framework for Mobile and Ubiquitous Multimedia Access Using Small Screen and Composite Devices", in Proc. of the ACM International Conference on Multimedia, Los Angeles, USA, October, 2000

[Sch01] M. Schmitt: "Subscriptionless Mobile Networking – Anonymity and Privacy Aspects within Ad hoc Networking", University of Siegen, 2001.

[SCH95] W. N. SCHILIT, A System Architecture for Context Aware Mobile Computing, *PhD-thesis Columbia University 1995*.

[SO00] P. Suthar, J. Ozzie: "The Groove Platform Architecture", Groove Networks, 2000.

[So01] A. Sotira: "What is Gnutella?", Gnutella.com, 2001.

[Wei93] M. Weiser: "Hot topic: Ubiquitous computing", IEEE Computer, pages 71-72, October 1993