Mobile IP Address Efficiency

Zhen Zhen and Srinivas Sampalli

Original scientific paper

***Abstract* – In future wireless networks, Mobile IP will be widely deployed as a general mobility protocol. Currently, in the protocol each mobile node (MN) should have one public home address to identify itself when it is away from home. Unlike the stationary host, the MN cannot simply use private addresses when NAT (Network Address Translation) is enabled. How to assign public addresses among mobile nodes is important to save the already limited IPv4 addresses. Even though Mobile IPv6 can provide a large address space, when communicating with IPv4 based hosts, the MN still needs to use one public IPv4 address. Protocol translation can map between IPv6 and IPv4 addresses; however, it is a NAT-based approach and breaks end-to-end communications. From a new perspective, we propose an address-sharing mechanism that allows a large number of MNs to share only one IPv4 public address while avoiding most of the drawbacks of NAT.**

***Index terms*: Network Protocols, Wireless, Mobile Computing**

I. INTRODUCTION

The widespread development of wireless networks has created several standards that are designed for different purposes: 3G (The Third Generation) cellular network for the wide area, 802.16 for the metro-area, and 802.11, HiperLAN, etc. for the local area. These networks are not only different in coverage, but also in bandwidth, price, and other features. Future wireless networks are intended to be heterogeneous in nature, taking the advantages of all candidate networks and choosing the best to be connected.

Mobile IP [1] will be an important protocol to support the seamless roaming over heterogeneous wireless networks. In Mobile IP, each mobile node (MN) has a home network. When the MN is away from home, it registers its current location with the home agent (HA). When the correspondent node (CN) needs to communicate with the MN, it sends traffic to the home network. The HA then tunnels the traffic to the foreign agent (FA), which locates in the foreign network and finally delivers the packets to the MN (Fig. 1).

Mobile IP assigns a home address for each mobile node. In order to identify the MN in the foreign network, the home address must be a unique public address. With the fast growth of wireless devices, this will cause a large demand for IPv4 addresses.

Manuscript received November 30, 2005; revised February 24, 2006

Z. Zhen is with the Faculty of Computer Sciene, Dalhousie University, Halifax, Nova Scotia B3H 1W5, Canada (e-mail: zhen@ cs.dal.ca).

S. Sampalli is with the Faculty of Computer Sciene, Dalhousie University, Halifax, Nova Scotia B3H 1W5, Canada (e-mail: srini@cs.dal.ca).

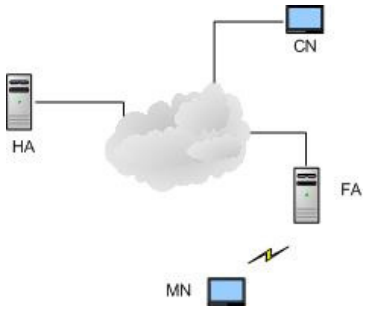

Fig.1. Mobile IP Entities

The immediate solution is to deploy IPv6 [2] and use Mobile IPv6 [3]. However, during the initial stage of the deployment, it is desirable for the IPv6-based MN to communicate with IPv4-based hosts. The interconnection between IPv6 and IPv4 networks can be realized in different ways. Having a dual IP stack on the MN requires one IPv6 address and one IPv4 address at the same time; hence this one-to-one mapping does not save addresses. Translating IPv6 and IPv4 packets at an intermediate router has the same drawbacks as NAT (Network Address Translation). NAT is advantageous in its simplicity - the only change is made on the router and hosts are update-free. However, its manipulation of packet headers causes non-transparency to other protocols, such as IPSec (IP Security) and TLS (Transport Layer Security), which compute security payloads based on the original IP header. If a header is manipulated during the transmission it will be taken as a security breach. Even Mobile IP itself needs extra processing to pass through NAT [15].

The non-transparency also affects applications. If an application includes the address in the payload of the packet, as opposed to the packet header, then the receiving application will read this address, which might be different from the address in the IP header because the IP header is modified during the transmission. When port translations are used in order to share a single public address among a number of hosts, it becomes impossible to call an internal host from outside the network due to the inability of the NAT router to determine the actual destination. Also, the port translation disables internal hosts from running servers, which would require specific port numbers.

With the deployment of Mobile IP or even the IPv6 protocol, the MN has already anticipated changes so that the simplicity of NAT is no longer a tradeoff for its shortcomings. In this paper, we propose an address-sharing mechanism in which one address that has already been assigned to one MN

1845-6421/05/5164

can be re-assigned to another. The principle is that after the packets have actually arrived at the HA, the HA can use, instead of a unique address, the session parameters (namely, the destination address/port and source address/port) obtained from packets to identify the destination MN. The justification is that the CN addresses, the CN ports, and the MN ports in two different sessions cannot all be the same (even if the MN addresses are repetitively reused), if the sessions are initiated from outside the network. In other words, if a session is initiated from a CN, then for different MNs, although they may have the same IP address and even the same port number, the port numbers on the CN must be different. This is based on the fact that if CN initiates the call, then it must not be a server (Only if the CN is a server, then two sessions can be connected to the same port of the CN).

For the sessions initiated from inside the network, the MNs share a statically assigned public address with different port ranges. This is similar to the port translation in NAT except that the address is statically assigned to the MN.

The paper is organized as follows. In Section II, we will review related work in this area. Section III describes the design principle and in the following sections (IV, V, VI) the protocol is described and discussed in detail. In Section VII an analysis using queuing theory is given to show that with one single address the system can serve a large number of mobile hosts. Section VIII concludes the paper.

II. Related Work

For the inter-operation between Mobile IPv6 and IPv4 hosts, there are already several RFCs that have been proposed. These RFCs address the IPv6/IPv4 transition problem in general.

In SIIT (Stateless IP/ICMP Translation) [4], IPv4 and IPv6 headers are translated through a middle SIIT server. But each host has one IPv4 address and an IPv6 address so no address efficiency is achieved. NATPT (Network Address Translation - Protocol Translation) [5] combines SIIT with NAT so that multiple hosts can share the same address; however, the drawbacks of NAT also result, as discussed in last section.

BIS (Bump-in-the-Stack) [6] inserts a translator between the IP stack and the network card driver. The translator snoops on the traffic passing by and converts the packet's format between IPv4 and IPv6 versions. BIA (Bump-in-the-API) [7] inserts the translator between the socket module and TCP/IP module so that the IPv4 socket API functions can be translated to/from IPv6 socket API functions. In this way, BIA can avoid the manipulation of the IP header, which needs to distinguish and adjust the part of payload in the packet from the transport layer and application layer. BIS/BIA is only used on IPv4 hosts to enable IPv4 applications to interoperate with IPv6 hosts because of the relative larger number of IPv4 applications. If used in IPv6 hosts, these two mechanisms still need one IPv4 public address for each IPv6 host, so the address efficiency is not achieved either.

TRT (Transport Relay Translator) [8] inserts a transport layer relay system between the IPv6 and IPv4 network to divide one end-to-end TCP/UDP session into two parts: between the IPv6 host and the relay system and between the relay system and the IPv4 host. The relay system converts between these transport sessions. This approach does not support IPSec protocol, which requires end-to-end communications. Also, TRT is still a one-to-one mapping between the IPv4 address and the IPv6 address.

DSTM (Dual Stack Translation Mechanism) [9] keeps an IPv4 public address pool to be temporarily allocated to IPv6 hosts that have the need to communicate with IPv4 hosts. The IPv6 host also implements an IPv4 stack, so no more centralized translator is needed. But the number of IPv4 addresses depends on the number of IPv6 hosts that would communicate with IPv4 hosts, which can be quite high when IPv6 is just deployed.

III. Design Principles

The essence of our approach is to categorize sessions into *calling-in* and *calling-out* sessions. The calling-in session is a session initiated from a CN outside the network; hence the CN must first obtain one MN's public address. In other words, the MN must own a public address while the session is being started. However, this address only needs to be kept unique for a short period of time. After the session has started, the address will be released and assigned to another MN. At this moment, the real MN destined by a packet with a certain destination address can be determined by comparing session parameters, i.e., the source address/port and the destination address/port. Even though the destination address may be the same, for calling-in sessions it is impossible to have two sessions with other parameters also being identical because this situation occurs only if the CN is a server but a server never calls up a client and initiates a calling-in session. In the case of "server-push", the client needs to first send a request for the push operation and then the server can deliver packets back to the client. This makes the session a calling-out session, as described in the next paragraph.

The calling-out session is started by the MN, so the MN can decide on its own address. We statically assign one public address and all its port numbers to be shared among MNs, i.e., all MNs communicate using one same public address but with different port ranges. Since each address can have a maximum of 65,536 port numbers, after deducting the well-known ports, the remaining ports can accommodate a large number of MNs because the number of ports allocated to each MN depends on the maximum number of concurrent *active* sessions (a session needs an address only if it is active), which is quite limited.

IV. Interconnection between Mobile IPv6 and IPv4

A. Some Assumptions

When deploying Mobile IP, the home agent is the gateway dividing the IPv6 and IPv4 network.. A tunnel between the HA and MN exists so that the traffic can be redirected to the foreign network. The MN should support both IPv6 and IPv4

packet formats, which can be achieved either by deploying complete dual stacks or by using BIS/BIA to convert between the two formats.

B. The Calling-in Session

B.1 DNS Query

For calling-in sessions, the first step is for the CN to find the MN. Because the MN's public IPv4 address is no longer static, we need to change the way how the CN addresses the MN. We require when an application causes the CN to communicate with the MN, the user inputs MN's FQDN (Fully Qualified Domain Name) instead of the IP address. For example, in an FTP session the CN should use ftp://save.cs.dal.ca instead of ftp://173.13.6.9. This gives the home network an opportunity to dynamically assign a public IPv4 address to the MN. Most applications support using FQDN in the command; therefore, the software update on the CN can thus be minimized. In order to dynamically allocate IPv4 address to an IPv6 host, the answer to the DNS query must not be cached on intermediate DNS servers. This can be achieved through setting the TTL (Time to Live) value for the DNS record to 0.

B.2 HA Allocating Addresses

Once the DNS server receives such a query, it looks up the MN's IPv6 address and sends it to the HA. The HA maintains several IPv4 addresses in an "Address Table" (Fig.2) and responds with one that is in "idle" state. The DNS server then sends the IPv4 address back to the CN so that the latter can start communicating.

The reason why the Address Table is put on the HA, instead of the DNS server, is that all traffic will be directed to the HA and then delivered to corresponding MNs. The HA needs to know MN's both IPv6 address and dynamically allocated IPv4 home address.

The HA then sets the address status to "monitored" and fills in MN's IPv6 address. If packets with destination address equal to this monitored address arrive, they are checked against a "Session Table", which records the information of all existing sessions, to see if the packets belong to any existing session by comparing their destination and source addresses/ports. As previously discussed, there are no calling-in sessions with identical parameters. If no matching session exists, then this is a new session. The HA creates an entry in the Session Table for the new session and fills in corresponding parameters. Once the new session entry is created in the Session Table, the address status in the Address Table is set to "idle" again so that it can be re-assigned to other MNs. This is different from the traditional use of the address in that it has to last until the MN stops its communication.

If the packet belongs to an existing session, the IPv4 packet is then tunnelled to the MN in an IPv6 tunnel [13], in the same way as specified in Mobile IPv6 specification to tunnel an IPv6 packet.

ADDRESS TABLE

Address	Status	IPv6 Address

SESSION TABLE

Src. IP	Src. Port	Dest. IP	Dest. Port	IPv6 address

Fig.2. Tables on HA

B.3 Translations on the MN

After receiving the packets, the MN extracts the source and destination addresses in the IPv4 header (the inner tunnel header) and then checks against a session table, as shown in Fig.3.

SESSION TABLE ON THE MN

Virtual Addr./port	Session Addr./port	CN Addr./port

Fig.3. Tables on MN

In the table, the virtual address is a fixed address used only within the MN to provide the IPv4 applications a view that the MN has a constant IP address. It can be a private IP address because it is not used outside. The "session address" is the address actually used by the MN in a particular session, in this case, it is the destination address in the IPv4 header, i.e., the address assigned by the HA for the session. The "CN address/port" is the address/port of the communication partner, gained from the responding fields in the inner IPv4 packet.

If this is a new packet, which does not belong to any existing session in the session table, the MN creates a new entry in its session table and then performs the address translation as described next. If this packet belongs to an existing session, an address translation from the session address to the virtual address is performed before the packet is handed over to the higher layer. For outgoing packets, the translation is from the virtual address to the session address. To determine the session address of one packet, the "CN address/port" is compared against the Session Table. Note that there are no port translations because session addresses are sufficient to identify different sessions for calling-in sessions. In other words, the port numbers remain intact so that servers can run on the MN.

B.4 Reverse Traffic

The reverse traffic from the MN to the CN should be tunneled to the edge of the IPv6 and IPv4 networks and then de-capsulated and routed in the IPv4 network. The HA can be one edge-router, but this may result in the triangle routing problem. To solve this problem, route optimization can be implemented if the CN supports it [1]. The requirement is for the local router itself or to find another "edge router" to de-

tunnel and route IPv4 packets. The format of the binding update message in Mobile IP should be modified accordingly.

C. The Calling-out Session

For calling-out sessions, the IPv4 stack on the MN uses a pre-configured public address in a certain range of ports. The justification is that in calling-out sessions the MN is the "client" – this means that the ports are not required to be standard numbers. Therefore, the translation from the original port to one of the assigned ports will not cause failures in applications.

Each address has at most 65,536 port numbers and the number of ports required by each MN should be greater than the number of *concurrent, active* calling-out sessions, which we take 20 in average (some hosts have many sessions turned on but they are passive). To avoid confusion on routers, the 1,024 well-known ports are not used [14]; therefore, one public address can be shared among (65,536 – 1,024)/20 = 3,225 MNs. A home network can choose to use one or more public addresses depending on the number of MNs it has.

In calling-out sessions, the IPv4 stack on the MN intercepts the packet from the upper layer to compare the address/port against the Session Table. If no matching session is found, this is considered to be a new calling-out session. A new port number is then allocated to the new session from the allocated port range to the MN and a new entry is created in the Session Table with the "session address" field equal to the static public address and the "session port" equal to the newly assigned port. The MN translates the source address/port of the packet from virtual address/port to the session address/port and then the packet is tunnelled to an "edge router" to be de-capsulated and routed to the IPv4 network.

When the incoming packet arrives, the translation is done in the reverse order, i.e., from the session address/port to the virtual address/port, then the packet is delivered to the upper layer.

V. Other Applications

Even though this approach is originally intended for use in the Mobile IP scenario, it can be used as a general transition mechanism for the IPv6 host to communicate with IPv4 hosts. The procedure is then simplified in that the HA does not exist any more and the local router plays the role of the HA.

As we mentioned in Introduction, Mobile IPv4 requires each MN to have one public home address and causes a large demand for the already-limited IPv4 address space. The same address-sharing mechanism can be also used in pure Mobile IPv4 scenario for the MN to share home addresses. In this scenario, the MN only deploys IPv4 stack and no format translations between IPv6 and IPv4 packets are necessary. Please refer to another paper of ours for a detailed discussion [10].

VI. Discussion

A. IPSec Transparency

Unlike NAT or other approaches, this approach is transparent to IPSec. IPSec uses two headers to protect the payload of the packet – the ESP (Encapsulating Security Payload) and the AH (Authentication Header). These two headers are calculated based on the IP header discovered on one end of the Security Association (SA). If the IP header is modified, when the other end of the SA receives the packet and re-computes the ESP or AH header, there will be a mismatch between the ESP/AH headers. The other end will take the mismatch as the sign that an attacker has tampered with the packet and hence it will discard the packet. In our approach, even though the HA monitors the traffic, the only thing it does is to add the outer IP header and tunnel the packet. The translation of the address/port occurs on the MN, hence can be arranged before the security computation in IPSec for out-going packets (after for incoming packets) to achieve the transparency.

B. Server Configuration on the MN

Usually in NAT-like approaches, the MN can only work as a client, vs. a server whose services are accessed. The reason is not only because the external visitor cannot initiate a call to the MN, but also with port translations, only one MN can run a server using the specific port number at any time. However, the approach proposed in this paper allows the MN to run servers independently because calling-in sessions are distinguished through comparing the address and port of the CN side, hence no port translation on MN side is required.

C. The Overhead

This approach is similar to NAT but with stronger functions in that each MN can communicate like it had one public address. To achieve this, several overhead sources are also created compared with the original Mobile IP protocol. One overhead is incurred at the HA to examine packets to nail down the correct MN. However, comparing with the overhead if NAT is deployed, this overhead is less because only calling-in session packets require searching the Session Table, which is the most time-consuming operation. The packets of calling-out sessions are identified by the statically allocated source/destination address and passed through without further processing.

The second overhead comes from the elimination of the caching of MN addresses on intermediate DNS servers. This overhead is not significant since all callers are scattered around the Internet and it is unlikely for them to share cached addresses.

The last overhead source, the translations on the MN, is also considered to be ignorable under current CPU power although more precise measurement may need to be performed in order to accurately determine this overhead.

D. Implementation Feasibility

The implementation of this proposal is practically feasible. On the home network side, modifying DNS software to communicate with other entities, such as the HA, has already been proposed in other literatures. Also, maintaining a virtual address on the MN and translating packet addresses back and forth is also possible. VNAT [11] (Virtual Network Address Translation) is a *Sourceforge* [12] open-source project. Even though the purpose is different, VNAT does implement the virtual address translation on Linux. It is implemented as a loadable kernel module so that it is possible to patch an existing MN to support the function. Also, the VNAT project has tested the use of the "virtual address" for some popular applications.

E. Limitation

Similar to NAT, even though the address translation is only performed on the MN, the approach is not transparent to the applications in which the IP address is not only included in the header, but also in the packet payload (as explained in the Introduction section). However, if the programmer uses the FQDN to identify the node when programming the application and only accepts the address in the IP header, then this problem can be alleviated. Actually using FQDN to identify a host is more desirable since the host name and the address should be different after all.

VII. ANALYSIS

In this section, we model the system using an $M/E_k/1$ queue to analyze the performance. "M" refers to Markov, meaning that calls (address allocation requests for calling-in sessions) arrive according to a Negative Exponential Distribution:

$$f(t) = \lambda \cdot e^{-\lambda t} \tag{1}$$

Here λ is the request arrival rate (requests/s).

The request response process is modeled using Erlang Distribution ("Er"), which is a more general Negative Exponential Distribution:

$$f(t) = \frac{(\alpha k)^k}{(k-1)!} \cdot t^{k-1} \cdot e^{-k\alpha t} \tag{2}$$

Here α is the response rate. Parameter k is a "shaper"; if $k = 1$, then Erlang Distribution is reduced to Negative Exponential Distribution.

The last "1" in the notation means that we have only one IPv4 address existing in the Address Table to be allocated to the MN.

Fig. 4 shows the expected length of this queue with different k and ρ values. ρ is calculated by following formula:

$$\rho = \frac{\lambda}{c\alpha} \tag{3}$$

In other words, ρ is the ratio of request arrival rate over the number of addresses (c = 1) times the request response rate. It is a measurement of the request intensity for the single IPv4 public address. From top down, the curves in the graph are ordered with k values from 1 to 8, and finally k = infinity.

When ρ=90%, we can see that the M/E1/1 queue has the longest expected queue length (8.1). Even with the longest length, the expected waiting time for each request in the queue is small because the time of the address being set to "monitored" state, which is equal to the round trip time between the HA and CN, is at most several hundred milli-seconds. If we take 20ms as the lower bound and 300ms (there may be queuing delay on the CN) as the upper bound, then we can have the mean response time of (20 + 300)/2 = 160ms. With 8.1 requests in the queue, the average waiting time is 160ms * 8.1 = 1.3s. On the CN side, the timer waiting for a DNS answer can usually tolerate several seconds.

When ρ becomes larger than 90%, there will be a dramatic increase in the queue length, as shown in Table I:

TABLE I
QUEUE LENGTH FOR P>90%

k \ ρ	95%	98%	99%
1	18.05	48.02	98.01
2	13.54	36.02	73.51
3	12.03	32.01	65.34
∞	9.03	24.01	49.01

When k=1 and ρ=99%, the theoretical queue size is 98.01. This means that the waiting time would be 98.01 * 160ms = 15.68s. If this is intolerable, we should add more public IPv4 addresses in the Address Table.

Next, let us calculate the number of requests being served with only one IPv4 public address. Let us take ρ=95%, then the maximum waiting time is around several seconds (18.05 * 160ms = 2.89s). If we deem this as tolerable, with α= 1/160ms (the response rate is the reverse of the average response time), c = 1, and ρ=95%, we will have λ= ρ * c * α= 5.94. In other words, the request arrival rate is actually 5.94 requests/second. This is equal to 21,384 requests per hour. If we subtract the requests still in the queue (18.05), we still have in average around 21,365 requests being responded per hour. This figure tells us that, with a certain number of MNs and only one public IPv4 address, 21,365 calling-in sessions can be served with tolerable waiting delays. If the number of con-current sessions does not exceed 20, then the number of MNs in the home network can be over 10,000.

VIII. CONCLUSION

In this paper, we have proposed an address-sharing approach that achieves high address efficiency and keeps the normal activities in the end-to-end Internet such as initiating calls from outside the network, running services, and being transparent to other protocols such as IPSec. It can be not only used in Mobile IP protocol to save IPv4 public addresses in both Mobile IPv4 scenario and Mobile IPv6 interoperating with IPv4 hosts, but also as a general IPv6 and IPv4 transition technique.

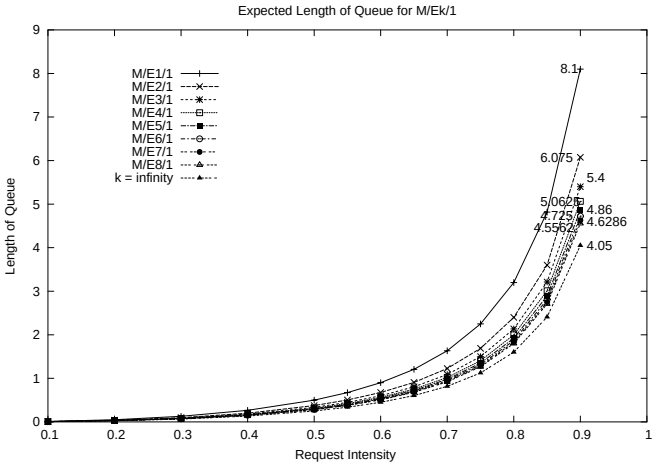

Fig. 4. Expected Length of Queue with Different k Values

REFERENCES

[1] C. Perkins, Ed. *IP Mobility Support for IPv4*, RFC3344, August 2002.

[2] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*, Network Working Group, RFC2460, December 1998.

[3] D. Johnson, C. Perkins,and J.Arkko, *Mobility Support in IPv6*, Network Working Group, RFC3775, June 2004.

[4] E. Nordmark. *Stateless IP/ICMP Translation Algorithm (SIIT)*, RFC 2765, February 2000.

[5] G. Tsirtsis and P. Srisuresh, *Network Address Translation - Protocol Translation (NAT-PT)*, Network Working Group, RFC2766, February 2000.

[6] K. Tsuchiya, H. Higuchi and Y. Atarashi, *Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS)*, Network Working Group, RFC2767, February 2000.

[7] S. Lee, M-K. Shin, Y-J. Kim, E. Nordmark and A. Durand, *Dual Stack Hosts Using "Bump-in-the-API"(BIA)*, Network Working Group, RFC3338, October 2002.

[8] J. Hagino and K. Yamamoto, *An IPv6-to-IPv4 Transport Relay Translator*, Network Working Group, RFC3142, June 2001.

[9] Dual Stack Transition Mechanism, URL: http://www.ipv6.rennes.enst-bretagne.fr/dstm/, retrieved on Nov. 2005.

[10] Z. Zhen and S. Sampalli, *Saving Public Addresses in Mobile IP*, To be published *in Proc. 5th International Conference on Networking (ICN'06)*, April 23-26, 2006, Mauritius.

[11] Gong Su and Jason Nieh, *Mobile Communications with Virtual Network Address Translation*, Technical Report CUCS-003-02, Columbia University, February 2002.

[12] *VNAT (Virtual Network Address Translation) Project in Sourceforge*, http://sourceforge.net/projects/vnat-linux, retrieved on October 24, 2005.

[13] Conta, A. and S. Deering, *Generic Packet Tunneling in IPv6 Specification*, RFC 2473, December 1998.

[14] J. Reynolds, editor. *Assigned Numbers*, RFC 3232, January 2002.

[15] H.Levhowetz and S.Vaarala. *Mobile IP Traversal of Network Address Translation (NAT) Devices*, RFC 3519, April 2003.

Zhen Zhen is currently a Ph.D. student in the Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada. Her research interests are in mobility management, security and AAA in heterogeneous wireless networks. She is an IEEE student member. She received her M.Sc. in 1997 from the Beijing University of Posts and Telecommunications and her B.Sc. in 1992 from the Xi'an University of Electronics Science and Technology. She also worked for several years as a software developer.

Srinivas Sampalli is a Professor and 3M Teaching Fellow in the Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada. He has been actively researching in the area of security and quality of service in wireless and wireline networks. Specifically, he has been involved in research projects on protocol vulnerabilities, security best practices, risk mitigation and analysis, and the design of secure networks. He was the Dalhousie principal investigator for the Secure Active VPN Environment (SAVE) project sponsored by the Canadian Institute for Telecommunications Research (CITR), a National Center for Excellence, and is currently the principal investigator for the Wireless Security project sponsored by Industry Canada. Dr. Sampalli has received many teaching awards, at the faculty-level, university-level and regional level. In 2005, he was honoured with the 3M Teaching Fellowship, Canada's most prestigious teaching acknowledgment.