Seamless Network Migration Using the Mockets Communications Middleware

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

Mauro Tortonesi
Engineering Department
University of Ferrara
Ferrara, Italy
mtortonesi@ing.unife.it

Tomás Esterrich III
Tactical ISR Division
The Charles Stark Draper Laboratory
Cambridge, MA
testerrich@draper.com

***Abstract*—Due to the underlying dynamic nature of wireless and ad-hoc communication environments, applications in tactical networks have to cope with network disconnections and communication channel degradations. Support for network migration, i.e., the ability of a node to seamlessly change its network attachment point without disrupting service sessions, is therefore essential to provide seamless connectivity to warfighters and unmanned autonomous systems. This much needed utility calls for novel middleware to provide mobility functions on top of the traditional communication infrastructure. The goal of this paper is to present the network migration support in Mockets, a communication middleware specifically designed to address the needs of tactical environments by providing handover capabilities transparently and seamlessly to applications. Mockets automatically takes advantage of the best available network, leveraging on predefined, as well as user-provided, policies to perform migration decisions according to the current network status. The performance tests we conducted demonstrate the effectiveness of our implementation and show that Mockets-based network migrations cause an average measured downtime of 17 milliseconds. From a user's perspective, this short interval makes service appear uninterrupted.**

Keywords-component: migration; handover; session mobility; Mocket

I. Introduction

As tactical edge networks move toward portable cellular infrastructures ranging from fixed installations to base stations mounted on HMMVWs, UAVs, and other mobile platforms, edge users will be expected to roam across these multiple networks in a transparent manner. In addition, when more than one network is available with overlapping coverage, users expect to be able to take advantage of the best connection available with an automatic network migration that seamlessly rebinds the connections over time. Typical handover protocols, such as MobileIP, create a temporary loss of connectivity during a handover. The network infrastructure they require for tunneling packets for moving nodes incurs added delay. In some situations the connectivity loss and the subsequent communication delay are unacceptable. Furthermore, the structure of tactical networks may make it impractical to deploy systems that require dedicated infrastructure to support handover. A relevant example is a tactical telemetry scenario where an unmanned vehicle exploring an area streams video feedback to a command center for real-time support. In such a scenario loss of video data can result in video artifacts and loss of interpretability.

To support the mobility requirements of applications in a tactical environment, there is the need for novel middleware to provide *network migration* functions on top of the traditional communication infrastructure. We define network migration as the ability of a node to change its network attachment point transparently while the connection is maintained and services continue running with minimal disruption. A session handover may be triggered from the physical movement of the device/user in space, causing passage through different networks, or from a network selection strategy that may detect the deterioration of the current network signal or the presence of a higher quality signal from a different network.

This paper presents the network migration support in Mockets, a communication middleware specifically designed to meet the communication needs of tactical environments, and the tactical field advantages provided by this capability. Mockets network migration provides the ability for existing applications to continue operating without interruption irrespective of changes to the attachment point and network addressing, thereby significantly increasing the flexibility of exploiting the best available communication links at any given point in time. Mockets supports flexible communications for applications running on tactical edge networks. Features such as multiple types of flows (e.g. unreliable/reliable and unsequenced/sequenced), prioritization, bandwidth control, message replacement, and a number of other attributes contribute to significantly improve communications performance for tactical applications.

The main objective of the Mockets network migration support is for the handover to be *seamless*, i.e., transparent to the application. No action is required from the application to trigger a handover or to take advantage of the best available communication link. Seamless also refers to the continuous, unbroken service that Mockets aims to provide to the application despite network migrations so that the user does not perceive performance degradation.

Mockets automatically selects the best network attachment point at any given time leveraging on decision strategies based on predefined or user-provided policies, e.g. always select a

WiFi link over a 3G link, broadband over narrowband, etc. Some strategies take advantage of the presence of multiple network interfaces to connect to different networks at the same time and test various parameters of the network, e.g., bandwidth and round trip time. These strategies can take advantage of the Mockets monitoring and statistics collection features. Other strategies measure low-level metrics such as RSSI levels. Our goal is to use best network selection strategies to provide the quality of service desired by the application/user by performing session handovers when a higher quality network is available.

We evaluated the performance of the Mockets network migration through a set of connection delay tests. We measured the time to establish a connection with support for migration and the time to perform a Mockets reconnection operation. We found that while the time for connection establishment suffers from the setup of the security architecture for peer authentication, the time to perform the Mockets reconnection is minimal, an average of 17 ms. In the best case scenario the reconnection time is the only downtime for the application. A qualitative analysis also highlights that Mockets features a reliable service more resilient than TCP to the temporary loss of connectivity caused by a session handover. Another benefit of the Mockets network migration is scalability due to the absence of an external infrastructure typical of other handover protocols (e.g. Mobile IP).

The rest of the paper is as follows. Section II discusses handover strategies proposed in the literature. Section III presents a tactical scenario that could benefit significantly from Mockets session handover. Section IV provides an overview of the Mockets framework and its main features and capabilities. Section V focuses on the network migration support of Mockets, describing the implementation and characteristics of the session handover in Mockets. Section VI discusses the best network selection strategies we implemented. Section VII presents a performance evaluation of our implementation with experimental results and a qualitative analysis of the advantages provided by Mockets. Section VIII provides conclusive remarks and discusses future work.

II. RELATED WORK

The standard for WLAN IEEE 802.11 allows handover between overlapping WLANs at the link layer but [1] measures the handover latency and states that it can be significant and subject to large variations. Mobile IP [2] is probably the most known migration protocol but it requires an external architecture and does not provide fast handovers. Handover strategies have also been added to the Session Initiation Protocol (SIP) [3] but also in this case the handover has been measured to be long and packets would be lost during the handover.

The handover strategies mentioned are inadequate in a tactical environment where an external infrastructure to support handovers may not be feasible and where a short handover time is required because an interruption of service may not be acceptable.

Most handover protocols are developed for mobile scenarios while Mockets also perform handovers to offer the best quality of service. An approach that realizes session handover at the application level to offer quality of service to IPTV streams was proposed in [4]. That approach may lead to adequate performance but does not target a tactical environment because the handover strategy is based on congestion levels within the networks and is not a good strategy for our scope.

III. A SEAMLESS HANDOVER SCENARIO

Considered herein is a scenario in which tactical environments are scattered with wireless networks ranging from fixed installations to base stations mounted on HMMVWs, UAVs, and other mobile platforms. The mobility of the radios demands a communication protocol that supports network migrations seamlessly. At the same time, a mechanism of best network selection and transparent, efficient handover could increase the link performance by allowing the device to take advantage of the presence of overlapping networks with different levels of quality of service, data rates or bandwidth constraints. In the following paragraphs we present a tactical telemetry scenario where the ability to perform network migration is crucial but existing handover protocols would cause an intolerable disruption of service.

The Test and Evaluation and Science and Technology (T&E/S&T) Program Office is tasked with developing technologies related to the test and evaluation of new military capabilities for the DoD [5]. Of particular interest is the testing of Unmanned and Autonomous Systems (UAS) in the ground environment where wireless communications are dynamic (i.e. time-varying) and challenging (i.e. multipath, fading, etc). The Unmanned Systems Integrated Roadmap outlines an evolving role of unmanned ground systems in support of the warfighter to which system testing figures as a key component in assessing technology readiness [6]. Before deployment, these systems will need to be tested to verify and validate the utility and capability of each system as well as the interaction between systems. The possible limited communications capability of individual systems during UAS testing (UAST) requires that emphasis be placed on reliable range telemetry for ground based systems at the Major Range and Test Facility Bases (MRTFB). Reliable range telemetry hinges on uninterrupted network service across multiple available radio access networks (RAN).

Small unmanned ground vehicles (S-UGV) and unattended ground sensors (UGS) have limited communication capabilities due to the nature of the tactical communications built into currently planned systems. These intrinsic communication capabilities make it difficult to test and evaluate systems in real-time from a common test command center where multiple systems' performance metrics are being observed. By outfitting these systems with dedicated range telemetry systems, the UAS test articles are able to communicate with a much more capable telemetry network system that covers a large geographic test area. It is envisioned that test ranges will be outfitted with high-speed wireless base-stations (e.g. IEEE 802.16/WiMAX) to cover wide-open outdoor areas as well as smaller footprint wireless base-stations (e.g. IEEE 802.11/Wi-Fi) placed in specific areas to cover the interior of buildings and tunnels/caves. These edge networks will be supported by highcapacity, reliable backhaul networks enabling the MRTFBs to conduct tests of multiple deployed systems and scenarios in a more efficient way and with better test situational awareness.

In one such scenario, an unmanned ground vehicle/robot is released within an urban area to act as an advanced scout. An operator located in a HMMVW controls the robot to the entrance of a building whereby it is then set into an autonomous mode to perform reconnaissance within the building. Meanwhile, the UAST command center is continuously monitoring the position and status of the robot to verify the transition to autonomous mode, assess range safety, and to coordinate specific events during the test (e.g. initiate test conditions, simulate sensory data, etc.). As the robot enters the building, the test area radio access link (i.e. WiMAX) may begin to fade and the range telemetry device may have to look for another network to connect to. The robot may also have to begin relaying video or imagery of its activity and may require additional bandwidth not available from the wide area network. If the building on the test site is outfitted with a short range network (i.e. Wi-Fi) that is capable of handling the UAS telemetry data feed then the range telemetry device can switch to this new network in order to continue sending its telemetry to the test command center. Under existing IP protocols, the session migration from the more capable wireless wide area network to the wireless local area network would cause an interruption in the real-time telemetry feed required to support real-time UAST activities.

IV. The Mockets Middleware

The Mockets Communication Middleware was developed to provide advanced communication capabilities and high performance in the tactical environment [7]. Mockets runs at the application level, sending and receiving messages over UDP. Mockets offers several delivery services that applications can choose to better serve their needs. The delivery of messages can be performed reliably or unreliably and messages can be delivered in sequence or not.

Most of the communication parameters in Mockets are customizable upon creation of a Mocket or while the Mocket is running. The customization upon creation can help to provide the best performance by adapting the framework to the network where the application is deployed regardless of different environments. Among the customizable parameters there are: MTU (Maximum Transmission Unit), pending packet queue size, keep-alive timeout, initial assumed RTT (Round-Trip Time), window size, SACK (Selective Acknowledgement) transmission timeout. Other parameters such as maximum lifetime of messages, enqueue timeout and message priority, can be set on a per message basis.

Mockets also offers a statistics collection feature and a component to monitor the network state [8]. These allow the developer to design applications that take advantage of the information collected to adjust the parameters of the connection to meet the desired level of quality of service.

A number of other capabilities of Mockets can be used to tune the transmission and obtain high performance in a tactical environment. Prioritization and bandwidth control can be combined for QoS. By exploiting the message tagging capability to mark messages belonging to different types of flows, cancellations and replacements can be performed. Cancellations could be used to remove messages from less important data flows when the communication channel is experiencing a temporary lower bandwidth. Replacements could be used to send only the most recent update in a specific flow by replacing older messages waiting in the queues with newer messages.

Tactical networks are moving toward portable wireless and cellular infrastructures with users expecting to be able to roam across these multiple networks in a transparent manner. Mockets was designed to support mobile service sessions. Two aspects of mobility are implemented: *session migration* to different nodes and *network migration* by dynamically changing the network attachment point [9]. Endpoint migration to a different node lets the application suspend the Mockets connection, retrieve the state of the endpoint, and send it to a different node that will use it to resume the connection and continue the communication with the peer. This is useful to change the device the services are running on without losing the open sessions. By providing support for dynamic change of the network attachment point, Mockets allows the preservation of end-to-end connectivity in spite of node mobility as well as the maintenance of the highest quality of service possible by using the best performing network available. In this paper we focus on this last aspect of the Mockets mobility: seamless network migration, its details will be further discussed in the next paragraphs.

V. Session Handover in Mockets

Handover protocols typically aim to maintain connectivity while a device moves into and out of wireless networks. While this is one of the aspects the Mockets network migration aims to cover, it also tries to maximize the performance of the communication by taking advantage of the network with the best performance available. The information gathered by Mockets, through its statistics collection and network monitoring, can be fed to handover agents that use network selection strategies to decide to request a handover to a new network attachment point. When a handover is requested, an IP address in the new network needs to be acquired and all the services using Mockets need to be re-bound using the Mockets reconnect function.

A. Handover Conditions

We have identified three situations where session handover may occur and Mockets network migration could improve the performance of the communication. The three situations are depicted in Figure 1.

In the first scenario, more than one network is available and the device is equipped with more than one network interface. In this situation, running services can keep using the current network while Mockets concurrently connects to another network over a second interface and analyzes the performance of both networks. In this situation network selection strategies can provide the highest quality of service to the application.

A second situation is when the device has a single network interface. Even if more than one network is present only one connection at a time is allowed. In this situation, network selection strategies may be based on the measurement of the RSSI of the networks and lead to a network migration request.

In the third situation, networks do not overlap in space, a single network is available at any given time. A movement may produce a connection drop before a new network becomes available. This situation does not require particular handoff strategies and is the one where the Mockets handoff presents the most benefit – no failure from the application perspective, while other communication protocols would drop the connection.

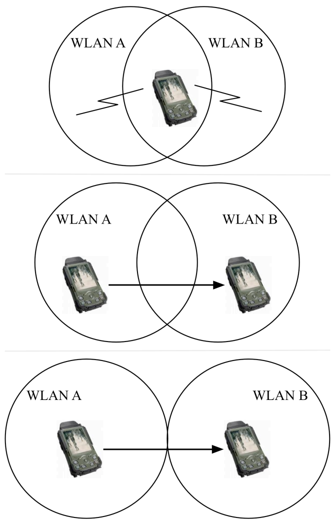

Figure 1. Handover scenarios: 1- Multiple overlapping networks and multiple network interfaces. 2- Multiple overlapping networks and single network interface. 3- Single available network at any given time.

B. *The Reconnect Function*

Once a handover is requested the reconnect function of Mockets is called in order to rebind the active connections to the new network attachment point.

The reconnect function sends a *reconnect message* to the peer node and waits for a *reconnect ACK*. This *reconnect message* contains data to allow the peer to authenticate the source of the message and the new IP and port where the Mockets communication should be rebound. When the peer receives a *reconnect message*, it tries to authenticate the source of the message as the communication peer. If the authentication succeeds, it proceeds by changing the peer IP and port at the Mockets level and sends an acknowledgement in the form of a *reconnect ACK* message to the new location of the peer. Regular communication can resume when the acknowledgement reaches the migrating peer waiting for the *reconnect ACK*.

The wireless networks available in tactical environments may be unreliable and suffer some level of packet loss. Our approach takes into account packet loss and implements a retransmission mechanism to ensure a successful handover. If the *reconnect ACK* does not reach the migrating node before a timeout expires, a new *reconnect message* is sent. *Reconnect messages* will be retransmitted until a *reconnect ACK* is received unless the migrating node waiting for the acknowledgement receives a regular communication message from the peer, in this case it can safely assume that the *reconnect ACK* was lost but the *reconnect message* reached the peer and the connection has been successfully rebound and hence the communication can continue. Furthermore, when a node receives a *reconnect message* from the IP address and port it is already connected to, it assumes the *reconnect ACK* was lost and it sends it again without any additional processing.

C. *Security Architecture for Peer Authentication*

Security could represent a critical aspect in network migrations. During a communication between node A and node B a malicious node could intercept and eliminate the packets from A to B and impersonate node A, claiming a session handover to a new network attachment point.

We implemented a security architecture that consists of a combination of symmetric and asymmetric key encryption algorithms and allows the migrating node to authenticate with the peer after a session handover. During connection setup, the peers agree on a connection UUID and a secret key (KS in Figure 2) using a public key cryptographic algorithm for the exchange. When a session handover is performed the reconnection message includes the connection UUID, new IP address and port, all encrypted with the secret key. When a node receives the reconnection message it can decrypt it and check if the UUID matches the original one and if IP address and port match the source address and port of the UDP message. The messages exchanged to set up the security architecture and to perform authentication at reconnection are shown in Figure 2.

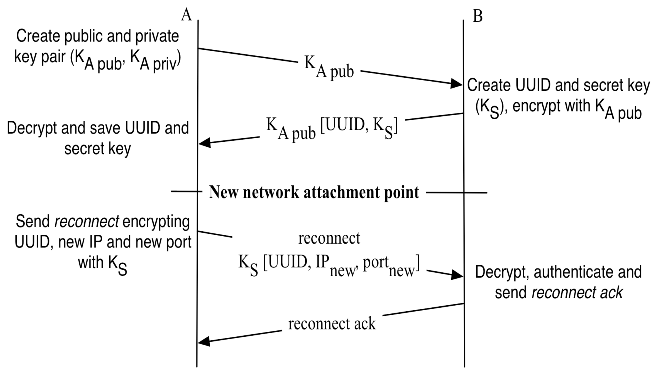
A
B
Create public and private key pair ($K_{A\ pub}$, $K_{A\ priv}$)
$K_{A\ pub}$
Create UUID and secret key (K_S), encrypt with $K_{A\ pub}$
Decrypt and save UUID and secret key
$K_{A\ pub}$ [UUID, K_S]
New network attachment point
Send *reconnect* encrypting UUID, new IP and new port with K_S
reconnect
K_S [UUID, IP_{new}, $port_{new}$]
Decrypt, authenticate and send *reconnect ack*
reconnect ack

Figure 2. Security architecture implemented to perform authentication during session handover.

Note that the security mechanisms realized in Mockets to prevent hijacking of connections is not a replacement for the security mechanisms typically employed in tactical radios. We assume that applications using Mockets normally execute on the "red" side of a tactical radio, with either the radio or a HAIPE (High Assurance Internet Protocol Encryptor) node handling the necessary encryption. The connection migration mechanism is agnostic to the underlying cryptographic security of the radio network.

D. *Seamless Network Migration*

The network migration is seamless to the application. A node is equipped, at the level of Mockets, with the strategies to decide whether to perform a handover and the ability to carry out the task without the intervention of the application. The application is unaware of the handover process in progress, should not perceive a service disruption during the handover, and is able to transparently take advantage of the best available network at any given time.

VI. Network Selection Strategies

Mockets implements an adaptive system that selects the best network connection among the available ones. The framework then dynamically and automatically rebinds the endpoints of all the open sessions to the selected network attachment point and network interfaces.

The best network selection system requires explicit support from network state monitoring mechanisms that provide accurate and timely information to perform effective decision making. Mockets continuously monitors the status of all network layer addresses and network interfaces on the device. Both passive, e.g., used bandwidth and round-trip time, and active, e.g., available bandwidth estimation via packet probing, measurements at the network level are supported. In addition, Mockets tracks several low-level metrics, such as RSSI level of wireless network interfaces or GPS location, where available.

The Mockets best network selection system implements both reactive and proactive handover decision algorithms. Each handover decision algorithms operates on a set of metrics of interest, e.g., available bandwidth and round-trip time. Reactive algorithms simply initiate the handover procedure when the value of those metrics goes beyond a given threshold. To prevent multiple handoffs between networks with similar characteristics, Mockets adopts hysteresis-corrected decision functions. Proactive strategies leverage predictions of future values for the metrics of interest often performed by means of Exponentially Weighted Moving Average (EWMA) forecast algorithms.

The handover decision algorithms implemented in Mockets are described in terms of KAoS policies [10]. This allows users to fine-tune the selection process according to their preferences, e.g., to consider economic costs of traffic on the different interfaces.

VII. Performance Evaluation

A number of performance tests have been conducted to demonstrate the efficiency of our implementation. We measured the time to establish a new connection when support for migration is requested and the time to complete a reconnect operation. The results of the experiments show low downtime to perform the session migration to a different network . They also show that the security infrastructure causes some overhead in the initial connection setup.

The tests have been performed on two laptop machines, the first acting as a Mocket server and the second acting as a Mocket client, connected through wireless 802.11 interfaces. The machines run Ubuntu and feature a 2.00 GHz process with 3GB of memory and a 2.40 GHz processor with 512 MB of memory, respectively.

Our approach to session migration presents a very low reconnection time. We measured 17 ms on average to perform a reconnect operation. The total downtime for an application performing a handover in Mockets is given by the *reconnection time*, plus the time to acquire a new IP address in the destination network, which we define as *IP address acquisition time*. Depending on the migration scenario, the time for the IP address acquisition could represent zero or up to few seconds of downtime for the application.

The first scenario is illustrated in Figure 1 when more than one network interface is present and more than one network is available. While one interface is used for the active connections a second interface could be used to scan the available networks to identify one that provides better performance than the one currently in use, acquire an IP address in the selected new network and trigger a handover. This is our best case scenario where the only downtime for the application is the few milliseconds of the reconnection time. In the second and third scenarios pictured in Figure 1, the IP address acquisition time needs to be taken into account as part of the downtime of the application. The authors in [11] measure the time to perform a handover at the link layer level, where a node registers with a different base station from the one it was connected to less than 10 ms. The additional time needed to be assigned an IP address is strictly dependent on the specific network we are connecting to.

The tests we performed also show that establishing a new Mockets connection without support for session migration takes 12 ms on average. When support for session migration is requested the average measured time to connect was 114 ms. The connection establishment takes into account the cryptographic operations as well as the data exchange between the client and the server. The largest component of the connection establishment time is the time taken to generate a new public/private key pair, which is CPU intensive and highly dependent on the system performance and the system load. Figure 3 shows the results of several runs measuring the time to perform the connection establishment with support for network migration and the time to reconnect. The time to establish a new connection shows some variations in different runs. This is due to the creation of the key pair, which is not a constant-time operation.

When the session migration is performed in networks with a certain level of packet loss, the performance of the handover may degrade. If the reconnect packet or the reconnect ACK packet are lost, they need to be retransmitted requiring more time to successfully reconnect to the peer and hence a longer downtime will affect the application. However network selection strategies aim to choose the link with higher quality. Therefore, in the presence of multiple networks, handover strategies take carefully into account packet loss rates among other parameters since losses may adversely affect the migration.

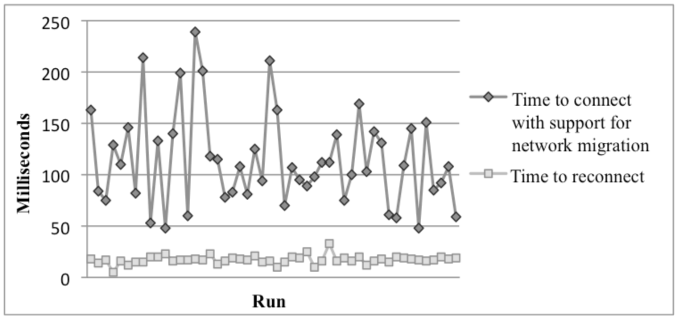

Figure 3. Several runs that show the time to perform the establishment of a new connection with support for network migration and the time for a reconnect operation.

The Mockets network migration approach not only allows a fast handover, it can also take advantage of the peculiar design of Mockets to overcome the short period of lack of connectivity without additional overhead. The authors in [12] present several tests that measure the disruption an application using TCP runs into when performing a session handover over Mobile IP. The handover causes a service disruption longer than 10 seconds to the application. This downtime is due in part to the actual time required to perform a session migration using Mobile IP that was measured to be between 2.77 and 5.91 seconds, depending on the movement detection mechanism used. But it is also due to the disruption that a period of missing connectivity creates on TCP. TCP reacts to the expiring timeouts of packets during handovers with exponential backoff. Once the handover is complete and the connection is restored, packets start to be acknowledged again while the slow-start algorithm implemented in TCP (as a congestion avoidance mechanism) gradually increases the sending rate. TCP responds to the period of missing connectivity with drastic measures. Mockets on the other hand makes available the same reliable service TCP provides but is resilient to the loss of packets since it is designed for tactical environments and wireless networks where the loss of a packet or a delay does not necessarily imply network congestion.

An additional benefit of the Mockets approach to network migration is scalability. Using Mockets we can perform network migrations without an external architecture or the support of other entities in the network, such as a Home Agent and Foreign Agent in the case of Mobile IP. Regardless of the number of nodes performing migrations and the number of migrations performed by each node, the only overhead added to the network is represented by the reconnect and reconnect ACK messages exchanged to perform the handover.

VIII. Conclusions and Future Work

Mockets provides tactical network applications with a session migration function that enables seamless handovers between different network attachment points. The Mockets network selection function allows exploiting the best available link according to network conditions and user preferences. Our experimental results show that the time required to migrate a session endpoint in Mockets is usually less then 20 milliseconds. The middleware performance is therefore adequate for most applications.

We are currently evaluating other strategies to perform network selection and are studying how different strategies perform in different network scenarios. The scenario that a strategy is tested on may bias the good performance obtained from one strategy. Using Mockets statistics collection and monitoring features, we could deduce the particular scenario the communication is in and apply the best strategy for that particular situation.

Extending the idea of the session handover that takes advantage of a device with multiple network interfaces, we are implementing a trunking mechanism that would allow a Mockets connection to exploit two networks at the same time instead of just migrating to a higher quality network. By trunking the multiple available networks, a Mockets endpoint could send and receive messages on multiple links in order to improve overall bandwidth, or use different networks for traffic with different priorities.

Acknowledgment

This work is supported in part by the U.S. Army Research Laboratory under Cooperative Agreement W911NF-04-2-0013 and by the Air Force Research Laboratory under Cooperative Agreement FA8750-06-2-0064.

References

[1] A. Mishra, M. Shin, and W. Arbaugh, “An empirical analysis of the IEEE 802.11 MAC layer handoff process”, ACM SIGCOMM 2003.

[2] A. Stephane and A. H. Aghvami, “Fast handover schemes for future wireless IP networks: a proposal and analysis”, IEEE Vehicular Technology Conference 2001.

[3] E. Wedlund and H. Schulzrinne, “Mobility support using SIP”, ACM WoWMoM 1999.

[4] G. Cunningham, P. Perry, J. Murphy, L. Murphy, “Seamless handover of IPTV streams in a wireless LAN network”, IEEE Transactions on Broadcasting Volume 55 Issue 4 Dec 2009.

[5] Test & Evaluation/Science & Technology Program Office. [Online]. http://www.acq.osd.mil/trmc/jipp/te-st/

[6] Office of the Secretary of Defense. Unmanned Systems Integrated Roadmap FY2009-2034 [Online]. http://www.acq.osd.mil/uas/docs/UMSIntegratedRoadmap2009.pdf

[7] M. Tortonesi, C. Stefanelli, N. Suri, M. Arguedas, M. Breedy, “Mockets: a novel message-oriented communication middleware for wireless internet”, WINSYS 2006.

[8] C. Stefanelli, M. Tortonesi, M. Carvalho, N. Suri, “Network conditions monitoring in the Mockets communications framework”, MILCOM 2007.

[9] C. Stefanelli, M. Tortonesi, E. Benvegnú, N. Suri, “Session mobility in the Mockets communication middleware”, ISCC 2008.

[10] N. Suri, M. Carvalho, J. Lott, M. Tortonesi, J. Bradshaw, M. Arguedas, M. Breedy, "Policy-based bandwidth management for tactical networks with the Agile Computing Middleware”, MILCOM 2006.

[11] R. Càceres, V. N. Padmanabhan, “Fast and scalable wireless handoffs in support of mobile internet audio”. Mobile Networks and Applications. Vol 3, Issue 4 (1998).

[12] N. A. Fikouras, K. El Malki, S. R. Cvetkovic, C. Smythe, “Performance of TCP and UDP during Mobile IP handoffs in single-agent subnetworks”, IEEE WCNC’99.