27. Peer-to-Peer and Ubiquitous Computing

Jussi Kangasharju (Darmstadt University of Technology)

27.1 Introduction to Ubiquitous Computing

Ubiquitous computing was introduced by Marc Weiser in the early 1990s [619]. In Weiser's vision, computers would become ubiquitous, that is, they would be present in every facet of human life. This vision has sometimes also been called the vision of the disappearing computer, since, as Weiser said, once computers become so commonplace that they are everywhere, they become such a natural part of the environment that we no longer notice them. Hence, it can be said that once computers are everywhere, they are, in fact, nowhere.

This vision is a logical consequence in the evolution of computer systems. In the early days of mainframe computers, several users shared a single computer's resources. In the current PC-era, one user typically uses a single computer. With the advent of new devices, such as PDAs and cell phones, we can already see that in the future a single user is likely to interact with several computers. However, today's world is still far from the vision of Marc Weiser, where computers are a natural part of the environment and interaction with them happens seamlessly.

The early days of ubiquitous computing were mainly focused along two axes. On the one hand, researchers were building new kinds of interaction devices, to explore how a person would live in a ubiquitous world. One of the most famous examples is the ParcTab [613] system which consists of palm-sized mobile computers that can communicate wirelessly through infrared transceivers to workstation-based applications. Since then, there have been many other projects around similar devices [602, 381, 598]. Such devices serve as prototypes for experimenting in building larger scale ubiquitous computing environments and serve as basis for determining the requirements such environments pose on the interaction devices.

Another important focus area has been location, or positioning systems. Many ubiquitous applications require that they be able to locate the user and adapt their behavior as a function of the movements of the user. An overview of different location systems can be found in [295]. Indoor location systems have been based on magnetism, ultrasound [12, 114], infrared [15] or radio waves (RF) [44, 467] as the transmission medium. Each of the systems has its strengths and weaknesses in terms of accuracy, range, and deployment cost, and none of them can yet be considered as "the best solution for all situations". It is likely, that future ubiquitous computing architectures will need to deal with several positioning systems concurrently.

R. Steinmetz and K. Wehrle (Eds.): P2P Systems and Applications, LNCS 3485, pp. 457-469, 2005.

Recent developments in ubiquitous computing are more concerned with defining and implementing architectures for formalizing and crystallizing the vision of ubiquitous computing. These architectures also have the role of tying together all the individual devices, services, and users. Some examples of such architectures are Sahara [499], Mundo [277] and Oxygen [465]. These architectures focus on the interactions between the different devices in the ubiquitous environment, with a particular emphasis on the communication needs and requirements of such environments. Indeed, these architectures highlight the need for efficient communication infrastructures, mechanisms, and protocols for building ubiquitous computing architectures.

In this chapter, we will focus on the communication needs of ubiquitous computing architectures. As we will see, ubiquitous computing architectures exhibit many properties similar to Peer-to-Peer systems. However, in contrast to the traditional, "PCs on the Internet"-like Peer-to-Peer systems, ubiquitous Peer-to-Peer systems have several additional challenges emerging from the nature of ubiquitous computing architectures. The goal of this chapter is to present an overview of the challenges and discuss current approaches for solving the communication problems in ubiquitous computing architectures.

The rest of this chapter is organized as follows. In Section 27.2 we discuss the characteristics of ubiquitous computing applications. In Section 27.3 we present the main features of communication architectures in ubiquitous applications. Section 27.4 looks into middleware in ubiquitous environments and discusses the requirements for such middleware. Section 27.5 compares the properties of Peer-to-Peer systems and ubiquitous computing applications and discusses the use of Peer-to-Peer middleware for ubiquitous computing. Finally, Section 27.6 presents the research challenges in Peer-to-Peer ubiquitous computing.

27.2 Characteristics of Ubiquitous Computing Applications

Ubiquitous computing applications have very different characteristics from traditional stand-alone computer applications, or Internet applications. In the ubiquitous computing world, we may have a very large number of devices involved in any interaction, typically in an ad hoc fashion. This is in stark contrast to the traditional computing world, where data is typically placed in certain well-known places and interactions are more structured. In this section, we will look at some of the main characteristics of ubiquitous computing applications, paying special attention to the characteristics which separate them from the "traditional" applications.

Table 27.1 presents a summary of the different characteristics we will discuss in this section.

Characteristics	Typical properties
Information	Small units, rapidly changing
Network	Wireless, ad hoc
Collaboration	Small devices, grouping, communities
Sharing	Limited devices, shared resources
Context	Behave "smartly", use context

Table 27.1: Summary of Characteristics of Ubiquitous Applications

27.2.1 Information

Information in ubiquitous applications plays quite a different role than in traditional applications. In the latter case, the user is often either a source or a consumer of information, and this governs the use and properties of the information transmitted in the applications.

In contrast, the information in ubiquitous applications is more often than not information sent from one small device to another; the information delivered to a human user plays a relatively small role in this world. Information in ubiquitous applications is also typically in small units and it may change rapidly. One good example which illustrates this are all manners of sensors which are a common building block of ubiquitous applications. For example, a motion sensor is constantly sending information about whether or not it detects movement. The actual amount of information is very small (i.e., is movement detected or not) and the state may change rapidly and asynchronously.

Ubiquitous applications must therefore be able to handle several flows of (possibly) rapidly changing information coming from a multitude of sources.

27.2.2 Network

Communication networks in ubiquitous computing applications also exhibit several characteristics which reflect these new applications. Often, the communications happen over ad hoc connections which are formed as the different ubiquitous devices come into communication range of each other. This implies that the network topology is highly dynamic and can change unpredictably.

Furthermore, since wired networks are typically not feasible for use in ubiquitous applications, owing to their distributed and dynamic nature, we must resort to wireless communication networks. Unfortunately, wireless network links are far more unreliable than wired links, and the data rates which can be achieved are considerably lower.

All of the above factors require us to rethink the communication abstractions in ubiquitous computing applications. We will revisit this issue in Section 27.3.

27.2.3 Collaboration

Because ubiquitous devices are typically small and have only very limited capabilities, they must collaborate in order to deliver useful services. This collaboration not only includes simple communications between devices, but extends to actual cooperation and active sharing of resources and information.

To achieve this collaboration, we need to resort to techniques such as grouping or communities which can be formed in an ad hoc manner. Such groups can be formed by, for example, the devices carried by a person, or all the devices in a given room. The devices within a group share a context and normally need to be aware of each other and each other's capabilities and needs.

27.2.4 Sharing Resources

Again, the limited resource of ubiquitous devices compel us to have them share their resources, in order to enable them to offer more sophisticated services. We consider the term "resources" in the broadest sense and include in it, not only computing power and storage capability, but also information contained in a device, as well as any additional devices attached to it.

27.2.5 Context Information

Since ubiquitous applications and devices can be used in many different circumstances, they need to be aware of the context in which they are being used. One good example of context information is the current location of a device, which may determine its behavior (e.g., when outside, a mobile phone might ring loud, whereas in a meeting it would set itself to vibration mode). Other context information are current environment conditions (light, temperature, etc.) or higher level information, such as the stored preferences of the current user [166].

27.3 Communications in Ubiquitous Computing Architectures

In this section, we will discuss the communication needs of ubiquitous computing architectures. In particular, we will present the publish/subscribe-communication paradigm as an attractive option for ubiquitous applications.

As we discussed in Section 27.2, the communication needs of ubiquitous computing applications differ greatly from those of traditional applications. Because of the highly autonomous nature of these new applications and their highly dynamic behavior, a centralized approach to managing connections (e.g., a fixed server handling all the devices in one room) is not possible. This leads us to investigate self-organizing communication paradigms.

Self-organizing communication models are very attractive for ubiquitous applications, since they allow us to handle the dynamics of the applications in a scalable and robust manner. One promising communication paradigm for ubiquitous computing is publish/subscribe communication [195, 103]. In the Publish/Subscribe interaction scheme, subscribers have the ability to express their interest in an event, or a pattern of events, and are subsequently notified of any such event, generated by a publisher, which matches their registered interest. The event is then asynchronously propagated to all subscribers that registered interest in it. The strength of this event-based interaction style lies in the full decoupling in time, space and synchronization between publishers and subscribers.

One major advantage of a publish/subscribe communication model is that it allows us to hide much of the communication handling from the application. Because ubiquitous environments may present many challenges for the communication layer (poor links, dropped connections, many errors, etc.) it is highly advantageous to be able to incorporate all communication code in a middleware layer. This frees the application programmer from having to worry about possible error conditions and it also allows the middleware to select the best possible communication means at any given time, without this having to be programmed into the application.

27.4 Ubiquitous Computing Middleware

The key glue which holds ubiquitous computing applications together is a middleware layer. This middleware provides the application with an interface for communicating with the "rest of the world", but it also protects the application from the environment (especially in terms of communication problems and issues).

However, because of the characteristics of ubiquitous applications, this middleware must take into account many different factors, and it must fulfill several requirements. In the rest of this section we will discuss these requirements in more detail.

27.4.1 Support for Heterogeneous Devices

The world of ubiquitous computing is highly heterogeneous; no longer is there a single dominant device type, such as a desktop computer. Instead, we have many different kinds of devices, small sensors, personal devices, traditional computers, large wall displays, etc. All these different devices have very different characteristics and these must be accounted for in a middleware. One example are the communication capabilities of the devices, i.e., what kinds of capabilities does the device possess and what does it "cost" to use them.

27.4.2 Resource Constraints

Many ubiquitous devices are very resource-constrained. Consider, for example, a simple temperature sensor which normally would include only the sensor, a simple processor with a minimal amount of memory, and means for communicating with other devices. Because of these constraints, a middleware must have a small memory footprint so that it can be embedded in as many devices as possible. Even though the cost of memory is decreasing, it will not be feasible to equip thousands and millions of ubiquitous devices with enough memory and processing power to run conventional middleware.

A consequence of the requirement of a small memory footprint is the modularity of the middleware. If we build the middleware around a small, minimal kernel, with additional services which can be plugged in and unplugged on-demand, we satisfy the requirements of a small memory footprint (since unnecessary services can be unloaded) and we have a flexible system which can adapt to rapidly changing conditions.

27.4.3 Mobility Support

Ubiquitous applications and devices are highly mobile, hence the middleware must be able to handle this. We can distinguish two kinds of mobility which require connections to be handed off during the communication.

On the one hand, we have horizontal handoffs (or handovers), which are currently commonly used in mobile phone networks. In a mobile phone network, a horizontal handoff occurs when the phone moves from the coverage of one base station to another. During this move, the base stations must transfer the communication resources to the new base station and the mobile phone must then switch to this base station. In the ubiquitous computing world, we do not necessarily have base stations, but can define a horizontal handoff in an analogous manner. A horizontal handoff occurs when the device must change its communication partners, but it continues to use the same technology (e.g., BlueTooth or WLAN) for the communication.

On the other hand, we have vertical handoffs which occur when the device must change communication technology in order to maintain the connection. For example, a device in a WLAN hotspot which moves out of the reach of the access point must switch to, e.g., UMTS, to remain connected.

Although horizontal handoffs are currently much more common, vertical handoffs are likely to become more common with the proliferation of WLAN networks. Some horizontal handoffs (especially in mobile phone networks) can already be handled with current technology, but other types of handoffs, especially vertical handoffs, usually result in broken connections.

Ubiquitous middleware must therefore have efficient support for both kinds of handoffs, across a wide range of different networking technologies.

27.4.4 Networking Support

Networks and communication links between devices in ubiquitous computing form spontaneously and in an ad hoc manner, without any necessary planning in advance. This type of spontaneous networking implies that nearby devices need to be found as they come in range, and communication links must be formed on demand. This sets requirements for the middleware, implying that it must handle these situations efficiently.

Another particular characteristic of ubiquitous networks is that devices may disconnect and be unavailable for long periods of time. In other words, "disconnected" is a normal state, not an exception as in traditional fixed networks. The middleware must support long disconnects without having the applications have to deal with them. In practice, this may seem to the application as if the network was performing extremely slowly, but when the network connection is resumed, the application should be able to continue as if nothing had happened.

Ubiquitous networks are typically based on radio communications (e.g., BlueTooth, GSM, UMTS, WLAN, etc.). Radio links are, by their nature, broadcast media and are therefore also well suited for (local) multicast communications. Unfortunately, BlueTooth for example, goes to great lengths to make it impossible for the application to use multicast, since BlueTooth makes all links point-to-point. Such behavior should be discouraged in a ubiquitous middleware, since the multicast (and broadcast) capabilities of the underlying radio medium come "for free".

27.4.5 Performance Issues

Ubiquitous computing environments present many challenges for building high-performance systems. Many of these challenges stem from the nature of ubiquitous devices, which was discussed in Section 27.2. Because of the

different nature of the applications, we must emphasize different factors when looking at the performance of ubiquitous middleware and applications.

Naturally, the more "traditional" performance metrics, such as processor speed, network bandwidth, etc., are also important in ubiquitous computing, but in addition to those, there are several other factors to consider.

For example, ubiquitous applications typically run over wireless networks. Although the bandwidth of wireless networks is increasing, it is doing so at a much slower rate than the bandwidth of wired networks or CPU speeds. This implies that wireless bandwidth should be treated, to some degree, as a "rare commodity", and middleware and protocols should be designed to take this explicitly into account. In other words, applications should not assume that data can be transferred easily from one node to another on demand; instead, applications should be prepared to work with slow connections and even the occasional disconnect.

Furthermore, ubiquitous devices are often very constrained in terms of processing power and memory. This puts a limit on the amount of processing that can be done locally, and further implies that any heavy computation might best be done on a networked server (keeping in mind the constraints of the slower network, as mentioned above). The same applies for the on-board storage of the device. Although CPU power and memory capacity are increasing at a rapid pace, it may not be feasible to equip every single device with a fast CPU and a large amount of memory for reasons of cost and power-efficiency.

Power-efficiency is the third major difference between traditional and ubiquitous applications. Many ubiquitous devices run on batteries and every action they perform consumes the on-board batteries. Like wireless network bandwidth, battery capacities are also growing very slowly (compared to other components) and are likely to be the primary limitation for ubiquitous devices and applications in the near future. Battery life is possibly the most crucial of the performance aspects, since when the battery runs out, the device can no longer function at all.

A high-performing ubiquitous application is therefore not necessarily one which performs its tasks in the shortest time, but one which can take into account the above three factors and be the most efficient in those terms. This may include designing protocols which, for example, avoid periodic maintenance messages (which consume battery power, since they require using the network device and can jam networks), and instead are able to tolerate network partitions and occasional inconsistencies (and thus prolong battery life).

27.5 Peer-to-Peer and Ubiquitous Computing

Peer-to-peer networks and systems are based on the principles of self-organization, resource sharing, and independent devices collaborating to form a larger system. These basic characteristics match well with the typical characteristics of ubiquitous applications, where the conditions are similar. Therefore, it is natural to consider Peer-to-Peer technologies as a building block of ubiquitous computing architectures and applications.

Peer-to-peer networks are typically based on (highly) autonomous peers collaborating to provide the different services. The peers provide the resources for the network and for other peers to use, but individual peers remain independent in their actions. Individual peers are free to go offline and come online as they please. These properties are the cornerstone of Peer-to-Peer organization principle and are one of the main strengths of Peer-to-Peer networks.

In fact, if we look at Peer-to-Peer systems and networks from this organizational point of view, we can abstract their inherent capabilities and properties and apply these same principles and properties to other kinds of systems. By drawing the parallels between Peer-to-Peer organization and the organization of entities in other systems (ubiquitous computing architectures in our case), we can observe the similarities and exploit the power of Peer-to-Peer systems in other fields.

Peer-to-peer principle is an attractive choice for organizing ubiquitous computing systems for several reasons. As we look at ubiquitous computing architectures, we can immediately see many common points with the Peer-to-Peer organization principle. In ubiquitous computing architectures, there are many completely autonomous devices (e.g., sensors, user devices, etc.) scattered in the environment. Each of these devices has some functionality which is important to other devices, i.e., each device provides some resources for all other devices, services, and applications. A device may rely on the information provided by another device in order to complete its function. Furthermore, the mobility of the devices and the limited range of communication implies that, from the point of view of a single device, other devices appear to have highly intermittent connectivity. In some cases, devices may also disconnect in order to conserve battery life.

From the above comparison, we can see a possible mapping of the Peer-to-Peer principle to ubiquitous computing architectures. At first sight, this mapping appears quite straight-forward, due to the large similarities between the two systems.

The main area of application for the Peer-to-Peer organization principle in ubiquitous computing lies in building communication architectures for ubiquitous computing. As mentioned earlier, the key focus in ubiquitous computing lies in efficient communication architectures which allow the different devices to offer their services to other entities in the environment. Such communication architectures exhibit many Peer-to-Peer-like properties and can

benefit from the same principles as traditional Peer-to-Peer systems in the Internet. However, ubiquitous computing has a host of special requirements, which stem from the nature of the applications and devices, as we have presented above. This implies, that the traditional Peer-to-Peer solutions may not be applicable in ubiquitous computing.

Before Peer-to-Peer solutions can be applied to ubiquitous computing, we need to evaluate their suitability. As our discussion above shows, there are many similarities between Peer-to-Peer systems and ubiquitous computing architectures which suggest the possibility to leverage Peer-to-Peer technologies in ubiquitous computing. However, many of the constraints and special requirements of ubiquitous computing make traditional Peer-to-Peer solutions unsuitable for ubiquitous computing. Peer-to-peer systems are based on overlay networks, which require significant effort just to maintain the overlay structure. For ubiquitous devices, such effort implies large battery consumption, just to be able to send messages to other devices. Traditional Peer-to-Peer solutions implicitly assume that the peers are PCs connected to the Internet with high-bandwidth links (where even a 56 kbps modem link would be called high-bandwidth for a ubiquitous sensor with only a few bits per second infrared communication means!). Traditional Peer-to-Peer solutions need to be adapted to handle the high dynamics and unpredictability of ubiquitous computing.

Abstracting the communication components into a middleware layer is especially important for ubiquitous computing. Even in the traditional Internet, code for communication has to handle several different kinds of failures (DNS failures, connection failures, etc.). In ubiquitous computing, the number of possible failures is much higher (see Section 27.4) and we cannot assume application programmers to be able to handle all possible errors. Indeed, this would be the same as assuming that the application programmer would be aware of all possible situations in which her application would be used. Clearly this is not feasible. A middleware layer is responsible for hiding the communication problems from the application and provides support for the application programmer.

Peer-to-peer middleware for ubiquitous computing is still a topic of ongoing research. The requirements for such a middleware are known, as outlined in Section 27.4, but there are still many open challenges. As a conclusion, we can say that Peer-to-Peer technologies and principles are a promising building block for ubiquitous computing middleware.

27.6 Research Challenges in Ubiquitous Peer-to-Peer Computing

In this section, we will outline the major research challenges in the area of ubiquitous Peer-to-Peer infrastructures. Each of the topics mentioned below

has several interesting research problems, which need to be solved before ubiquitous infrastructures can become a commodity. Some of these problems have been studied in the more traditional computing world, but whether existing solutions are applicable to ubiquitous computing (due to the special requirements) remains an open question.

27.6.1 Heterogeneous Devices

The heterogeneity of devices and networks in the ubiquitous computing world is a big challenge. Applications and communication protocols must adapt themselves to a variety of platforms and conditions. This includes running on low-power devices with slow connections, as well as high-power servers with gigabit networks. In addition, networks in the ubiquitous computing world are typically wireless and therefore highly unpredictable, both in terms of available bandwidth, as well as the availability of a network connection in the first place.

27.6.2 Efficient Algorithms

Because of the constrained nature of ubiquitous devices, we need efficient algorithms for the organization of, and communication in, these networks.

On the one hand, the algorithms must be efficient in terms of the algorithmic performance; for example, a search algorithm must find the desired object or service in a short time. This aspect of efficiency can in some cases be directly adopted from the algorithms in the traditional systems, but may need to be adapted to the specifics of the ubiquitous application.

On the other hand, considering the constrained resources of ubiquitous devices, the algorithms must be efficient in how they use the available resources. This includes taking into account things such as signaling overhead, and possibly leveraging compression technologies for cases where we can trade off CPU and battery power for better bandwidth utilization.

27.6.3 Security and Privacy

In a world where a multitude of devices are scattered around and are observing their environment, security and privacy issues are of paramount importance. Security allows us to authenticate the devices with which we communicate (and vice versa!) and is an important building block for establishing trust. The world of ubiquitous computing is moving in a direction where more and more of our everyday activities are taking place in the digital world and

therefore it is vital that we (as human users) are able to trust the devices and architectures handling our affairs.

Likewise, in a world with many devices observing us, privacy issues have an important role to play. These issues include the technical problems of handling and preserving user privacy, through the use of technologies such as anonymous (or pseudonymous) communications and transactions. Another aspect of privacy concerns the non-technical issues, such as user acceptance, and more generally, the expectations of the society as a whole with regard to what level of privacy can be expected in widespread adoption of ubiquitous computing architectures.

27.6.4 Scalable Architectures

Due to the large number of devices in ubiquitous infrastructures (millions, even billions in a global infrastructure), the underlying architecture must be very scalable to handle them. This ties in with the efficient algorithms, but also goes a step beyond, by extending the notion of efficiency to the global level. The architectures must be designed in such a way as to allow the efficient algorithms to work and enable a seamless interaction between any components that need to communicate.

27.6.5 Next Generation Peer-to-Peer Middleware

Probably the most important challenge in the research on ubiquitous infrastructures is the development of a next generation middleware for ubiquitous applications. As we discussed in Section 27.4, the requirements of such a Peer-to-Peer middleware are already laid down, but a lot of work is still needed to turn those requirements into an efficient and scalable infrastructure.

Peer-to-peer technologies are an attractive building block, since they, by their nature, support autonomous self-organization, intermittent ad hoc networks, and exploitation of resources of the edge devices.

27.7 Summary

In this chapter we have discussed ubiquitous computing architectures and how Peer-to-Peer technologies can be applied to them. We have outlined typical characteristics of ubiquitous applications and analyzed their needs, especially in terms of communication. Efficient communication architectures are a key component in building middleware ubiquitous computing architectures. Middleware is an important building block for ubiquitous computing architectures and asynchronous communication means, such as publish/subscribe

provide an interesting basis for communication architectures in ubiquitous computing.

The Peer-to-Peer organization principle can also apply to ubiquitous computing. In the Peer-to-Peer principle, peers act autonomously, but collaborate with other peers to provide services. In a similar vein, ubiquitous computing devices act autonomously, but depend on each other in their actions. Peer-to-peer middleware is an interesting possibility for solving the communication problems in ubiquitous computing and providing an asynchronous communication abstraction to the applications. We finished the chapter by providing an overview of open research problems in the area of ubiquitous Peer-to-Peer infrastructures.