# Supporting User Mobility with Peer-to-Peer-based Application Mobility in Heterogeneous Networks

Dan Johansson, Karl Andersson, and Christer Åhlund
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
SE-931 87, Skellefteå, Sweden
Email: {dan.johansson, karl.andersson, christer.ahlund}@ltu.se

***Abstract*—When migrating applications between devices during runtime, one has achieved application mobility. In this paper we present the XAM system, providing application mobility through a peer-to-peer based solution over heterogeneous networks. Our architecture takes into consideration established requirements of application mobility, being application identification and distribution; context-awareness and context quality; seamlessness; heterogeneity support; and usability. Our system differs from traditional systems as it provides application mobility in a decentralized manner over heterogeneous networks, using different communication technologies. Following the evaluation of our system, we also discuss major challenges and possibilities for the continuing evolution of application mobility systems.**

## I. Introduction

Application mobility is when moving an application from one device to another during runtime, keeping all states and information that is relevant for the user [1]. This brings mobility to the user, in that she can send or fetch applications between the devices she owns or uses. It also requires the services run by the application to be mobile or revoked, along with ongoing session, thus creating strong mobility. In Yu's classification [2], the application can be migrated within a subnet, between a subnet and the Internet, or between heterogeneous third-party controlled networks. The same classification can be done for devices (private or public devices, homogeneous or heterogeneous in nature).

Architectural proposals on how to achieve application mobility do exist (e.g. [3], [4], [5]), but actual deployments and evaluations within the field are very scarce. To prove the viability of a concept, it is our belief that it has to be simulated or, even better, prototyped and then examined. In this paper we present applied application mobility, manifested in a peer-to-peer based system called XAM. The system is designed to be decentralized and functional in heterogeneous environments (multiple networks with multiple service providers), thus differing from traditional systems for application mobility and contributing to state-of-the-art research for this mobility type.

## II. Related Work

The idea of migratory applications was first presented by Bharat and Cardelli in 1995 [1]. They envisioned applications that where not tied to one specific user or device, but could migrate between devices along with context data and user interfaces. This would enable users to bring their applications when switching device, thus increasing user mobility. Then, after migration, a former host should be able to shut down without affecting the application. The concept of context-awareness supported application mobility is explored and validated in [6].

Traditionally, the migration of an application is split into three stages: suspension, migration and resumption [7]. Suspension is when pausing an application on the original host device, saving its states and gathering the relevant information to be migrated. Migration is the actual transfer of byte code from the original host device to the new one, followed by the closing resumption (i.e. making the migrated application run on the new host device). In laboratory settings, users have been comfortable with application migration latencies of 5000 ms [5] and 9000 ms [8].

Among architectural proposals, we find many that are centralized in their layout (e.g. [5], [7], and [9]). A weakness that comes with centralized systems is the dependency of the central node and constant Internet access. A malfunctioning or overloaded central node could render application mobility impossible. Decentralized systems do not suffer from this weakness to the same extent. A general principle of a decentralized system such as a peer-to-peer network [10] is that all nodes can act as both servers and clients. This means that data such as files and information can be shared between peers and accessible throughout the whole network. Therefore, if a peer is removed (or, for that matter, added), the system can continue to function with none or minimal disturbance.

Partial migration of applications – states in particular – is examined in [11]. MDAgent [7] only migrates application logic and data, not the complete application. Yu et al [12] also make use of the MDAgent, measuring what they call suspension, migration and recovery through simulation. The A2M architecture [8] allows a user to migrate an application between heterogeneous devices. The design is decentralized, using a migration manager middleware installed on each potential host device. The experiments involve multiple network interfaces. In [3] the same system is evaluated, but with a different application and in combination with a user study. The application is migrated from a laptop to a mini-PC with WiFi connection and finally onto a desktop computer, while maintaining a videoconference session. The Open Migration Service Platform (MSP) [5] realizes migration of applications

978-1-4799-0540-9/13/$31.00 ©2013 IEEE

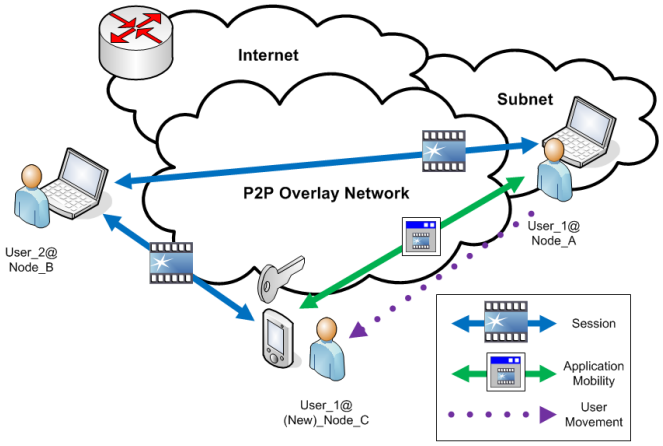


Fig. 1. XAM Architecture

between devices while still maintaining sessions. The system is centralized, requiring access to both a migration server and an application server. Experiments do not cover migration between different networks. Table I summarizes.

There is indeed a lack of published experimental data when it comes to deployments that feature application mobility. Few experiments cover migrations over wireless networks and we have found no publications demonstrating cross-network application mobility in practice. Furthermore, the vocabulary for naming the migration phases differs and there are no standard benchmarks for application mobility.

## III. System Description

System architecture is based on our initial proposal, extensively described in [3]. The architecture aims to fulfill requirements such as ability to handle application distribution and identification, context-awareness, context quality, provision of seamless migration, support for heterogeneous environments and high degree of usability.

The architecture (see Figure 1) is based on the peer-to-peer paradigm. This design results in a decentralized system, where new devices should be easily addable while other devices could be removed in an ad hoc manner. The architecture also supports nodes to connect from outside the local subnetwork, allowing application mobility between heterogeneous networks. Context data is distributed and stored throughout the network, making it available to all peers. This also means that if a device hosting an application is removed from the network, it can automatically locate a peer eligible to become the new host for the application and migrate it accordingly.

Heterogeneity support is desirable, as is mobility. Application mobility should be extended to a multitude of devices and migration should be possible between different networks. The rationale behind the architecture is that the application should run on the device that suits the user best at any given situation. Therefore different kinds of devices in regards of input and output capabilities, level of mobility and other important resources, should be addable to the system.

Our general system design differs from traditional systems as it provides application mobility in a decentralized manner over heterogeneous networks. We call our implementation XAM, featuring peer-to-peer based application mobility with heterogeneous devices and networks. The system consists of four tiers: network infrastructure and devices, an overlay network, a migration manager and the migratable applications. The system was developed using Java.

To migrate an application, there has to be an available network. The type of network does not matter, as long as it is based on TCP/IP. WiFi (802.11), Ethernet (802.3), UMTS and CDMA networks are all examples of networks that can be used as underlying infrastructure. When several connection alternatives are available, a user can be always best connected using the best network interface in a given situation through updating the system with the new IP address. Devices are allocated IP addresses from the network operator and users are identified through fully qualified domain names in the style of user_id@P2Pdomain.

We use a peer-to-peer style overlay network to create a virtual network in which migration can take place. We have chosen Juxtapose (JXTA) [13] as this open source protocol supports different programming languages and allow message exchange independent of the physical network topology.

Devices added to the system become nodes in the peer-to-peer network via so called peer endpoints consisting of available network interfaces. Every node is given a unique Peer id trough the JXTA address allocation mechanism. The nodes are then organized as a JXTA peer group, allowing them to propagate messages throughout the cluster of nodes to which they belong. At least one device in every subnet is given the role of rendezvous peer, making it possible for devices outside the network to connect to the peer group. Concretely, we use JXTA protocols to let the nodes identify each other in the network and send request (this is a prerequisite for finding eligible hosts for an migratable application), retrieve information about other nodes, create virtual communication links between each other, and exchange messages. All protocols are based on the XML format. The message documents are called advertisements and give the nodes a common ”language”. Context data is propagated through advertisements and thus available throughout the whole network. A typical context advertisement in our deployment consist of peer id, information about whether or not the device is hosting a migratable application, user preferences, system status (e.g. battery level, memory and CPU load), input and output capabilities etc.

A middleware orchestrating migrations is installed on each potential host device. This middleware is a further development of the A2M migration manager [8], an open source middleware for application mobility. The middleware is responsible for listening to incoming migration requests and provides a GUI for the user to manage applications, save states, review system status and request to retrieve or store an application. The migration sequence is carried out in the following steps: 1) *Migration initialization*. Migration can be initiated either through push (the user wants to send the application to another

TABLE I
EXISTING EVALUATIONS OF SYSTEMS SUPPORTING APPLICATION MOBILITY

| *System* | *Network properties* | *Mobility Delivered* | *Application Size* | *Migration Latency* |
|---|---|---|---|---|
| Unnamed [11] | Centralized, Single subnet (WLAN) | Partial | 37 kB | 8470–22290 ms |
| MDAgent [7] | Centralized, Single subnet (Ethernet) | Partial | No byte size specified | 1000 ms |
| MDAgent [12] | Centralized, Single subnet (Ethernet) | Full | 32 kB | 450–1529 ms |
| A2M [8] | Decentralized, Single subnet (Ethernet and WLAN) | Full | 2.4 MB | 5879–13481 ms |
| A2M [3] | Decentralized, Single subnet (Ethernet and WLAN) | Full | 2 MB | 4000-11000 ms |
| MSP [5] | Centralized, Single subnet (95 Mbit/s emulator) | Full | Simulated link delay (0-500 ms) | 1000-24000 ms |

device) or pull (a user wants to fetch an application). The latter method can be invoked through the GUI or by using an RFID key ring to inform the system that the user has switched device. The following steps will assume migration initialization through pull; 2) *Application search*. The peer uses the Discovery protocol to find an application Host. Application advertisements containing information about the application and its requirements are collected; 3) *Evaluation of host eligibility*. The application requirements from the received advertisement are compared with the capabilities of the device. If the rules are passed, the device is considered a eligible host; 4) *Migration setup*. A socket is created to allow the migration of the application from the original host to the new device. The Pipe Binding Protocol is used to create a unicast communication channel between the two devices. When the host device receives the socket request it also suspends the application and saves its states; 5) *Application migration*. The application is moved from one device to another along with its states. The states are stored in an external XML-file. This phase lasts until the last ACK is sent and the files are completely written to the new disc; 6) *Socket closure*. The JXTA pipe used for migration is closed; 7) *Post-migration Context advertising*. Context advertisements are propagated to inform the peers of the new host roles; 8) *Application resumption*. States are initialized and the GUI is updated.

For our experiments, we developed a simple gaming application allowing two users to play a game of Battleships online. The default size of the application is 1340 kB while the states (current ship positions, player name, user preferences etc) are stored in a 2 kB XML document.

## IV. EXPERIMENTS, RESULTS AND DISCUSSION

We used three different test beds to evaluate the XAM system. The first experiment consisted of a migration between two laptops (equipped with Intel Core Duo 2 2.26 Ghz CPU, 2 GB RAM and Windows 7 Pro) connected to the peer group through the same WiFi network. The second experiment involved the same two laptops, but the new host was connected to the peer group via a CDMA2000 interface, using one of the WiFi connected laptops as a rendezvous peer. The third experiment had the same setup, but the new host device then consisted of a mini PC (equipped with an Intel Atom 1.6 Ghz CPU, 2 GB RAM and Vista Home Premium). In all experiments, an ongoing game of Battleships was played out between the initial host and another user (Laptop 1), and then resumed after migration to the new host device. Table II summarizes the network properties.

TABLE II
NETWORK THROUGHPUT (MEAN VALUES)

| *Test bed* | *Network* | *Downlink* | *Uplink* | *Delay* |
|---|---|---|---|---|
| Test bed 1 | WiFi (802.11g) | 13.38 Mbit/s | 15.22 Mbit/s | 24 ms |
| Test bed 2 | WiFi (802.11g) | 12.51 Mbit/s | 15.23 Mbit/s | 26 ms |
| | CDMA2000 | 0.28 Mbit/s | 0.27 Mbit/s | 117 ms |
| Test bed 3 | WiFi (802.11g) | 13.01 Mbit/s | 16.19 Mbit/s | 16 ms |
| | CDMA2000 | 0.48 Mbit/s | 0.19 Mbit/s | 129 ms |

TABLE III
TEST RESULTS (MEAN VALUES)

| *Test bed* | *Susp.* | *Migration* | *Resum.* | *CM/CA* | *Total latency* |
|---|---|---|---|---|---|
| Test bed 1 | 31 ms | 6773 ms | 213 ms | 3671 ms | 10689 ms |
| Test bed 2 | 34 ms | 68386 ms | 228 ms | 6783 ms | 75430 ms |
| Test bed 3 | 38 ms | 57519 ms | 692 ms | 6183 ms | 64432 ms |

In our measurements of migration latency, the following definitions were used: *Suspension*: The time it takes to store application states. This is performed on the original host in parallel with Migration setup; *Migration*: The time it takes to open a socket between the host device and the host to-be plus the time it takes to migrate the application and its states plus the time it takes to close the socket; *Resumption*: The time it takes to load the application states and update the GUI on the new host; *Context Management/Context-awareness*: The time it takes to find and evaluate the eligibility of the potential host device and update the system with the context changes. Context propagation involves both pre- and post-migration context advertisements and calculations.

When migrating over a local network (test bed 1, see Figure 2) the total migration latency was approximately 10 seconds in average. 3500 ms was due to Context-awareness features. Note that the Migration bar in Figure 2 contains three sections; these correspond to migration setup (blue), application migration (red) and socket closure (green). Also, the Context management/Context-awareness (CA) bar contains two sections, which correspond to pre- (blue) and post-migration (red) context advertisements and calculations. Test beds 2 and 3 included a secondary network, with cellular technology. Migration time was considerably higher in these test rounds due to the low bandwidth.

The XAM system provides full application mobility, i.e. the migration of application code, states and related information during execution. Through our decentralized architecture, we achieve a lower degree of network dependency compared to centralized systems (e.g. [5] and [9]) and also simplify addition and removal of new nodes. The vulnerability to the

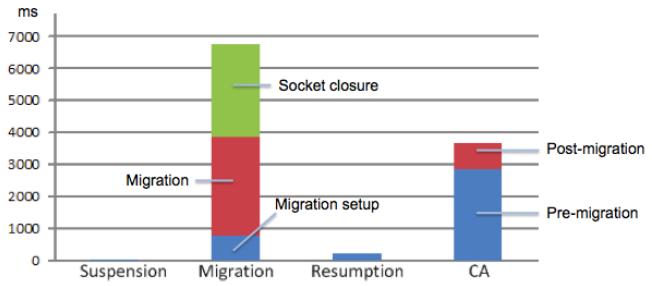


Fig. 2. Total Migration Latency (ms), Test bed 1

loss (or removal) of key nodes decreases as context data is distributed throughout the network. For application mobility to be useful outside the laboratory setting, it must support heterogeneous devices and networks. As these networks can be controlled by different network providers and have different configurations and limitations, our architectural proposal was created with a real world setting in mind, not restricting its deployment to a laboratory, LAN or other user controlled network infrastructure. Thus XAM differs from all related work presented in Section II. Architectures and deployments such as [5], [7], [8], [9], [11], and [12] are all application mobility approaches, not designed for and/or not evaluated using heterogeneous networks with different service providers.

During experiment setup, it was obvious that the traditional suspension-migration-resumption phases did not cover all the operations needed to perform application migration in a context-aware, cross-network scope. As no benchmarks for application mobility exist, we believe it is crucial to break down the different steps in the migration process to ensure validity. We propose that the definitions presented in the previous section should serve as guidelines on how to declare measurements of migration latency.

We use three different sources of input to inform application migration: device context (e.g. input/output capabilities and system resources), user context (e.g location and preferred host device) and application context (requirements etc.). In a larger setting, context could include more values but also more parameters. As the XAM context-awareness motor is fairly simple, we believe that heavier calculations should be left to a standalone module, entirely dedicated to context awareness. The peer-to-peer network could still be used for propagation, but to minimize migration latency, calculations and context compilation should be handled by a Context Manager, e.g. a middleware communicating with the Migration Manager.

Migration times within a single network are acceptable when comparing them to the system performance expected by the user. Cross-network measurements however show that network quality is a major factor when it comes to migration latency. A network offering low bandwidth inevitably results in long migration times. User studies should be conducted to examine the acceptance of latency when migrating between different networks. Our hypothesis is that user tolerance of migration latency increases in settings outside a local network.

## V. Conclusion

In this paper, we presented a decentralized system for application mobility, supporting heterogeneous devices and networks, thus differing from traditional centralized application mobility systems deployed in single subnets. A prototype was designed and evaluated. System performance was compared with recognized requirements for application mobility, showing the viability of both the concept and our chosen architecture. Additional outcomes of our experiments were that the traditional migration phases of suspension-migration-resumption proved to be insufficient to describe the migration process. We also discussed major challenges and possibilities for the continuing evolution of systems supporting application mobility. Future work should target these areas.

## Acknowledgment

This work was supported by the NIMO project (see www.nimoproject.org), funded by EU Interreg IVA North.

## References

[1] K. A. Bharat and L. Cardelli, “Migratory applications,” in *Proceedings of the 8th annual ACM symposium on User interface and software technology*, UIST ’95, (New York, NY, USA), pp. 132–142, ACM, 1995.

[2] P. Yu, X. Ma, J. Cao, and J. Lu, “Application mobility in pervasive computing: A survey,” *Pervasive and Mobile Computing*, vol. 9, no. 1, pp. 2–17, 2013.

[3] D. Johansson, A. Åhlund, and C. Åhlund, “A mip-p2p based architecture for application mobility,” in *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, MUM ’11, (New York, NY, USA), pp. 85–93, ACM, 2011.

[4] I. Satoh, “Self-deployment of distributed applications,” in *Scientific Engineering of Distributed Java Applications* (N. Guelfi, G. Reggio, and A. Romanovsky, eds.), vol. 3409 of *Lecture Notes in Computer Science*, pp. 48–57, Springer Berlin / Heidelberg, 2005.

[5] K. Hojgaard-Hansen, H. C. Nguyen, and H. Schwefel, “Session mobility solution for client-based application migration scenarios,” in *Proceedings of the Eighth International Conference on Wireless On-Demand Network Systems and Services*, WONS, pp. 76–83, jan 2011.

[6] D. Johansson and M. Wiberg, “Conceptually advancing ”application mobility” towards design: Applying a concept-driven approach to the design of mobile it for home care service groups,” *International Journal of Ambient Computing and Intelligence*, vol. 4, pp. 20–32, jul 2012.

[7] Y. Zhou, J. Cao, V. Raychoudhury, J. Siebert, and J. Lu, “A middleware support for agent-based application mobility in pervasive environments,” in *Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops*, (Los Alamitos, CA, USA), p. 9, IEEE Computer Society, 2007.

[8] A. Åhlund, K. Mitra, D. Johansson, C. Åhlund, and A. Zaslavsky, “Context-aware application mobility support in pervasive computing environments,” in *Proceedings of the 6th International Conference on Mobile Technology, Applications & Systems*, Mobility ’09, (New York, NY, USA), pp. 21:1–21:4, ACM, 2009.

[9] A. Ranganathan, C. Shankar, and R. Campbell, “Application polymorphism for autonomic ubiquitous computing,” *Multiagent Grid Syst.*, vol. 1, pp. 109–129, Apr. 2005.

[10] J. F. Koegel Buford, H. H. Yu, and E. K. Lua, *P2P networking and applications*. Amsterdam: Elsevier/Morgan Kaufmann, 2009.

[11] K. Zhang and S. Pande, “Minimizing downtime in seamless migrations of mobile applications,” in *Proceedings of the 2006 ACM SIGPLAN/SIGBED conference on Language, compilers, and tool support for embedded systems*, LCTES ’06, (New York, NY, USA), pp. 12–21, ACM, 2006.

[12] P. Yu, J. Cao, W. Wen, and J. Lu, “Mobile agent enabled application mobility for pervasive computing,” in *Proceedings of the Third international conference on Ubiquitous Intelligence and Computing*, UIC’06, (Berlin, Heidelberg), pp. 648–657, Springer-Verlag, 2006.

[13] Project Kenai, “Jxta. the language and platform independent protocol for p2p networking,” nov 2011.