Session Mobility in the Mockets Communication Middleware

Cesare Stefanelli, Mauro Tortonesi
{cstefanelli, mtortonesi}@ing.unife.it
Engineering Department, University of Ferrara,
Ferrara, Italy

Erika Benvegnù, Niranjan Suri
{ebenvegnu, nsuri}@ihmc.us
Florida Institute for Human & Machine Cognition
Pensacola, FL, USA

Abstract

Taking advantage of the benefits of modern networking, a growing number of users are exhibiting mobile behavior. As they roam between different network localities, they access the Internet and the Web exploiting both wired and wireless communications and using several heterogeneous devices. Mobile users want to access their subscribed services anywhere, anytime, and want to preserve their currently opened service sessions as they roam between different network localities or switch between different devices. Mobile users' requirements call for novel middlewares to provide support for mobility on top of the traditional Internet infrastructure. In this context, we have developed Mockets, a communication middleware specifically designed to address the challenges of wireless networks and mobile computing. In particular, Mockets supports session mobility in terms of seamless handover for preservation of end-to-end connectivity in spite of node mobility, automatic detection and exploitation of best available connectivity, and migration of service session endpoints from one node to another.

1. Introduction

Modern networking technologies enable users to be on-line while moving. As they roam between different network localities, they access the Internet and the Web exploiting both wired and wireless connectivity, e.g., via IEEE 802.3, IEEE 802.11, and Bluetooth. In addition, users exploit several heterogeneous devices, e.g., smart phones, PDAs, laptops, and desktops. In this scenario, mobile users want to access their subscribed services anywhere, anytime. This introduces new and challenging requirements because users want to preserve their currently opened service sessions as they roam between different network localities and even when they switch to a different device.

The requirements of mobile users are not satisfied by the traditional Internet infrastructure, which is the substrate upon which most applications are constructed. In fact, the Internet Protocol Suite does not support mobility of service sessions, thus causing the interruption of established services upon node migration to a different network location or when switching to a different device [1]. Mobile users' requirements call for novel communication middlewares for the support of service sessions that are resilient to both user and node mobility. When built on top of the Internet Protocol Suite, the new middlewares could achieve high portability and easy deployment, although the Internet networking environment requires to take security in due consideration.

In this context, we have developed Mockets, a communication middleware specifically designed to address the challenges of wireless networks and mobile computing [2] [3]. Mockets (that stands for mobile sockets) is an application-level library that resides on top of the operating system and communication protocol stack and enables the realization of mobile distributed applications.

In particular, this paper presents the Mockets support for session mobility, providing a theoretical characterization of mobile service sessions, an in-depth overview of the Mockets features enabling session mobility, and the detailed description of the architecture and implementation of our framework.

Mockets preserves end-to-end connectivity of service sessions in spite of node mobility, allowing users to traverse different networks without breaking their open service sessions. In addition, Mockets automatically takes advantage of best network

978-1-4244-2703-1/08/$25.00 ©2008 IEEE

connectivity in case mobile nodes enter locations where several networks overlap or multiple network attachment points are simultaneously available. Finally, Mockets enables the migration of session endpoints to different nodes, allowing users to preserve their subscribed service sessions when switching to a different device. The Mockets support for session mobility enables the realization of novel distributed applications which are better suited to the wireless Internet environment.

Mockets-based mobile sessions have already been adopted in several mobile agents applications on the NOMADS platform [4], thus enabling mobile agents to migrate from host to host while maintaining their connections, as well as in dynamic service oriented architectures with Agile Computing [5], where services are migrated from one node to another without interrupting the communication with their clients. In addition, Mockets has been successfully used by the Army Research Laboratory as part of the Warrior's Edge initiative of the Horizontal Fusion Portfolio's Quantum Leap demonstrations and in the C4ISR On The Move exercise.

2. Supporting Mobility in the Wireless Internet

The Internet infrastructure has greatly changed in recent years. Thanks to the advances in wireless and cellular technologies, the traditional broadband/cabled Internet is now complemented by many local wireless and cellular networks, thus creating a scenario commonly referred to as the wireless Internet [6]. Users can now easily move in the wireless Internet while accessing on-line services. For instance, they can travel from their homes to airport/railway/underground stations, to their company buildings, and finally to their offices, dynamically exploiting a wide range of different and sometimes overlapping networks, e.g., a GPRS/UMTS cellular network, an IEEE 802.11 municipal WLAN, and an IEEE 802.3 office LAN. In practice, mobile users typically take advantage of a combination of wired and several wireless networking technologies, with many using wireless networks unless they require the high-bandwidth provided by wired networks. In addition, mobile users exploit a plethora of heterogeneous devices, e.g., smart phones, PDAs, laptops, and desktop PCs.

Mobile users in the wireless Internet have new requirements which were not supported by the traditional Internet infrastructure. Not only do mobile users want to access their subscribed services anywhere, anytime, but they want their currently opened service sessions to be continuously available as they roam between different network localities. In addition, users want to exploit the best available network connectivity in a transparent and uninterrupted manner. Finally, users want to be able to switch to different devices while preserving the currently opened service sessions. In short, there is the need to support *mobile service sessions*, i.e., service sessions that can continue even in the presence of both user and node mobility or when changing network connectivity.

Session mobility is not addressed in traditional IP-based middlewares and calls for the introduction of novel middlewares better suited to the wireless Internet environment and to mobile users requirements. While there are many issues to be considered when dealing with middlewares supporting mobile service sessions, such as location/tracking, discovery, and security, this paper focuses only on the main issue of supporting session mobility. In particular, mobile sessions should not break as nodes roam between network localities (preserving end-to-end connectivity), they should dynamically and automatically rebind to different networks or network interfaces in order to take advantage of the best available connectivity (automatic network selection), and the mobile session endpoints should be able to migrate to a different node (endpoint migration).

Preserving end-to-end connectivity of service session in spite of terminal mobility requires support for session handoff between different network layer addresses. Not only does this require a mechanism to change the network layer address of the session without tearing it down and restarting it, but it also requires notification of changes to the peer endpoint after the handoff. In addition, to promptly respond to node mobility, the mobility middleware must continuously monitor the current network attachment point in order to detect changes in network access point, and to react accordingly.

The need for automatic network selection occurs because mobile nodes might enter locations where several network overlap, or where several communication links using different technologies, e.g., IEEE 802.11 and Bluetooth, are simultaneously available. Mobility middleware should detect the best available network connectivity and should perform the handover of all the currently open sessions.

Finally, the case for session endpoint migration arises from the fact that mobile users typically exploit different devices in their roaming, e.g., a PDA while travelling, a desktop PC in the office, and a laptop at home, frequently switching between them. Session endpoint should move from one host to another so that

users can preserve their subscribed service sessions when switching to a different device, without having to close and restart them. This needs mechanisms to suspend a connection, to save its state, to transfer it to the new node, and finally to resume the connection from the saved state.

3. Related Work

Many research studies have addressed the problem of preserving end-to-end connectivity of service sessions on mobile nodes. Network-level proposals, such as Mobile IP/IPv6 and HIP, try to disambiguate the dual nature of locator and identifier of IP addresses upon which the Internet Protocol Suite is built [1]. In particular, they assign unique identifiers (Home Address or HIP Address) to mobile nodes, allowing them to change their IP address as they traverse different networks without breaking open service sessions, and introduce special entities (Home Agent, Rendezvous Server) for node mobility management, e.g., location and tracking. Transport layer proposals instead take a different approach, by introducing explicit support for session handoff in the communication protocols or routing service sessions through mobile proxies deployed at the edge between the wireless and wired portion of the network [7] [8]. All the above mentioned solutions require modifications at the operating system level, thus significantly hindering their adoption. Application layer proposals, typically built on top of SIP [9], seem to be more suited for the heterogeneous wireless Internet environment.

The automatic best network selection problem has also been intensively investigated recently. Some proposals, such as [10], introduce at the mobile nodes a component which monitors available network attachment points and automatically takes advantage of the best one. Other proposals, such as MUM [11], also consider QoS- and context-awareness, and leverage on mobile proxies to provide best available connectivity for subscribed service sessions. However, this approach requires a careful engineering of mobile proxies to reach a satisfying level of performance, and poses significant challenges for redeployment of proxies to follow mobile nodes in their roaming, as proxies can only migrate to preconfigured locations.

Finally, other research studies focus on the migration of session endpoints. [12] and [13] respectively build on top of SIP re-INVITE/REFER functions and on an underlying P2P routing infrastructure to implement session transfer. However, these proposals only address the problem of session handoff between different nodes but do not consider the migration of session state as well. In addition, these solutions are designed to support always-on nodes and do not consider temporary network disconnections, e.g., they do not provide mechanisms to temporarily suspend sessions and to resume them later.

4. The Mockets Communication Middleware

Mockets is a communication middleware specifically designed to address the peculiar challenges of mobile users in the wireless Internet [2] [3]. The middleware provides applications with the mocket abstraction. A mocket is an entity representing a mobile communication endpoint, like a traditional BSD socket, and can support both stream-oriented and message-based communications.

Mockets is realized as an application level middleware, as the middleware approach allows overcoming the impedance mismatch between applications and the network stack without sacrificing portability. In fact, by introducing additional intelligence at the application level, the middleware solution supports applications with richer communication semantics and network conditions monitoring functions. At the same time, by building on top of the Internet Protocol Suite, the middleware approach also guarantees considerable portability as it allows the deployment of distributed applications on all platforms supporting a TCP/IP stack, regardless of the underlying hardware and operating system.

The Mockets mobility support at the application level is particularly convenient, as it does not require a preinstalled infrastructure such as Mobile IP, Mobile IPv6, and split-connection based solutions. In addition, the end-to-end approach allows Mockets to combine the benefits of transport layer mobility, e.g., low handoff latency, with the benefits of application-layer mobility, e.g., knowledge of mobility events and security improvements.

Mockets is integrated with the Agile Computing Infrastructure [14] to provide applications with information about the current environment, e.g., changes in the set of neighboring nodes and available resources and services, and with the KAoS policy framework for realization of policy-based QoS enforcement.

5. Mockets Session Mobility

Mockets is designed to support mobile service sessions, in order to facilitate the design and

deployment of novel distributed applications better suited to mobile users in the wireless Internet. Central to the support of mobile service sessions is the concept of the mocket endpoint and its behaviour in presence of mobility. First of all, a mocket endpoint can be dynamically rebound to permit the preservation of end-to-end connectivity in spite of node mobility. The Mockets middleware can also automatically take advantage of the best network connectivity at any time, selecting the network attachment point and network interface which represent the best connection to the wireless Internet. Finally, Mockets supports migration of session endpoints to different nodes.

5.1. Preservation of End-to-End Connectivity

Mockets preserves end-to-end connectivity of service sessions in spite of terminal mobility. In particular, Mockets detects changes in network attachment point, and automatically performs rebinding of session endpoints to the new network layer address. In fact, Mockets-based service sessions have virtual endpoints that can be dynamically rebound to different network layer addresses. This feature of Mockets is fundamental to permit applications to continue their execution even in the presence of node mobility, without forcing them to shut down and to reopen all their network connections.

et us notice that session mobility introduces the problem of locating devices when both endpoints simultaneously perform a session handoff. In fact, in this case both endpoints try to notify the change of network layer address to the remote endpoint, but their attempts may fail as they are sending messages to an old (and possibly unreachable) network layer address. Mockets provides mechanisms for session mobility, but currently does not solve the above mentioned problem by enabling the location and tracking of mobile session endpoints.

5.2. Automatic Best Network Selection

In addition, the Mockets middleware can dynamically and automatically provide best network connectivity by rebinding the open sessions to the network attachment point and network interfaces which provide the best performance. Mockets can continuously monitor the status of all network layer addresses and network interfaces on mobile nodes and select the best ones according to various heuristics. The middleware then automatically rebinds the endpoints of all opened sessions to the address and interface with represent the best connection to the wireless Internet. To this end, Mockets can exploit the information about network interfaces, e.g., signal levels for wireless interfaces, provided by the X ayer substrate [1].

At the moment, the automatic best network selection feature of Mockets is still under development. In particular, we are currently evaluating both simple heuristics for session handoff decisions, such as the azy ell Switching and Eager ell Switching strategies adopted in most Mobile IP and Mobile IPv implementations, and more sophisticated heuristics, e.g., predictions on future availability of a particular network attachment point based on monitoring of SSI level of wireless network interfaces. In fact, handover strategies can have a significant impact on the o S provided to applications, e.g., in terms of available bandwidth and or latency, and not every strategy might be the best suited for every situation.

5.3. Endpoint Migration

Mockets also allows the migration of session endpoints to a different node. More specifically, the middleware permits applications to suspend an open mocket and retrieve its state as an opaque object. Applications can then request the Mockets middleware to resume the previously suspended mocket by providing the corresponding object. Applications can also move the object representing the saved session state to another node and resume the previously suspended session from there.

The endpoint migration feature is fundamental for the development of applications which enable users to migrate all their subscribed services to a different node while preserving service continuity. Mockets does not currently automate the transfer of session and application state to another node, but its support for endpoint migration enables applications to easily realize it.

Endpoint migration also has interesting applications in mobile code systems. By taking advantage of Mockets-based session endpoint migration, mobile agent platforms could allow mobile agents to move to a different location while carrying not only its current context of execution but also its bindings with the local environment, including all network connections. This solution is currently being used in the MADS platform, which supports weak, strong and forced agent mobility [ ].

inally, endpoint migration is very important in Pervasive o mputing, where servers could perform transparent hand-overs to client applications as mobile nodes roam, in order to provide contextual and location-based service provisioning.

5.4. Architecture for Mobility Support in Mockets

The Mockets middleware has a modular architecture and builds on top of the traditional T P IP protocol suite. This design guidelines permit to achieve portability, extendibility, ease of integration, and facilitates the Mockets deployment in all available platforms and scenarios, regardless of the underlying hardware and operating system.

In the Mockets architecture, as depicted in igure 1, the Session Manager (SM) is the component which implements mobility of service sessions. SM is composed of 3 modules: the and off Manager (M), the Migration Manager (MM), and the o ordinator.

The andoff Manager module provides mechanisms for session handoff. When requested to perform a session handoff to another network layer address and or network interface, M rebinds the local mocket endpoint and triggers the transmission of handoff notification to the peer endpoint by interacting with the Datagram Transmitter and eceiver (DT) co mponent.

The Migration Manager module provides mechanisms for endpoint migration. More specifically, MM performs suspension and resumption of connections, interacting with DT to carry out the suspend esume protocol. MM also performs serialization and deserialization of the mocket state, providing applications with the opaque connection state object as a result.

The o ordinator module supervises and coordinates the session management functions provided by M and MM in order to realize session mobility. More specifically, the o ordinator enables endpoint migration by processing application requests and suspension requests from the peer endpoint, and carrying them out leveraging the functions provided by MM.

The oordinator also realizes preservation of end-to-end connectivity and automatic best network selection by leveraging information received from the et work onditions Monitor (M) component. M monitors network status, processes the collected information and provides it both to oordinator and to applications. In particular, M detects changes in the network layer address of a device and detects peer unreachability via a keep-alive mechanism. In addition, the M component integrates with the X ayer substrate, where available, to obtain cross-layer information from network interfaces, e.g., signal strength, reliability, network load, etc.

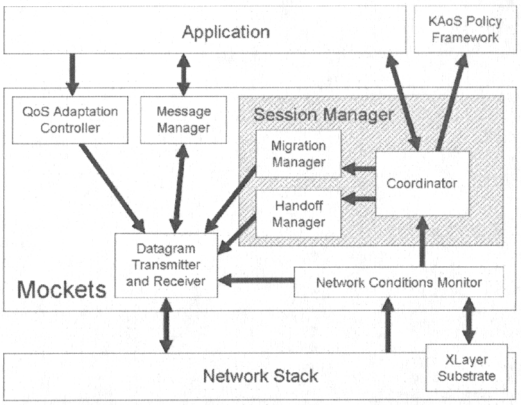

Figure 1. Architecture of Mockets

6. Mockets Session Manager Implementation

The Session Manager component manages service sessions and enables their mobility. It is composed of the oordinator, the M, and the MM modules. The M module implements the mechanisms for supporting session handoff, while the MM module mainly deals with session suspend resume. The Session Manager and its modules are developed as ++ components, along with the rest of the Mockets middleware. Mockets also supports applications written in Java and # via J I and Managed ++ wrappers. In the case of Java, the Mocket class (that applications use as the connection endpoint) implements the Serializable interface, making it easy for applications to serialize the connection endpoint for migration.

6.1. Session Handoff in the Handoff Manager

The M module contains the session handoff mechanism that performs the rebinding of Mockets service sessions endpoints to different network layer addresses. When M receives a handoff request from the o ordinator, it informs the DT component to start using the new network layer address for message transmission reception. In order to minimize potential message losses caused by the handoff, DT also keeps listening for messages on the previous network layer address, until a predetermined interval of time (currently set to 2 * T T) is elapsed from the reception of the first packet to the new network layer address.

M then requests DT to transmit an address change notification to the peer session endpoint. Address change notifications contain authentication information that allows the peer endpoint to verify that the message was not forged by using a secret key exchanged at connection establishment time, thus

preventing hijacking of open service session from malicious entities. The notification is periodically retransmitted until the first packet to the new network layer address is received.

inally, let us notice that session handovers do not cause any change in service session state. Applications running on both the session endpoints can continue to exchange messages. owever, handovers trigger the reset and recalculation of some session-related statistics, e.g., T T.

6.2. Session Suspend/Resume in the Migration Manager

The MM module provides operations for session suspend esume, to enable the suspension of a service session, the retrieval of the state of suspended service session, and the resumption of a previously suspended session. In particular, Mockets provides applications with three primitives: suspend which suspends a currently working connection, getState which returns the entire state of the session (including messages in the transmission receival queues) as an opaque object, and resume which reinstates a connection from its previously serialized state.

The realization of mobile sessions, whose endpoint can be migrated to another node, requires the modification of session state machine. In particular, new states must be introduced to support the suspension and resumption of running sessions. igure 2 depicts the state machine of Mockets-based session.

pon receiving a session suspension request from the application, the oordinator orders the MM component to initiate the suspension procedure for the current session. MM changes the session state from ESTAB IS E D to S SP E D_PE DI . The session remains in the S SPE D_PE DI state until all the messages in the transmission queue are flushed or a timeout expires, according to the application preference. MM then interacts with the DT component to trigger the transmission of a suspended session notification message and changes the session state to S SP E D_ SE T. pon receiving a suspended session acknowledgement notification from the peer endpoint, the suspension procedure is complete and session state is changed to S SP E DED.

Applications can then retrieve the state of the suspended session. In this case, MM performs the serialization of the entire state of the session, and returns it to the application as an opaque object. It is up to the application to transfer the suspended session state to another node before resuming the session from there. et us notice that serialization is a generic feature that could be later extended to support other needs, e.g., distributed checkpointing.

When the application calls resume, the oordinator component creates a new session endpoint and then orders MM to change the current session state to the previously saved one. This triggers the transmission of a session resumption notification to the peer endpoint, which is periodically retransmitted until a timeout occurs (similar to the timeout for connection establishment).

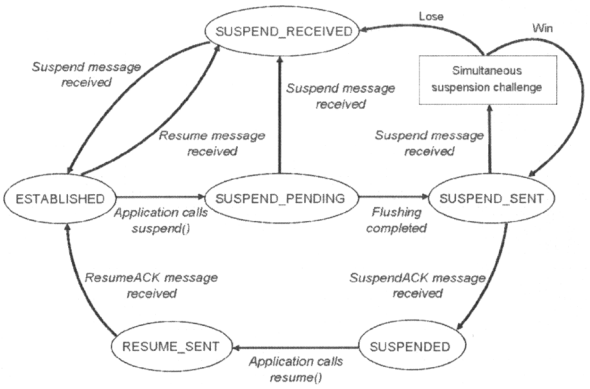

Figure 2. State diagram for suspend/resume of Mockets connections

n the other session endpoint, oordinator processes messages related to session suspend resume. pon receiving a session suspend notification, the o ordinator notifies the application that a suspension request has been received from the peer endpoint and orders MM to initiate a passive session suspension. MM in turn changes the session state to S SPE D_ E EIVED. The session remains in this state until a session resume notification is received.

7. Experimental Evaluation

This section presents the first experimental results of the Mockets support for session mobility. Testing the overall performance of session migration in a reproducible way is extremely difficult because of the large number of parameters which contribute to its determination (network conditions, size of message transmission and reception queues at the session endpoints, etc.), the dynamics of signaling involved in endpoint migration, and the difficulty in isolating the impact of the various contributions. or these reasons, the measures of the performances in a real wireless

networking environment are significantly influenced by the particular topology and conditions of the test network. As a result, we have decided to present here the performance of the session suspend resume mechanism, that is the key contribution to the overall performance. These measurements are taken in a reproducible and controlled laboratory environment.

We have performed the experiments in a testbed composed of 2 Pentium III 900Mhz desktop P s with 12MB A M, running edora ore inux eleas e (kernel version 2. .11). The P s are interconnected via a 100 Mbps IEEE 802.3 wired network. Both P s run I ST et [1], a network emulator which can apply several effects to the outgoing traffic flow in order to reproduce the behavior dynamics of the wireless Internet scenario. In particular, we have configured the I ST et instances to apply a round trip time delay between the two P s and to enforce several levels of packet loss rate in order to emulate a networking environment with the highly unreliable communications typical of the wireless Internet.

Table 1 shows the average time (out of 100 iterations) that it takes to complete the session suspension and then resume it from the same node. The proposed metric does not take into account the time to retrieve the opaque object representing the suspended session state. In fact, this last time depends on the amount of data in the transmission and reception queues, and is therefore extremely volatile and difficult to retrieve in a reproducible fashion and in the context of a realistic scenario.

otice that as the delay and the packet loss rate increase, both the times to suspend and resume the service session increase. This is because network delay and packet loss disturb the notification messages exchanged between the endpoints in the session suspend esume procedure. Also note that the suspend time is usually larger than the resume time, because the Mocket endpoint waits to flush any data in the output buffer before suspending the connection. This reduces the size of the state information when migrating the connection state from one node to another.

Although the code of the Mockets prototype used for the tests is still in a development stage and has not been optimized yet, the results of our tests seem promising. In particular, they show that, even in presence of significant packet loss, the time to suspend and resume a Mockets-based service session is comparable to IEEE 802.11b [17] and Mobile IP [18] handoff times. Therefore, the process of migrating a session endpoint can be reasonably fast, at the point that users will not experience a significant loss of service liveness in case of a session endpoint migration.

Table 1. Time to suspend and resume a Mockets service session

Round Trip Time Delay	Packet loss rate	Average time for session suspension (ms)	Average time for session resumption (ms)
No delay	0%	353	34
100 ms	0%	417	132
100 ms	10%	1023	385
100 ms	20%	1333	712
200 ms	0%	553	243
200 ms	10%	1197	464
200 ms	20%	2292	803

8. Conclusions and Future Work

The Mockets middleware has been designed specifically to address the requirements of mobile users in the new wireless Internet scenario. The Mockets support for service session migration makes it possible to preserve end-to-end connectivity in spite of node mobility, to automatically exploit the best available connectivity, and to migrate a service session endpoint to another node.

We are planning to extend the Mockets middleware in several directions. irst of all, we are working to integrate components for location tracking of mobile nodes sessions. In addition, we are already working on enhanced heuristics for session handoff between different networks (IP addresses) and or network interfaces. The new heuristics should be integrated with a policy system, allowing users to define policies and algorithms for handoff strategies.

9. Acknowledgements

This work is supported in part by the . S. Army esearch aboratory under ooperative Agreement W911 - 0 -2-0013, by the .S. Army esearch aboratory under the ollaborative Technology Alliance Program, ooperative Agreement DAAD19-01-2-0009, by the Air orce esearch aboratory under ooperative Agreement A87 0-0 -2-00 , and by the Italian MI in the framework of the Project "M MA: a middleware approach to M b ile MultimodAl web services

10. References

[1] X. Fu, D. Hogrefe, D. Le, "A Review of Mobility Support Paradigms for the Internet", IEEE Communications Surveys and Tutorials, Vol. 8, N. 1, pp. 38-51, 2006.

[2] M. Tortonesi, C. Stefanelli, N. Suri, M. Arguedas, M. Breedy, "Mockets: A Novel Message-oriented

Communication Middleware for the Wireless Internet", in Proceedings of International Conference on Wireless Information Networks and Systems (WINSYS 2006), Setúbal, Portugal, August 2006.
[3] C. Stefanelli, M. Tortonesi, M. Carvalho, N. Suri, "Network Conditions Monitoring in the Mockets Communications Framework", in Proceedings of 26th Military Communications Conference (MILCOM 2007), Orlando, FL, USA, October 2007.
[4] N. Suri, J. Bradshaw, M. Breedy, P. Groth, G. Hill, R. Jeffers, "Strong Mobility and Fine-Grained Resource Control in NOMADS", in: Proceedings of the 2nd International Symposium on Agents Systems and Applications and the 4th International Symposium on Mobile Agents (ASA/MA 2000), Springer-Verlag, 2000.
[5] N. Suri, M. Rebeschini, M. Arguedas, M. Carvalho, S. Stabellini, M. Breedy, "Towards an Agile Computing Approach to Dynamic and Adaptive Service-Oriented Architectures" in Proc. of the IEEE Workshop on Autonomic Communication and Network Management (ACNM'07).
[6] N. Banerjee, Wei Wu, S.K. Das, "Mobility support in wireless Internet", IEEE Wireless Communications, Vol. 10, No. 5, Oct. 2003, pp. 54-61.
[7] M. Atiquzzaman, A. Reaz, "Survey and Classification of Transport Layer Mobility Management Schemes" in Proceedings of 16th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications, September 11 - 14, 2005, Berlin, Germany.
[8] M. Bernaschi, F. Casadei, P. Tassotti, "SockMi: a Solution for Migrating TCP/IP Connections", in Proceedings of 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'07), 2007.
[9] N. Banerjee, A. Acharya, S. K. Das, "Seamless SIP-Based Mobility for Multimedia Applications", IEEE Network, Vol. 20, No. 2, March/April 2006, pp. 6-13.
[10] S. Lahde, M. Doering, L.Wolf, "Dynamic transport layer handover for heterogeneous communication environments", Computer Communications, Vol. 30, No. 17, November 2007, pp. 3232-3238.
[11] P. Bellavista, A. Corradi, L. Foschini, "Context-Aware Handoff Middleware for Transparent Service Continuity in Wireless Networks", Elsevier Pervasive and Mobile Computing Journal, Vol. 3, No. 4, pp. 439-466, Aug. - 2007
[12] K. Oberle, S. Wahl, A. Sitek, "Enhanced Methods for SIP based Session Mobility in a Converged Network" in Proceedings of Mobile and Wireless Communications Summit 2007, 1-5 July 2007, pp. 1-5.
[13] T. Elsayed, M. Hussein, M. Youssef, T. Nadeem, A. Youssef, L. Iftode, "ATP: autonomous transport protocol", in: Proceedings of the 46th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS '03), Vol. 1, pp. 436- 439, 27-30 Dec. 2003.
[14] N. Suri, J. Bradshaw, M. Carvalho, T. Cowin, M. Breedy, P. Groth, R. Saavedra, "Agile computing: bridging the gap between grid computing and ad-hoc peer-to-peer resource sharing", in Proceedings of 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003), pp. 618-625, 12-15 May 2003.
[15] M. Carvalho, N. Suri, V. Shurbanov, E. Lloyd, "A Cross-layer Network Substrate for the Battlefield", in Proceedings of the 25th Army Science Conference, Orlando, FL, USA, 2006.
[16] M. Carson, D Santay. "NIST Net – A Linux-based Network Emulation Tool", ACM SIGCOMM Computer Communication Review, Vol. 33, No. 3, 2003, pp. 111-126.
[17] H. Velayos, G. Karlsson, "Techniques to reduce the IEEE 802.11b handoff time", in Proceedings of IEEE International Conference on Communications 2004, Vol. 7, pp. 3844-3448, 20-24 June 2004.
[18] R. Hsieh, A. Seneviratne, "A Comparison of Mechanisms for Improving Mobile IP Handoff Latency for End-to-End TCP", in Proceedings of the 9th annual International Conference on Mobile Computing and Networking, pp. 29-41, San Diego, CA, USA, 2003.