AD HOC MOBILITY PROTOCOL SUITE (AMPS) FOR JTRS RADIOS

Thanh Cheng, Proveen Gurung, John Lee, Sumit Khurana, Anthony Mcauley, Sunil Samtani, Larry Wong, Ken Young (Telcordia Technologies, Inc., Morristown, NJ, USA; {tcheng, pgurung, jolee, sumit, mcauley, ssamtani, larryw, kcy}@research.telcordia.com), Michael Bereschinsky, Charles Graff (CECOM, Fort Monmouth, NJ, USA; michael.bereschinsky@mail1.monmouth.army.mil, charles.graff@us.army.mil)

ABSTRACT

This paper describes several technologies to provide rapidly deployable, secure, robust IP communications among tactical mobile ad hoc nodes supporting JTRS radios. This Ad-Hoc Mobility Protocol Suite (AMPS) provides six basic functions: a) automatic configuration and reconfiguration, b) mobility management, c) routing, d) reliability, e) security and f) visualization. In addition, we also include Differentiated Services/Bandwidth Broker (DS/BB) approach for service differentiation and admission control to provide appropriate Quality of Service (QoS) guarantees to mission-critical and real-time applications. This paper describes technologies to support advanced networking on JTRS radios.

1. INTRODUCTION

Future battlefield networks, defined by the Objective Force, present several critical challenges, which do not greatly affect commercial networking. The networks must be self-configuring and self-maintaining so that they can be rapidly deployed and quickly reorganized when needed. All elements must be highly mobile and survivable, including routers that make up the communications infrastructure. Entire networks may move, or nodes and links may be added to an existing network. As networks merge and split, dynamic border routers must adjust route distributions dynamically. Available bandwidth must be used efficiently, and information must be disseminated rapidly, so that the right information is available at the right place and time.

Future Combat Systems pose more stringent requirements for security and reliability than commercial networks. Links are susceptible to failures because of mobility of nodes, or loss of connectivity due to unreliability of wireless links or terrain effects. Nodes may fail because they have been destroyed during combat. Such systems need rapid re-configuration to activate key network functionality in event of failures. Also, traditional mobility and location management functions in the network cannot rely on the existence of fixed home agents and must maintain persistent sessions through the use of smart software in individual nodes. The most prevalent services of a Future Combat Systems (FCS) environment are mission critical real-time multimedia multi-party collaborative applications that allow commanders and or war-fighters to receive "live" view of the battlefield and adapt their strategy accordingly, and communicate it to the war-fighters, robots, and smart weapons in a timely manner.

The new war fighting paradigms, such as Joint Vision 2010, emphasize mobile, flexible networks that automatically adapt to the war fighter's needs. These paradigms, as expressed in the Joint Tactical Radio System (JTRS) ORD, require mobile networking capabilities significantly beyond what is possible with currently fielded technology. JTRS networking protocols must support a variety of services, including automatic neighbor and link quality discovery, automatic network reconfiguration, quality-of-service guarantees, precedence and priority marking, and the automatic routing and relaying of traffic. In addition, these services must be supplied with efficient protocols that minimize overhead. The development and implementation of mobile networking protocols is a technical challenge to the JTRS effort. We have defined an ad hoc mobility protocol suite (AMPS) for FCS networks consisting of network configuration, routing, mobility management, QOS (DiffServ/Bandwidth Broker) and security protocols. These protocols are major enhancements to the standard TCP/IP protocol suite. We are porting the AMPS software on a JTRS radio platform to be Software Communications Architecture (SCA) compliant.

The JTRS SCA specification is primarily geared to the logical structure of a software radio and has not focused to any great extent on support needed for advanced network services. One makes the assumption that when using a JTRS stack, services such as a TCP/IP will be provided by a Commercial Off The Shelf (COTS) vanilla stack, for example, from VxWorks, Spider Software, Trillium, LynxOS and others. Such a stack is typically incorporated in the JTRS kernel and can be called as a *Network Resource* device. Such stacks are adequate for existing services but do not contribute in any meaningful way to the deployment of advanced services. As an example, DiffServ requires the manipulation of queue structures that are very deep within the kernel to garner adequate performance across a network. If JTRS SCA is followed, DiffServ and other services such as packet replication for multicast and forwarding need to be brought up to the user application level and then attached to the stack. Since nearly all the networking and link layers are implemented in the user-space as per the JTRS SCA specification, and the API between these layers is defined and implemented using CORBA, multiple traversals through the CORBA bus will be required to send a packet from the application to the physical device. Also, advanced services such as IP tunneling and mangling of IP packets to support mobility will require shim in the stack implementations and to keep the stack layers vendor neutral, this will entail more traversals through the CORBA bus.

This paper describes ad hoc mobility technologies for supporting key networking services. We recommend that these technologies should be included in the JTRS network stack. In this paper, we briefly discuss our design and implementation as we port the AMPS technologies within the Rockwell Collins Wideband, wireless Networking Engine (WNE) Joint Tactical Radio System (JTRS) Radio (WJR).

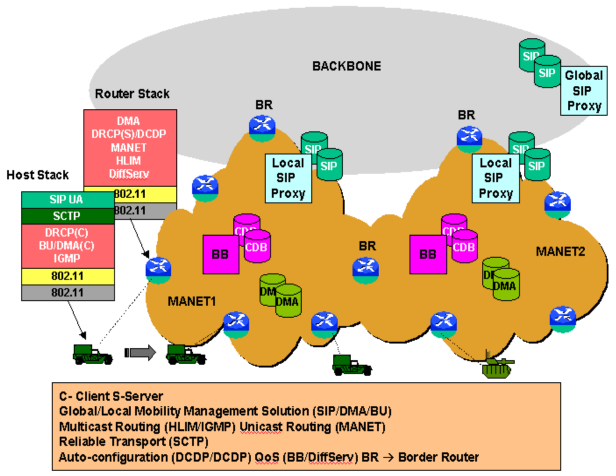

Figure 1. Example Mobility Architecture

2. AD HOC MOBILITY PROTOCOL SUITE

Our experience with army networks lead us to strongly believe that there is no single-technology solution, for example, ad hoc routing, to ad hoc mobility, but rather the best ad hoc mobility and QoS solution consists of the integration of several technologies, spanning several layers in the protocol stack.

As shown in the example architecture in Figure 1, the integrated solution also looks at promising research ideas in development to ensure the architecture can easily accommodate them. To arrive at the best overall solution, we have used the best-of-breed technologies available and integrated them into a robust and adaptable solution focused on Future Combat Systems (FCS) objectives. The result is a complete system for:

a) Configuration. Automatic configuration and reconfiguration of all nodes in a network using the auto-configuration suite: Dynamic and Rapid Configuration Protocol (DRCP) [1], Dynamic Configuration Distribution Protocol (DCDP), configuration and status reporting protocol (YAP) and an Adaptive Configuration Manager (ACM). Future Combat Systems must rapidly configure themselves without human intervention; moreover they must rapidly reconfigure themselves to generate new formations as conditions change. DRCP configures layer-3 information at hosts and routers to support inter-working and DCDP automatically distributes address-pools and other configuration information to every subnet in a domain in a scalable manner. The Adaptive Configuration Manager and status reporting capabilities allow nodes to be re-configured in event of server failure.

b) Integrated Mobility Management. AMPS supports combined personal, session and terminal location management through an enhanced Session Initiation Protocol (SIP) [2]. Dynamic Mobility Agent (DMA) [3, 4] provides local points-of-attachment to reduce mobility management overhead and provide fast handoffs. Dynamic mobility agents allow organizing networks into domains to group nodes for reducing the number of binding updates and global registrations. Since ad hoc networks are highly dynamic in nature, our mobility management scheme does not assume the existence of fixed home agents/networks and adaptable applications that adapt to change in IP addresses. Our scheme builds smart end-hosts that react to change in IP addresses by correcting the destination addresses of the outgoing packets.

c) Routing. We use conventional and MANET routing protocols tied together by Dynamic Border Routers (DBR). Although the FCS does not assume a specific hierarchy, the DBR will play an important role in inter-working of FCS Cells. The DBR is auto-configured as networks merge and split. The appropriate size of routing domains is determined using modeling and simulation. Efficient scoped multicast is provided through the Hierarchical Level-based IP Multicast (HLIM) [5] solution. Scopes are flexible in nature and can be used to create a logical hierarchy in the FCS at a point in time with no underlying physical hierarchy in place. HLIM scoping limits signaling traffic and avoids traffic concentration.

d) Reliable Transport. AMPS supports Stream Control Transport Protocol (SCTP) [6] for mobile wireless communications for MOSAIC. SCTP offers some unique features including: a) partial order in sequencing packets, b) message orientation (removes need for applications to add explicit record marking), c) resistance to denial of service attacks, d) multi-homing (e.g., for network fail-over, load balancing), e) dynamic endpoints switchover (e.g., to support mobility), f) selective acknowledgements, g) support for Explicit Congestion Notification (ECN), which can be exploited to differentiate between packet loss and congestion.

e) Security. AMPS combine Internet Key Exchange (IKE) and IP Security (IPsec) [7] protocols to provide a comprehensive solution for user authentication, key exchange, message integrity, confidentiality and replay protection. We support COTS implementation of IPSec and IKE adapted to provide end-to-end security for all our data and signaling transfers (e.g., protect routing update traffic).

f) Protocol Visualization. We support two GUIs for protocol visualization: 1) a Network Graphical User Interface (N-Gui) that displays the network configuration of any node in the network and 2) a. Local Graphical User Interface (L-Gui) that displays local status and capabilities of any node.

g) Service Differentiation and Admission Control. Bandwidth Brokers (BB) [8], working with Differentiated Services routers, provides integrated resource management and admission control functionality in dynamically allocating network resources to assure IP quality of service in mobile ad hoc networks. Our scheme allows us to exploit layer 2 service differentiation, where available.

h) Network Services. In addition to the networking technologies, AMPS supports network services for military needs such as Voice over IP using SIP, multimedia-conferencing and collaborative whiteboard. Figure 2 shows a network protocol stack diagram of the current MOSAIC AMPS Stack.

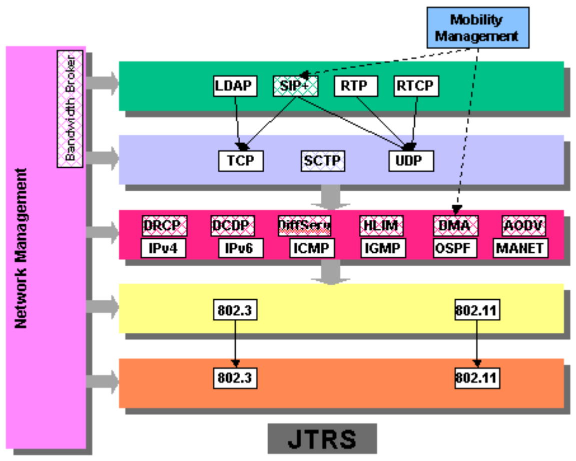

Figure 2. MOSAIC AMPS Stack

3. AMPS JTRS IMPLEMENTATION

The MOSAIC ATD program integrates the communications technologies that address the objectives of the Future Combat Systems (FCS). A key component of the program is to integrate mobility mechanisms of Ad-Hoc Mobility Protocol Suite (AMPS) within the Rockwell Collins Wideband, wireless Networking Engine (WNE) Joint Tactical Radio System (JTRS) Radio (WJR).

The WJR is in compliance with the JTRS Software Communication Architecture (SCA) Specification (SCAS) Version 2.0. A component of this architecture for the WJR is Operating Environment (OE), which consists of the Lynx 3.1.1 Operating System (OS) and a separate external TCP/IP communication stack that resides in the application space of the OE. The separate communication stack is required by SCAS Version 2.0 Security Supplement to prevent exposure of the internal IP stack to the external world. The TCP/IP stack is from Spider, Inc., ported to the Lynx 3.1.1 application space to meet the requirements for the external stack. The WJR contains a red/black separation with emulated encryption/decryption from the red to black and black to red sides, respectively. The high-level architecture for the WJR port is shown in Figure 3.

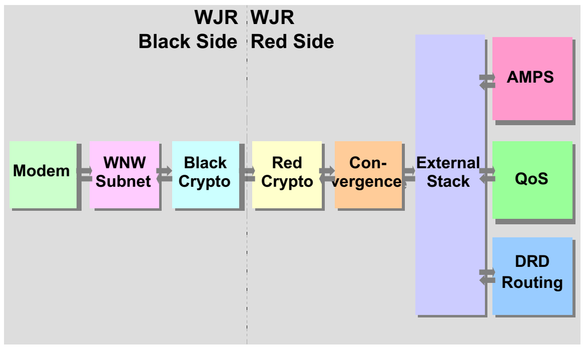

Figure 3. WJR Baseline Architecture

We have developed the original AMPS software using the Linux 2.4.7 OS. The AMPS software interfaces to this OS both for lower-level kernel functions, such as timers and semaphores, as well as to the Linux 2.4.7 TCP/IP stack for external communication. To move the AMPS software to the WJR, the AMPS software must be ported to run exclusively in the OE application space and interface to the Lynx 3.1.1 OS for kernel operations (low-level primitives) and to the WJR Spider stack (external stack) for subnet communication (inter-box). In addition, the AMPS resource must be provided with the Common

Object Request Broker Architecture (CORBA) Application Programming interfaces (compliant to SCAS Ver. 2.0) and the JTRS Core Framework (CF) interfaces to allow for startup, resource connection, and interoperability with other JTRS resources.

4. ADVANCED NETWORK SERVICES

This Section describes the advanced networking services such as tunneling, mangling and multicast to support scalable ad hoc mobility and how the JTRS TCP/IP stack needs to be modified to support these services.

IP Packet Mangling

Our Integrated Mobility management solution uses an application variation of the network rebind [9], the SIP re-invite. The mobile hosts sends the SIP re-invite to tell the corresponding host that it has moved to a new IP address. In other SIP based mobility solutions [10], the SIP User Agent (SIP UA) informs applications to send packets to the new IP address. By putting the rebind in the application, it a) eliminates any need to do encapsulation (with its 20 byte header), b) avoiding triangular routing, c) does not rely on home networks. The drawback, however, is that SIP requires changes to the applications: in particular, the applications must be able to dynamically bind to the new IP addresses. To overcome this problem of application transparency, our solution allows applications to use PIP (Permanent IP Address) to address correspondent hosts. These addresses are then internally mangled and de-mangled to globally routable addresses at the hosts as shown in Figure 4. The SIP re-invite maintains the binding cache when the host moves from one domain to the other. Applications use PIP to transport data and are not aware of the care-off addresses, thus, even when care-off IP addresses change, transport layer sessions will be maintained without requiring modifications to existing applications.

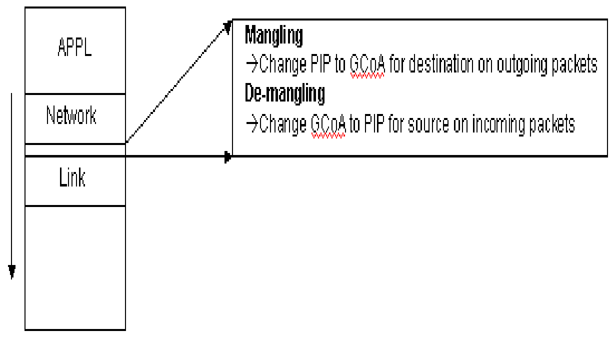
APPL
Network
Link
Mangling
→Change PIP to GCoA for destination on outgoing packets
De-mangling
→Change GCoA to PIP for source on incoming packets

Figure 4. Persistent connections using mangling of IP addresses

The following scenario exercises this functionality.

1. Mangler is initialized to intercept packets from the application.
2. Mangler receives the first packet bearing the PIP of the Correspondent Host (CH) as the destination IP address.
3. If Mangler does not have an entry in its Binding Cache, it informs SIP UA that a lookup is required for the PIP.
4. SIP UA initiates a look up to the registration-database via the SIP Proxy.
5. SIP Proxy returns the current care-off address.
6. SIP UA informs the mangler of the current care-off address (COA).
7. Mangler updates its binding cache with the new <MH name, PIP, COA> mapping.
8. Mangler intercepts the next and subsequent packets from/to the application and replaces the source and destination PIP addresses with their corresponding COAs from the binding cache.
9. Mangler times out the entries in the binding cache if no re-invite is received for the CH.
10. In this case, a new SIP lookup for the COA of the CH is initiated.

In the JTRS port, the IP packet mangler would need to reside within the external TCP/IP stack. To enhance system portability, this service could also reside as a separate *Network Resource* outside the external stack with its own API. An API for the mangler must be general enough to set the filter based on the following parameters:
1. Traffic Flow:
IN - filter only on incoming packets.
FORWARDING - filter only on forwarding packets.
OUT - filter only on outgoing packets.
2. IP Packet Header fields:
The API must provide enough granularity to specify a filter on various IP header fields like the protocol (TCP/UDP/ICMP), addresses and ports.

IP-IP Tunneling

Our solution uses an IP-IP tunnel for communication between a Dynamic Mobility Agent (DMA) and mobile nodes within a local domain. When a mobile node first moves into a domain, it obtains a local IP address from the subnet-specific configuration agent. The subnet-configuration agent also assigns a DMA to a mobile node as a part of the configuration. If the assigned DMA is different than the one the mobile node previously had, then the mobile node performs an inter-domain mobility registration; else it performs an intra-domain registration.

In the case of inter-domain registration, the mobile node notifies the DMA of its local address and requests a new global address. In the case of intra-domain registration a mobile node keeps its previously assigned global address, hence the mobile node only notifies the DMA of its new local address. As a result of a registration a bi-directional IP-IP tunnel is setup between the DMA and a mobile node. On the mobile node, routing is setup so that the default route for all the packets points to the tunnel. Similarly routing is setup in on DMA so that the packets destined for a mobile node's global address is routed through the tunnel to the mobile node.

It is fairly simple to construct a delivery mechanism for packets destined to the mobile node. A DMA receives an IP packet transmitted to a mobile node's global care-of address. The DMA then encapsulates the packet (with the mobile node's local care address as the outer destination header) and then forwards it to the mobile node using conventional IP routing within the domain. The mobile node receives this packet, de-encapsulates the outer header and then processes it.

For the reverse path (for packets from the mobile node to a correspondent host), reverse tunneling is used.

A JTRS IP Stack would need to support API's to create IP-IP tunnel. Typically such an API needs to configure the following parameters for an IPIP tunnel:

- Local Address: IP address of one of the local interface
- Remote Address: IP address of a host where the tunnel ends.
- Tunnel Address: IP address assigned to the tunnel interface.

Additionally, the name for the tunnel, for example, tunll, can be specified as a parameter to the API, or can be a return value (i.e. let the system decide the name). Also the IP stack needs to support API's to modify routing tables so that the packets can be routed through the tunnel.

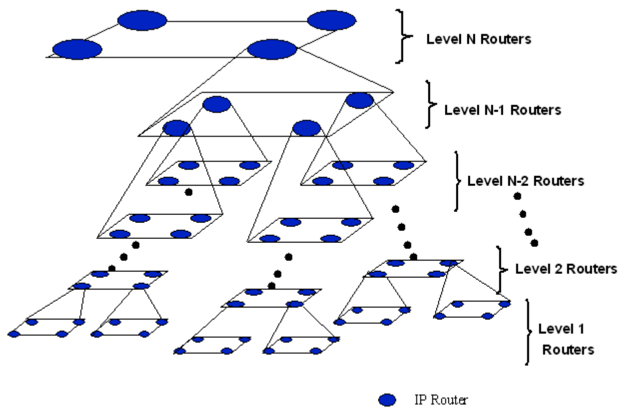

Figure 5. Multicast Hierarchy

IP Multicasting

HLIM (Hierarchical Level-based IP Multicast) protocol is basically a scoped version of a flood-and-prune multicast protocol, such as DVMRP. It was designed and developed to address requirements such as host mobility, network mobility, reliability, survivability, and scalability. HLIM: a) assigns a hierarchical level for each IP router, b) creates scope regions bounded by two hierarchical levels as shown in Figure 5 (one for the highest tier and the other for the lowest tier that a multicast packet for a group can reach), and c) includes this scope information in a multicast address. Each IP router determines whether or not it should discard a multicast packet received by comparing its own level with the scope levels specified in the packet. HLIM's scopes allow:

Flexible Scopes: Scopes can be defined based on functions or roles rather than the crude IP TTL (Time To Live) commonly used in existing multicast schemes. The levels assigned by HLIM can be either logical or physical.

Limit signaling traffic. Unlike existing source-initiated multicast schemes, such as DVMRP and PIM-DM, signaling traffic is limited to a scoped region.

Avoid traffic concentration. Unlike existing receiver-initiated multicast schemes, such as CBT and PIM-SM, there is no hot spot (e.g., core router).

Manageability. A manager can see from the multicast address which links will be affected by a given HLIM multicast sessions.

Table 1 lists the API that a stack needs to support for HLIM protocol. This API details the important HLIM operations.

Table1. API for HLIM

System Call – setsockopt()	Features
setsockopt(igmp_socket, SOL_SOCKET, SO_RCVBUF, (char *)&bufsize, sizeof(bufsize))	To assign Socket Layer buffer size.
setsockopt(igmp_socket, IPPROTO_IP, IP_HDRINCL, (char *)&bool, sizeof(bool))	To build IP header at user space.
setsockopt(igmp_socket, IPPROTO_IP, IP_MULTICAST_TTL, (char *)&ttl, sizeof(ttl))	To set TLL for IP multicast packets.
(setsockopt(igmp_socket, IPPROTO_IP, IP_MULTICAST_LOOP, (char *)&loop, sizeof(loop))	To activate a multicast session in a host to be looped back to the host.
setsockopt(igmp_socket, IPPROTO_IP, IP_MULTICAST_IF, (char *)&adr, sizeof(adr))	To specify out-going interface for multicast packets.
setsockopt(igmp_socket, IPPROTO_IP, IP_ADD_MEMBERSHIP, (char *)&mreq, sizeof(mreq))	To join a multicast group.
setsockopt(igmp_socket, IPPROTO_IP, IP_DROP_MEMBERSHIP, (char *)&mreq, sizeof(mreq))	To leave multicast group.
setsockopt(igmp_socket, IPPROTO_IP, MRT_INIT, (char *)NULL, 0)	To activate Multicast Routing.
(setsockopt(igmp_socket, IPPROTO_IP, MRT_DONE, (char *)NULL, 0)	To stop Multicast Routing.
setsockopt(igmp_socket, IPPROTO_IP, MRT_ADD_VIF, (char *)&vc, sizeof(vc))	To register multicast interface to Multicast Routing.
setsockopt(igmp_socket,	To deregister a

System Call – setsockopt()	Features
IPPROTO_IP, MRT_DEL_VIF, (char *)&vifi, sizeof(vifi))	multicast interface to Multicast Routing.
setsockopt(igmp_socket, IPPROTO_IP, MRT_ADD_MFC, (char *)&mc, sizeof(mc))	To add a multicast forwarding cache into Multicast Routing Table (MRT).
setsockopt(igmp_socket, IPPROTO_IP, MRT_DEL_MFC, (char *)&mc, sizeof(mc))	To delete a multicast forwarding cache from MRT.
getsockopt(igmp_socket, IPPROTO_IP, MRT_VERSION, (char *)&vers, &len)	To get the version of Multicast Routing.

IP Auto-configuration

IP auto-configuration configures and re-configures IP addresses and routing on an IP interface of a host and a router. To be able to do this, the JTRS Network API and the implementation should provide enough flexibility to configure and re-configure IP addresses and routes with minimal or no disruption to active applications on the host/router.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we described ad hoc mobility technologies for supporting key networking services that would add value if included in the JTRS network stack. Also, we discussed our design and implementation for the port of AMPS technologies within the Rockwell Collins Wideband, Wireless Networking Engine (WNE) Joint Tactical Radio System (JTRS) Radio (WJR). We plan to conduct detailed performance evaluation of auto-configuration, IP mobility and multicast services on a JTRS radio and publish a vendor independent API for AMPS network stack.

6. REFERENCES

[1] A. McAuley, K. Manousakis, “Self Configuring Networks,” *4’th ATIRP Conference*, College Park, Maryland, March 2000.

[2] M. Handley et al., “SIP: Session Initiation Protocol”, RFC 2543, March 1999.

[3] K Chakrabarty et al, "Implementation and Performance Evaluation of TeleMIP," *ICC*, May 2001.

[4] S. Das, A. Misra, A. McAuley, A. Dutta, S. Das, “A Generalized Mobility Solution Using a Dynamic Tunneling Agent,” *ICCCD*, December 2000.

[5] J. Lee and M. Cheng, “Hierarchical Level-based IP Multicasting for Tactical Networks,” *IEEE Milcom’99*, November 1999.

[6] R. Stewart et al., “Stream Control Transmission Protocol”, RFC 2960.

[7] S. Kent et al., “Security Architecture for the Internet Protocol,” RFC 2401, November 1998.

[8] K. Kim et al, “A Bandwidth Broker Architecture for VoIP QoS” *ITCOM*, August 2001.

[9] R. Jain et al, “Mobile Internet access and QoS guarantees using Mobile IP and RSVP with Location Registers”, June 1998.

[10] H. Schulzrinne, “SIP Registration,” Internet Draft, http://search.ietf.org/internet-drafts/draft-schulzrinne-sip-register-01.txt, April 2001.