Mobile computing

Ludìk Matyska and Eva Hladká

Faculty of Informatics, Masaryk university Brno

1 Introduction

Contemporary working environment is characterized by its mobility. Increased number of people does not have a single work place, they are often using a variety of even remote equipment while requiring seemingly permanent contact among themselves and with corporate headquarters. This trend can be easily seen also at universities, where information, computational and other services are to be available for people moving literally around the globe. Increased number of people is being equipped with mobile devices, ranging from small personal ”wearable” devices like mobile phones and PDAs (personal digital assistants like Palm or recent iPAQ from Compaq) to more powerful systems like notebooks. All these devices can be in some way connected to other and increasing number of them provide ways to be connected directly to Internet, using some wireless connection system (infrared, GSM or similar cellular telephone systems, wireless network protocols like IEEE 802.11b etc.). When using such devices, people expect to be given the same range of applications regardless of actual place of use-they expect support for applications mobility.

By mobile computing in the broad sense we will understand any computing activity when either users, computing elements or the computational processes are not bound to one physical place and change it while performing same or similar sets of actions. We particularly add the mobility of the computational process to the common definition of mobile computing, as this completes the symmetry of the picture of mobile computing landscape and also allows us to find parallels between mobile computing infrastructure requirements and the GRID systems build to support ubiquitous computing environment. Many interest is currently given to ”always on” systems, which are always connected to larger network. A typical example of such system is a mobile phone, which can be used virtually at any place and at any time for a duplex communication (provided the area where the user is moving is covered by the appropriate signal) and is even able to support such a communication while the user (and device) is moving. Another example may be a PDA with a communication card or even a portable computer with a cell-phone card. Also, all ”wearable computer systems” [1] belong to this category. However, the utility of such systems is restricted by their size (they have to be truly ”wearable”) and by the capacity of the wireless link and they are in no way a replacement of desktop or large systems.

On the other hand, the computing, memory and permanent storage capacity together with the capabilities of graphical subsystem of contemporary notebooks leads to replacing desktops with such inherently mobility supporting devices. Users are expecting to have the same (or, in fact, even larger) set of applications running on notebooks. They are also expecting that these applications behave the same way regardless whether they are connected to high speed (usually 100 Mb/s Ethernet) network at office, to much less powerful network (using analog, digital or cable modems or more advanced systems like DSL or ADSL, with capacity from tens of kilobits per second to megabits per second), to narrow band wireless network (using cellular phones) or even when temporarily disconnected from the network. The development of applications and infrastructure supporting such a behavior is very challenging and far from finished task [2].

There is third kind of mobility, which lies in the mobility of computing processes. People are using shared remote computing systems and they initiate large scale computations which should not be completed at the same place they were started. Also, as people are collaborating, the computation initiated by one person may be taken over by a different one, from different location, or they may be even ”looking over” such a computation together [3]. While currently this is mostly area of large scale scientific computing experiments, we can envision a not so distant future where this will be used by much broader commercial and entertainment community. Even now, this kind of problems is investigated within the area of large web servers. These are usually realized as farms of computers where processing of end user requests is moved from heavily to lightly loaded ones without any notice to the user.

Small step further from the computational mobility leads us to the GRID systems. As GRID we will understand (physically distant) computers connected via high-speed network and behaving as one large scale ubiquitous computer [4]. The primary goal of GRID research is creation of an illusion of a computer with (almost) unlimited power available from any place-but this is in fact the same behavior (although differently stated) as we expect from mobile computing support. With only mobility in mind we focus to access and connectivity, while with GRIDs we are interested in broader set of features of the whole system [5].

In this paper we will try to look at mobility from a slightly different point of view-we will try to hide the effect of mobility and discuss basic infrastructure and its features necessary to create an illusion that user or computation did not move. We will also show in more detail how this relates to the GRID technology. Much more ambitious approach to the pervasive computing can be found in [6].

2 Infrastructure for mobility

The restricted mobility we are interested in covers situations where end user moves from one location to another with the following expectations:

1. She will be able to work the same way, with the same applications, on all places.
2. She will be able to work while moving; however, shoe does not expects synchronous communication (like event notification) to occur ”on the move”.
3. She will see no substantial difference on her work with respect to the quality of connection (i.e., she expects the same application behavior regardless of the actual network throughput).

The first and second expectation is trivially satisfied if all applications are local on the portable device and do not use any external data. Also the third expectation can be trivially satisfied if the ”usual” behavior is based on the lowest available (e.g., 9.6 kb/s) transmission rate. We are, however, not interested in such trivial solution as it severely restricts potential utility of mobile devices-the collaboration between more people or between one person and more computing systems is limited (but could be very important, as mobile phones are perfect examples of such systems).

Let our ”base” system be a notebook connected to high speed network at users’ office where applications and especially data are shared among different users and systems (either as a result of licensing policy or due to the simplicity of support). The environment we are moving in is ”ordinary” Internet, i.e., we will not deal with virtual private networks or other closed systems. On the contrary, we are interested in mobility to areas where user can expect only publicly available services (no private pre-build systems should be considered).

2.1 Connection

The first problem encountered when moving to a new location lies in establishing a connection to the Internet and through it to the ”home” servers and systems. The situation is far more complicated than just finding an Ethernet or phone outlet and use it to connect the computer to Internet. While increased number of hotels offers direct Internet connectivity, and there are companies that provide global Internet connectivity, the problem lies in the change of mobile system identity. Each device connected to the Internet has it unique identifier, the IP number. However, the currently used Internet Protocol (IPv4) does not have provision for a service known from advanced telephone systems, the ”number portability”, which means the actual number is a function of place and often also of time (in systems like DHCP with dynamically assigned addresses). Unfortunately, many simple authentication systems still use the IP number as identification of the host accessing a particular service. Large database providers (e.g., in university environment, many bibliography providing databases) usually license the use to a predefined set of IP numbers. This may not pose a problem within the organization (even with the local use of DHCP protocol, it is sufficient to specify all the dynamically assigned IP numbers), but it leaves out all users outside the licensee premises.

Firewalls, widely used for security reasons, provide similar threat to the mobile users. Firewalls are set to distinguish between ”insiders” and ”outsiders” (those behind and after the firewall) and they do not grant ”outsiders” the same access rights to internal services.

Another example are e-mail systems (based on SMTP), which are increasingly set up in such a way as to accept mail for outside delivery from only pre-defined set of machines, specified again by their IP numbers (otherwise they can be easily used to deliver spam messages). A mobile user can thus be easily disconnected from her mailing system and may not be able to send e-mail completely or at least not under her usual identity (different, usually local IP connectivity provider related identity will appear in the ”From:” field). While the consequences of the former are obvious, even the later may prohibit a use of e-mail, as many electronic e-mail based conferences identify authorized users by the ”From:” e-mail field.

A simple solution to both problems lies in authentication which does not rely on IP numbers nor simple notion of ”outsiders” and ”insiders”. A virtual secure channel must be established between the mobile device and the service providing system, based on some authentication protocol, and access to services should be granted to only authenticated systems or persons. In fact, a tunnel between mobile device and service provider (this could be a firewall granting access) passing through the whole Internet will serve this purpose. The SSL (Secure Socket Layer) is the most prominent example, allowing a secure connection between web browser and web server regardless of their respective placement (the authentication uses a shared secret, i.e., password, to authenticate the user). In the UNIX world, ssh (secure shell) serves the same purpose.

More advanced systems, based, e.g., on the IPv6, does have support for IP number portability. In these systems, instead of authenticating to the services, the user authenticates herself to the network and, when the authentication is successful, is given the same identifier which can be directly used by higher levels for user authentication. While very perspective, such systems are not yet widely available on the Internet.

2.2 Application support

Providing local copies of applications is not enough to fulfill the ”same application” expectation. While many applications are compact and self-contained (e.g., text processors, spreadsheets, . . .), increased number of applications has client and server sides. And even the compact applications usually work with files or other potentially shared data obejcts. While the client side can be easily copied, this can not be done with the server side without additional support. Let as consider the following client-server applications:

1. Database system.
2. Web server and browser.
3. Distributed filesystem.

All these three examples have several important features in common:

1. The mobile device usually does not have enough capacity to store server side of the whole application and associated data. There also could not be enough computing power to support such complex applications.
2. There may be connections with other systems, e.g., there are more than just one web server and the relevant information may be dispersed among them. Also, the database may not be compact, and the same could apply to any advanced distributed filesystems.
3. The security implications are too serious. Even if the capacity allows, copy of corporate data on mobile device poses a serious security risk. This risk can be completely removed only be forbidding the remote access, but a usual trade-off between utility and security requires only necessary data to be copied (made available), as this approach provides more control over the risk if the mobile device is lost or stolen.
4. The synchronization problem. Whenever a change in any stored data is made, user expects to have such change propagated and available to other users of the system.
5. Some server functionality must be presented on the mobile system if we have to support work in disconnected state.

One possible solution of this problem may be based on creating applications that are mobility aware, i.e., which detect the connection state and adapt to its changes [2]. This solution has its own drawbacks, most serious being the necessity to create specialized applications-the solution highly discouraged by software engineering recommendations. Better approach is based on separation, where only some subsystems are mobility aware and the rest simply relies on correct behavior (and extended semantics) of such subsystems.

We can use a cache model to support such functionality. This is relatively easy for stateless servers used in, e.g., NFS distributed file system or http protocol, where caches are already widely used to enhance their functionality. However, the mobility aware cache has a different semantics. The ”ordinary” cache works as a proxy, i.e., it performs some operations and/or actions on behalf of the user (towards the server) or on behalf of server (towards the user). If the cache cannot fulfill the requirement, it simply sends it to the other side that should be able to perform the desired action. No support for disconnected state is provided, nor the caches adapt (beyond the obvious adaptation of the underlying transport protocol like TCP/IP) to changes in network throughput.

On the other hand, the mobility aware cache must be able to work satisfyingly in the disconnected state and must be able to adapt to the changes in connection quality. The expected behavior should be supported by the following functionality:

1. Pre-loading data which could be used when the system is in disconnected state. This can be either automatic (based on user behavior and habits observed in the past) or manual (user must specify which data she probably will need).
2. Journalling (logging) all changes made to the data in cache when in disconnected state. The local cache state should change as if these changes were propagated to the server. System could be optimized not to perform redundant or useless operations when reconnected (e.g., user creates a temporary file which is later on deleted-all operations on such file, including its creation and deletion, are in fact null operations and should not be repeated on the server). The journalling protocol should be network savvy, i.e., it should transfer as few data as possible. This is already default operations in database journalling systems, but should be extended to other client-server system (e.g., in the filesystem situation, after editing a large file transmission of actual operations performed and repeating (”replaying”) them on the file stored on the server is usually much more efficient than transmission of the whole file).
3. Prioritizing the logged operations when reconnected. Again, let us assume the filesystem cache-operations on user own (non-shared) files should have the lowest priority for synchronization, while any changes on shared files should be propagated as soon as possible. The prioritizing scheme should include both directions of the synchronization and propagate the most important changes first. This will usually require some manual support and also an estimation of time the synchronization would take (based on measured or reported (see GRID information service below) available network bandwidth).

2.3 Connectivity illusion

Mobility aware caches inserted between clients and servers create an illusion of permanent connectivity. In a properly configured system, there is no way how the user can distinguish on-line work from the off-line one for all applications that do not require synchronous (real-time) communication. In fact, the cache system can increase the perceived efficiency of application, as all operations are performed locally (on a mobile device) and the response time does not rely on the state (connected or disconnected) nor its quality (e.g., low or high speed network connection).

Electronic mail

E-mail is a typical representative of asynchronous communication system, which has also many features of database and document delivery systems. The basic scenario is as follows.

All incoming mail is stored in one mailbox on a mail server. The same mail server is used to store all personal mailboxes, which are created and updates as a consequence of user initiated actions. User accesses mailboxes on the mail server either directly or via POP3 or IMAP protocols. For the former, she simply logs into the mail server (using telnet or ssh protocol) and uses some local mailing program (like elm or pine) to read, write, and delete individual mails and to manipulate whole mailboxes. Sometimes, she uses IMAP protocol to access incoming mail and to work on it locally. The main problem with IMAP (and to even more extent with the POP3) protocol is its inability to support both local and server based mailboxes [7]. Using IMAP, user can download a copy of any mailbox to her mobile device, to read and write messages and to send them even in disconnected state (the outgoing mail is simply stored by what we can consider a simple mail-cache and forwarded to mail server when reconnected). However, any changes on a mailbox are not directly propagated to the server, no synchronization between the locally changed mailboxes and the server is provided. It is naturally possible to copy individual mailboxes back to the server, but this approach has following drawbacks:

1. It assumes that user accessed the mail server via the mobile device only. If user (or some automatic script on the server) modified any mailbox, the changes are lost.
2. The incoming mailbox cannot be copied back as there could be new mail that would become overwritten.
3. The changes on individual mailboxes are usually very small compared to their total size. Transmission of whole mailboxes is therefore very inefficient and time (and money) consuming.

To support the *mobile e-mail system* the IMAP (or similar) protocol should be enhanced with the cache-like behavior discussed above. At the beginning of use, the cache will store all (relevant) personal mailboxes including the incoming one. Instead of making a local explicit copy, all mailboxes will be simply cached. The content of mailboxes can be changed only via a predefined API which understands their semantics and which is responsible for logging all actions taken. When the system is on-line, all changes are directly propagated to the server and replayed there to achieve real-time synchronization. In disconnected state, the changes are performed locally and the log of all changes is stored. After the reconnection, the log file is transferred (usually in chunks whose size is a function of actual network throughput) to server and there replayed to synchronize the server with the local state. Simple priority scheme will flush out the outgoing mail, then load new incoming mail (or vice versa) and only afterwards will synchronize individual personal mailboxes. As only log file is actually transferred, the amount of transmitted data is minimized. The protocol can be easily enhanced to support compression between cache and server, reducing further the amount of data transmitted. Having the mailboxes in cache only secures that no unnoticed changes are made (e.g., editing the appropriate file with a file editor) which would not be propagated to server.

Connection between cache and server could be authenticated by any method (symmetric, like Kerberos, or asymmetric, like PKI, cryptography protocols can be used and they can even be replaced without any visible change to the application behavior), the connection itself can be tunneled via some secure channel and can even pass through firewall (the authentication protocol could have several phases, the first one being responsible for authentication to a firewall).

Filesystems

Filesystems provide access to files, data objects that are manipulated by the majority of application programs. Network and distributed filesystems provide access to files stored on remote server(s), using cache as a tool to increase user perceived efficiency of the filesystem. However, these caches do not support work in disconnected state nor they directly support change of IP identity (this is especially true for the common NFS protocol, which uses IP based authentication). The filesystem caches should be enhanced to support mobility by allowing IP-independent authentication and support for work in disconnected state.

With the increased interest in journalling filesystems (like XFS from SGI or ReiserFS in Linux) it will be rather straightforward to use the journalling facility in filesystem caches, too. Any operation on a file is logged and replayed on the server whenever the server becomes available. The log entry is cleared when server modification is confirmed (committed). The log file can be continuously optimized and redundant (or null) operations can be removed (we already presented an example of creation and subsequent deletion of a temporary file, but this also includes opening a file in read-only mode, opening in read/write mode without actually modifying the content of the file etc.). Entries in the cache log file can be also prioritized and when the mobile device is reconnected, the filesystem changes can be replayed in different order then they actually occurred (naturally, the overall semantics must not be changed). It is also possible to enhance the server log behavior in such a sense that all the files stored locally are known to the server and any changes on the server (on any of these files) are kept in a permanent log (a kind of a copy of internal log) and this log is replayed on the mobile device. This way the amount of data transmitted between server and client is minimized.

The authentication of a local filesystem cache should fulfill the same conditions as discussed above for the e-mail cache, in particular it should not depend on actual IP of the mobile device. Again, tunnels and staged authentication should be used to provide access through firewalls.

2.4 Discussion

Infrastructure support for ”always-on” mobility is complex and requires deep changes both in protocols and applications and also in hardware used to support them. On the other hand, support for the ”after move” mobility can be based on an illusion of permanent access, and this illusion can be created using system of mobility aware caches. These caches should provide an API that mimics the API of the remote server. All operations should be logged and the persistent log replayed on the server after the connection is established. The log entries can be optimized (redundant and null operations could be removed) and they also can be prioritized, i.e., the order they are executed on server is different from the order they were issued locally. Further optimization can be achieved sending compressed data between cache and the server. The cache can also take care of all the necessary security, from authentication to the server (or to the firewall(s)) to providing a secure communication layer.

Access to the cached data objects is allowed through the cache API only, which means no modification can go unnoticed. As all the mobility awareness is delegated to the cache, unmodified local (client parts of) applications can be easily used without any changes. The server side may not be modified, either, as the cache logs the user issued commands and can thus behave in exactly the same way as the user. Naturally, modification of servers and making them aware of the existence of the cache system may further enhance the efficiency (as was discussed within the filesystem support framework).

More advanced systems will require some interaction between caches to provide global priority scheme and to avoid multiple authentication requests.

3 GRIDs and mobility

GRIDs are built with the purpose to support collaborative work and to enhance sharing and combination of resources available on Internet. Providing power to tackle the most challenging data storage and computing requests, GRIDs can be also seen as a computational infrastructure of tomorrow. Like nobody is building its own electric power plant and uses the outlets available virtually in any place, GRIDs will provide us with similar access to almost unlimited (and uninterrupted) computational and data storage power. In order to fulfill this expectation, research in GRID development shares many problems with the support for mobility:

1. Access to GRID should be ubiquitous, from any place. While within GRIDs this requirement reflects the support for collaborative work (you do not know in advance where will be people you would like to work with), the resulting infrastructure is perfectly suitable to support users’ mobility as well.
2. The computational processes on GRID could migrate (either as a consequence of local failure, expiration of local authorization or due to limited resources available on a particular place) and user is expected to keep track of all such moves. Moreover, it is expected that a continuous flow of data (e.g., for a real-time visualization) could go to one or even more users regardless of the actual placement of processes creating the flow. This is a symmetric situation to the end user mobility support, where the

computation (e.g., servers) are immobile but users move, and it is not surprising that the underlying support should be very similar.

3. GRID users are expected to move and to have access to ongoing computation after (and more often even during) the relocation. This is especially important when large scale computing is performed (which could last several days or even weeks) and users must observe the progress and to modify parameters influencing the computation-the so called computational steering. Here, the GRID and mobility requirements are the same.
4. GRID systems are supposed to provide information about available resources (capacity of computational and storage servers, network throughput etc.) and the state of the GRID itself. Mobile users can use the same information to optimize use of network, access to some resources (like availability of a neighborhood service server) and also for resource discovery (e.g., access to local printers).

GRID computing goes beyond simple mobility support. It is focused on large-scale resource sharing and new distributed applications. The ”GRID problem” is defined as ”flexible, secure, and coordinated resource sharing among dynamic collections of individuals, institutions, and resources” [5]. However, as shown above, the challenges encountered when providing such environment, especially authentication and authorization, resource access and discovery are in principle equivalent (even if on larger scale) to those known from the mobility support requirements.

GRID aware authentication protocol (e.g., GAA) can be easily used by mobile systems to authenticate to their home services. Tunneling through firewalls, inter-domain security issues are also common both for GRID and mobile environment.

GRID file transfer protocols (e.g., GRIDftp) can be used to transfer data between the immobile server and the mobile device. The GRID information service (e.g., MDS) can be used to find the closest resource (e.g., printer) or can provide information necessary to estimate file transfer times.

Virtual organizations introduced within the GRID community [5] can be understood as models of institutions together with their mobile users-they spans or shrinks in time as mobile users move.

4 Conclusion

Powerful mobility support can be provided combining IP number independent authentication and authorization and the mobility aware cache system. This model uses illusion of a permanent connection to allow applications work seemingly the same way (from the user point of view) regardless they are on-line or off-line. The model requires powerful mobile devices like notebooks with sufficient storage and computing power to create and maintain such an illusion. It is well suited for applications and systems which are or can be understood as asynchronous, like e-mail messaging systems, access to databases (at least when there is no real-time requirement) or filesystems (again, when delayed file sharing is acceptable). The further advantage of this model is its support of unmodified applications-the caches can hide all the mobility dependency from both the client and server side of an application.

The model is not suited for applications requiring synchronous communication, where the real-time behavior is imposed. It is, however, expected that increasing number of even such applications will include also asynchronous alternatives. Telephone is a premium example of real-time communication, but with the invent of memo-boxes the asynchronous communication is also supported. Another example is scientific visualization, where we can see a move from direct interactive graphical interactions (they are extremely demanding on network bandwidth and graphical subsystem capabilities) to video streaming, which is much less interactive but can be viewed from almost anywhere.

GRID computing, while having different purpose and heading from different direction, shares many infrastructure related challenges with the mobility support. Combination of GRID and mobility research, especially at the middle-ware layer, may lead quickly to new and more usable systems.

Acknowledgment

This work was supported by the research intent of CESNET, z.s.p.o. under the code MSM 000000001.

References

[1] A. Smailagteosic, D. P. Siewiorek: Modalities of interaction with CMU wearable computers. IEEE Personal Communications, 3(1), pp. 14-25, 1996.

[2] G. S. Welling: Designing Adaptive Environment-Aware Applications for Mobile Computing. PhD dissertation, Rutgers, SUNJ, 2000.

[3] G. Allen et al: Early Experiences with the Egrid Testbed. To appear in Proc. Cluster Computing and Grids, Australia, 2001.

[4] I. Foster, C. Kesselman: The GRID: Blueprint of a new computing infrastructure. Morgan Kaufmann Publishers, San Francisco, 1999.

[5] I. Foster, C. Kesselman, S. Tuecke: The Anatomy of the GRID. To appear in Intl. J. Supercomputer Applications, 2001 (http://www.globus.org/research/papers/anatomy.pdf).

[6] G. Banavar et al: Challenges: an application model for pervasive computing. Proc. Int. Conf. Mobile Computing and Networking, ACM/IEEE, pp. 266-274, 2000.

[7] http://www.imap.org