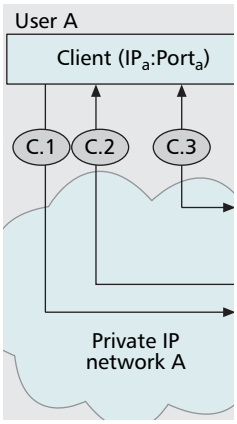
ACCEPTED FROM OPEN CALL

iP2P: A PEER-TO-PEER SYSTEM FOR MOBILE DEVICES

CHIEN-CHUN HUANG-FU AND YI-BING LIN, NATIONAL CHIAO TUNG UNIVERSITY
HERMAN CHUNG-HWA RAO, FAR EASTONE TELECOMMUNICATIONS CO., LTD.



The authors propose iP2P, a hybrid P2P system for mobile devices. iP2P utilizes the short message service as the control protocol to identify the address of the called peer.

ABSTRACT

Peer-to-peer communications is a major trend in wireless Internet. In a P2P system, the calling peer must identify the network address of the called peer before establishing a P2P connection. This article proposes iP2P, a hybrid P2P system for mobile devices. iP2P utilizes the short message service as the control protocol to identify the address of the called peer. Our approach provides an efficient identification mechanism without the requirement for the maintenance of a centralized registrar server in a hybrid P2P system. We also show how iP2P can integrate effectively with the existing Network Address Translation traversal mechanisms to solve the private IP address issue.

INTRODUCTION

Peer-to-peer (P2P) communication has become one of the major trends in wireless Internet. In a P2P system, a participating peer plays the roles of both client and server. Logically, the peers establish P2P connections directly with each other without passing through network servers. An Internet-based P2P communication session consists of two phases: the identification phase and the communications phase. In the identification phase, the calling peer identifies the *transport address* of the called peer (i.e., the Internet Protocol [IP] address and the port number). In the communications phase, the calling peer directly connects to the called peer based on the transport address retrieved in the identification phase.

Existing IP-based P2P systems can be categorized into two types: *hybrid P2P* and *pure P2P*. Hybrid P2P systems utilize centralized servers to maintain the mapping between the identifications and the transport addresses of the participating peers. The P2P connection procedure of a hybrid P2P system consists of the following steps:

Step A.1 To join the system, a peer starts the P2P application, which attaches the peer to the IP network.

Step A.2 The peer registers to the centralized server by providing its identification and the transport address.

Step A.3 To initiate a P2P connection, the calling peer queries the centralized server to retrieve the transport address of the called peer.

Step A.4 Upon receipt of the transport address of the called peer, the calling peer establishes a P2P connection directly to the called peer without involving the centralized server.

In this procedure, Steps A.1 and A.2 are executed to maintain the mapping between the identification and the transport address. Step A.3 executes the identification phase, and the communications phase starts with Step A.4. A hybrid P2P example is the conventional Session Initiation Protocol (SIP) system, where the peers are SIP user agents (UAs), and the centralized server is a SIP registrar server that maps the SIP uniform resource identifier (SIP-URI) to the transport address (IP address and port number) for each registered UA. A hybrid P2P system provides efficient query at the cost of maintaining the address mapping in a centralized server (i.e., Steps A.1 and A.2), which may incur a scalability problem.

In contrast, the identification mechanism of a pure P2P system is distributed among the participating peers. Depending on how the peers communicate in the identification phase, a pure P2P system can be unstructured or structured. In an unstructured P2P system, a query for identifying the transport address of a participating peer is flooded through the network. Unstructured P2P is not required to maintain centralized mapping between the identifications and the transport addresses at the cost of expensive flooding query. In contrast, in a structured P2P system, the identification phase is achieved by a peer querying other peers in a specific routing structure (e.g., a ring, a grid, or a tree). A structured P2P example is the peer-to-peer SIP system [1] that utilizes the distributed hash table for transport address registration and query. Structured P2P provides efficient query in a decentralized fashion at the cost of maintaining the distributed mapping tables among the participating peers.

To implement a P2P system for mobile devices that have the capability to access mobile telecommunications networks, such as the universal mobile telecommunications system (UMTS) [2, 3], we propose a hybrid P2P system called *iP2P*. iP2P effectively reuses the UMTS mobility management mechanism and short message service

1536-1284/09/$25.00 © 2009 IEEE

(SMS) as the control protocol in the identification phase and saves the power of the mobile devices significantly. This article describes the iP2P system and shows how it works when the peers are assigned private IP addresses.

iP2P System Architecture

In the iP2P system, a participating peer utilizes its mobile station ISDN number (MSISDN; the telephone number) as the peer identification, which is globally unique. The iP2P control messages are encapsulated in the short messages. As illustrated in Fig. 1, the SMS delivery procedure consists of the following steps: when the calling peer (Fig. 1 (1)) sends a short message to the called peer (Fig. 1 (9)), this message is delivered to the source mobile-switching center (MSC; Fig. 1 (3)) through the base station subsystem (BSS; Fig. 1 (2)). The source MSC passes this message to a short message service center (SMSC; Fig. 1 (4)), which then forwards the message to the target MSC (Fig. 1 (7)) through the SMS gateway MSC (SMS GMSC; Fig. 1 (5)). The SMS GMSC identifies the target MSC of the called peer by interrogating the home location register (HLR; Fig. 1 (6)). Then the GMSC forwards the message to the target MSC that delivers the message to the called peer (Fig. 1 (9)) through the BSS (Fig. 1 (8)). In this system, the HLR is the centralized server that maps the telephone number to the location of the called peer.

In iP2P, we assume that the mobile devices can communicate with each other through the SMS network (path (a) → (b) → (c) → (d) → (e) in Fig. 1) or through the broadband wireless (BW) network (e.g., WiFi, WiMAX, or wideband code division multiple access [WCDMA]; path (f) in Fig. 1). In UMTS, both the SMS and the WCDMA BW networks are accessed through the same base stations. We separate them in Fig. 1 for the purposes of description.

With the background knowledge of SMS, we describe the iP2P modules in a mobile device. When a client application (e.g., a Telnet or File Transfer Protocol [FTP] client; Fig. 2 (1)) is executed, a connection process is invoked for establishing a connection to the server application in another mobile device (Fig. 2 (2)). The iP2P control function (Fig. 2 (3)) is responsible for accessing the iP2P control messages through the SMS API (Fig. 2 (4)) and controlling the connection through the iP2P socket API (Fig. 2 (5)). The iP2P socket API is the standard socket API with minor modifications. Specifically, the domain name system (DNS) resolution procedure within the iP2P socket API is capable of recognizing the MSISDN format. If the domain name for DNS resolution is an MSISDN, the iP2P socket API does not follow the standard procedure to query the DNS server in the IP network. Instead, it directly passes the MSISDN to the iP2P control function. The iP2P control function then queries the transport address of the called peer through SMS (Fig. 2 (6) and path (a)→(b)→(c)→(d)→(e) in Fig. 1). After the calling peer has identified the transport address of the called peer in the identification phase, a BW module (Fig. 2 (7)) is activated to deliver user data in the communications phase (path (f) in Fig. 1).

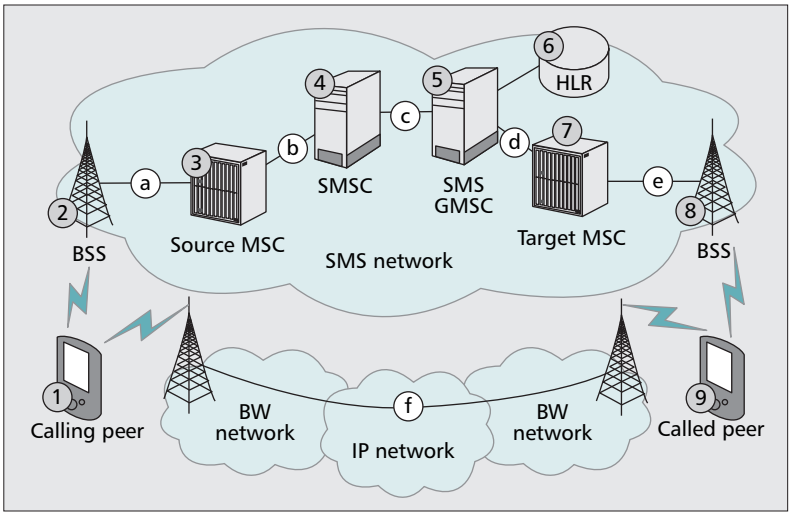


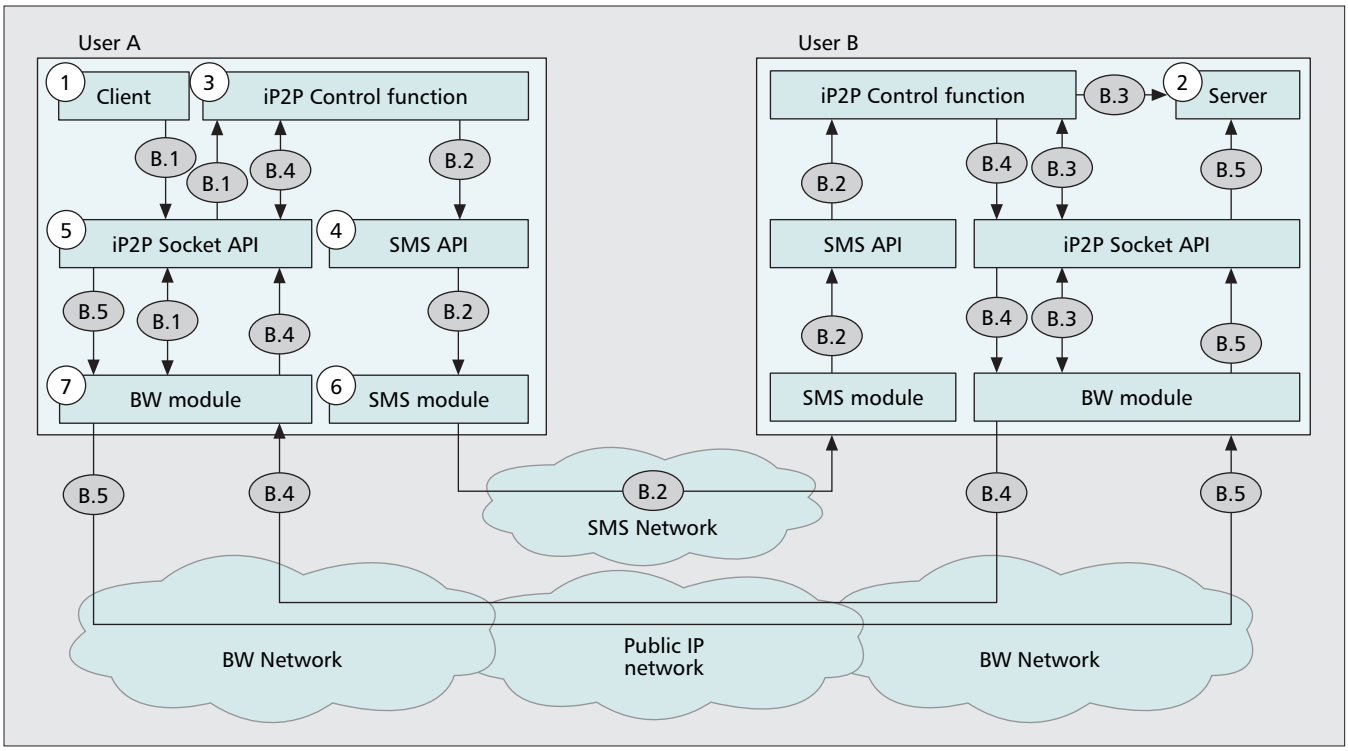
Figure 1. *Wireless and mobile service network architecture.*

A BW module provides high-speed access to the IP network at a cost of consuming much more power than the SMS module. For example, the power consumption of Cisco PCM-350 is 390 mW in the idle mode and 1600 mW in the active mode, whereas the power consumption of the global system for mobile communications (GSM) module is about 10mW in the idle mode [4]. Therefore, in a mobile device, the BW modules are typically turned off, and only the SMS module (and the voice module) is turned on to receive the signals from the outside world. In iP2P, we assume that the called peer only turns on the SMS module in the identification phase. After the called peer receives the iP2P control message through SMS, its BW module is turned on to attach the called peer to the IP network. Then, the P2P connection can be established in the communications phase.

In most mobile telecom operations, the mobile devices are not assigned static IP addresses. Therefore, this article assumes that an iP2P peer can reside in the public (or private) IP network in which the IP address is dynamically allocated, for example, through the Dynamic Host Configuration Protocol (DHCP). Furthermore, when a calling peer in the private IP network connects to a called peer outside its local network, a network address translation (NAT) server is required to translate the private transport addresses to a public transport address. For User Datagram Protocol (UDP) applications (e.g., Voice over IP) implemented in iP2P, the NAT traversing issue is solved by the interactive-connectivity-establishment (ICE) mechanism [5]. On the other hand, for the Transmission Control Protocol (TCP) applications (e.g., HTTP and FTP), the NAT server must monitor the TCP connection state or the sequence number in the TCP header to screen illegal packets. We focus on the TCP connection set up in this article.

iP2P Connection Setup for Public IP Network

Figure 2 illustrates the TCP connection establishment between two iP2P users residing in the public IP network. In this case, no NAT is



■ **Figure 2.** *iP2P connection setup (public-to-public).*

involved. Assume that the SMS modules of both peers are activated, and the BW modules are turned off to save power (and therefore, are detached from the IP network).

Step B.1 User A activates the BW module, attaches to the IP network, and obtains the public IP address IP_A through the DHCP. Using User B's MSISDN, User A starts the client application (called App-A) and executes the DNS resolution procedure with User B's MSISDN. In DNS resolution, the iP2P socket API detects that the destination domain name is an MSISDN. Instead of querying an external DNS server, the iP2P socket API reserves a local port (i.e., $Port_A$) for App-A and passes the application name and the App-A transport address (i.e., IP_A:$Port_A$) to the iP2P control function.

Step B.2 User A's iP2P control function sends an iP2P control message including the App-A transport address (IP_A:$Port_A$) and the application name (e.g., FTP) to user B through SMS. User A's iP2P control function then listens on $Port_A$.

Step B.3 Upon receipt of the iP2P control message, user B activates the BW module, attaches to the IP network, and obtains the public IP address IP_B through the DHCP. Then, user B starts the corresponding server application (called App-B) and listens on $Port_B$.

Step B.4 User B's iP2P control function sends the App-B transport address (IP_B:$Port_B$) and the application name to user A's iP2P control function through the BW and the IP networks. Upon receipt of user B's response, user A's iP2P socket API maps user B's MSISDN to its transport address.

Step B.5 App-A establishes a TCP connection to App-B through the IP network.

In this procedure, the identification phase consists of Steps B.1–B.4, and the communications phase starts with Step B.5.

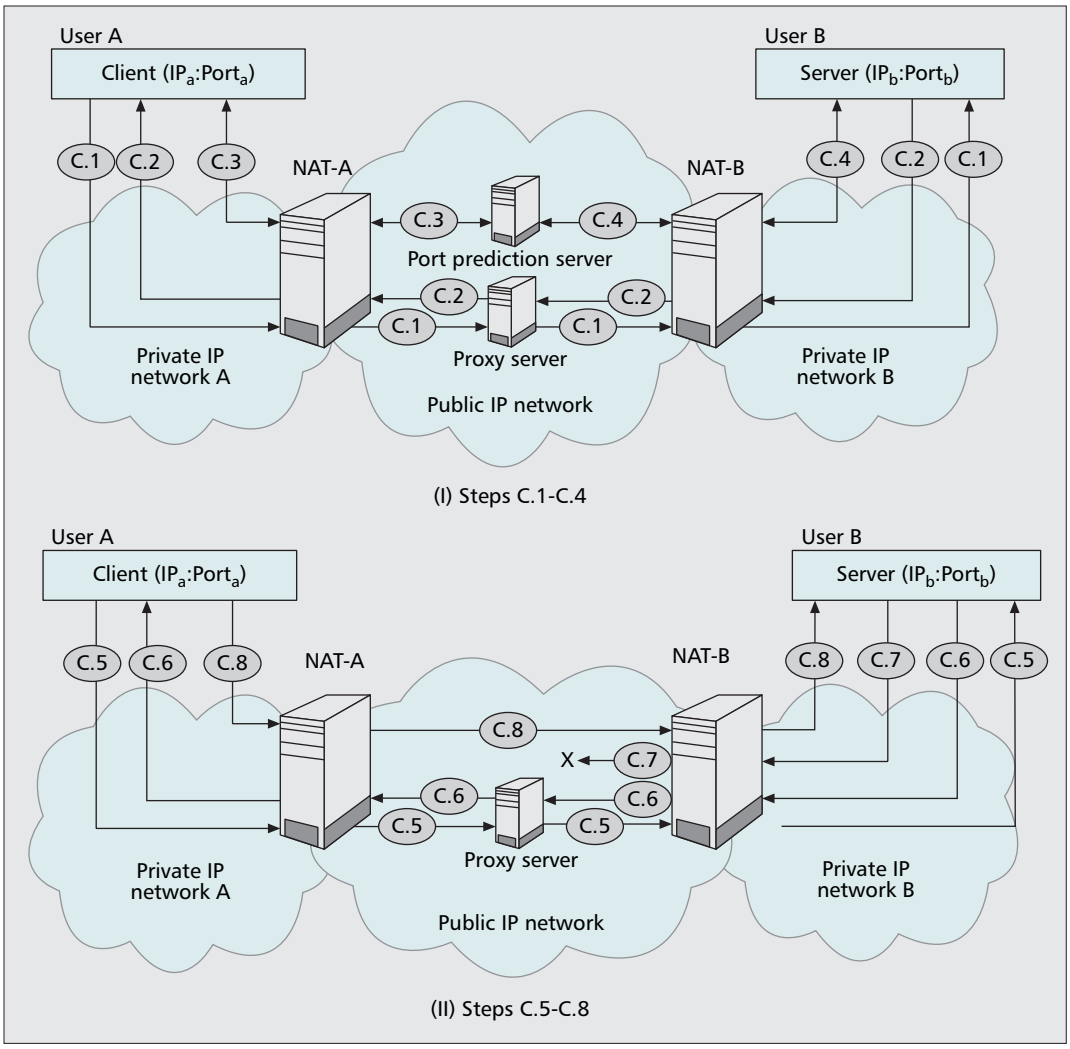
iP2P Connection Set Up for Private IP Networks: STUNT #2

Suppose that the calling and the called peers reside in different private IP networks. To resolve the NAT traversing issue for TCP, iP2P must adopt a NAT traversal mechanism [6–8] to establish the P2P connection. In this section, we use the STUNT #2 approach [6] as an example to illustrate how iP2P solves the NAT traversing issue for a P2P application.

First we describe the STUNT mechanism when both the server and the client applications already have attached to the private IP networks. As shown in Fig. 3, user A (the STUNT client) resides in the private IP network A behind the NAT server NAT-A, and user B (the STUNT server) resides in another private IP network B behind the NAT server NAT-B. The P2P connection set-up procedure is described as follows:

Step C.0 At the beginning, both the user A client application App-A and the user B server application App-B establish TCP connections (called $Connection_A$ and $Connection_B$, respectively) to a proxy server in the public IP network and then register their identifications to the proxy server through these TCP connections. $Connection_A$ and ConnectionB are always maintained so that App-A and App-B can access the public IP network through the proxy server.

Step C.1 To inform user B to establish a P2P connection, user A sends a connection request to the proxy server with user B's identification through $Connection_A$. The proxy server then



■ **Figure 3.** *Connection setup of STUNT #2 approach.*

iP2P effectively reuses the UMTS mobility management mechanism and short message service (SMS) as the control protocol in the identification phase and saves the power of the mobile devices significantly.

relays this connection request to user B through Connection_B.

Step C.2 If user B accepts the connection request, it replies with an acknowledgment to user A through the same path in the reverse direction.

We note that Connection_A and Connection_B are not used for P2P data transfer to avoid the scalability problem in the proxy server. Therefore, to establish a P2P connection directly between users A and B, STUNT utilizes a port prediction server to find the public transport addresses for the private transport addresses of the peers at Steps C.3 and C.4 (described below). Then, the corresponding mapping entries are created in NAT-A and NAT-B, respectively, at Steps C.7 and C.8. Then, the P2P connection can be established directly between App-A and App-B.

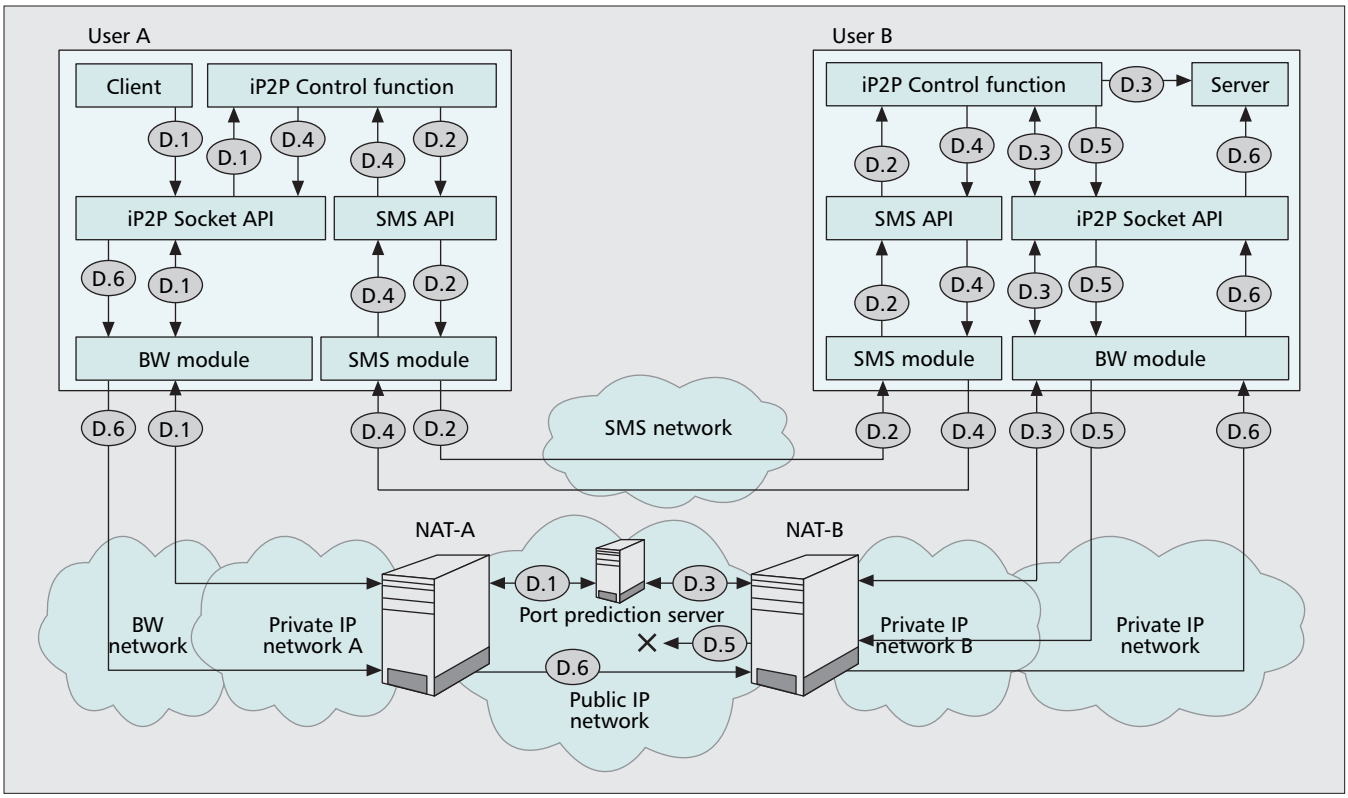
Step C.3 Upon receipt of user B's acknowledgment through Connection_A, App-A reserves a local port (i.e., Port_a), and uses private transport addresses (IP_a:Port_a) to query a port prediction server in the public IP network [9]. The port prediction server predicts a public transport address (IP_A:Port_A) that will be assigned by NAT-A to map to the private transport address (IP_a:Port_a) of the P2P connection.

Step C.4 Similarly, App-B reserves a local port (i.e., Port_b) for the P2P connection and queries the port prediction server to predict public transport addresses (IP_B:Port_B) to be assigned by NAT-B.

Step C.5 App-A sends the mapping of its public and the private transport addresses (IP_a:$\text{Port}_a \leftrightarrow \text{IP}_A$:$\text{Port}_A$) to App-B through Connection_A and Connection_B.

Step C.6 Similarly, App-B sends the mapping of its public and the private transport addresses (IP_b:$\text{Port}_b \leftrightarrow \text{IP}_B$:$\text{Port}_B$) to App-A through the same path in the reverse direction.

Step C.7 Upon receipt of App-A's mapping of its public and the private transport addresses, App-B sends a synchronize (SYN) packet from App-B's private transport address (IP_b:Port_b) to App-A's public transport address (IP_A:Port_A). This SYN packet is sent with a time-to-live (TTL) parameter large enough to enable the SYN packet to pass through NAT-B so that the corresponding mapping entry can be created in NAT-B. On the other hand, the TTL value must be sufficiently small such that the packet is dropped before it arrives at NAT-A. If this SYN packet arrives at NAT-A before NAT-A has created



■ **Figure 4.** *iP2P connection setup with STUNT (private-to-private).*

(a) The mapping entry in NAT-B

Source address	Alias address	Destination address
IP_b:$Port_b$	IP_B:$Port_B$	IP_A:$Port_A$

(b) The mapping entry in NAT-A

Source address	Alias address	Destination address
IP_a:$Port_a$	IP_A:$Port_A$	IP_B:$Port_B$

■ **Table 1.** *The mapping entries in NAT servers.*

the corresponding mapping entry (which is performed at Step C.8), NAT-A might regard it as an attack packet and block $Port_A$ in NAT-A. When this SYN packet passes through NAT-B, NAT-B creates the mapping entry as illustrated in Table 1a. We assume that the SYN packet passes through NAT-B and is dropped before it arrives at NAT-A.

Step C.8 App-A waits for a period (so that the corresponding mapping entry in NAT-B has been created at Step C.7), and sends a SYN packet from its private transport address (IP_a:$Port_a$) to App-B's public transport address (IP_B:$Port_B$). Upon receipt of the SYN packet, NAT-A creates the mapping entry as illustrated in Table 1b. This SYN packet passes through NAT-B because NAT-B already has the corresponding mapping entry. Upon receipt of the SYN packet, App-B executes the standard TCP three-way handshake procedure, and the P2P connection is established.

Figure 4 illustrates the connection set-up procedure of iP2P with the STUNT #2 approach. As described in the previous section, we assume that the SMS modules of both peers are activated, and the BW modules are turned off to save power. The following steps are executed.

Step D.1 User A activates the BW module, attaches to the IP network, and obtains the private IP address IP_a through the DHCP. User A then starts the client application App-A and executes the DNS resolution procedure with user B's MSISDN. The iP2P socket API reserves a local port (i.e., $Port_a$) for App-A, queries the port prediction server to obtain a predicted public transport address (IP_A:$Port_A$), and passes the application name and the App-A mapping of its public and the private transport addresses (IP_a:$Port_a$ ↔ IP_A:$Port_A$) to the iP2P control function.

Step D.2 User A's iP2P control function sends an iP2P control message with the App-A mapping of its public and the private transport addresses (IP_a:$Port_a$ ↔ IP_A:$Port_A$) and the application name to user B through SMS.

Step D.3 Upon receipt of the iP2P control message, user B activates the BW module, attaches to the IP network, and obtains the private IP address IP_b through the DHCP. User B then starts the server application App-B, which listens on $Port_b$. User B uses the private transport address (IP_b:$Port_b$) to query the port prediction server to obtain a predicted public transport address (IP_B:$Port_B$).

Step D.4 According to the content of user A's iP2P control message, user B knows that user A is in a different private IP network. User B

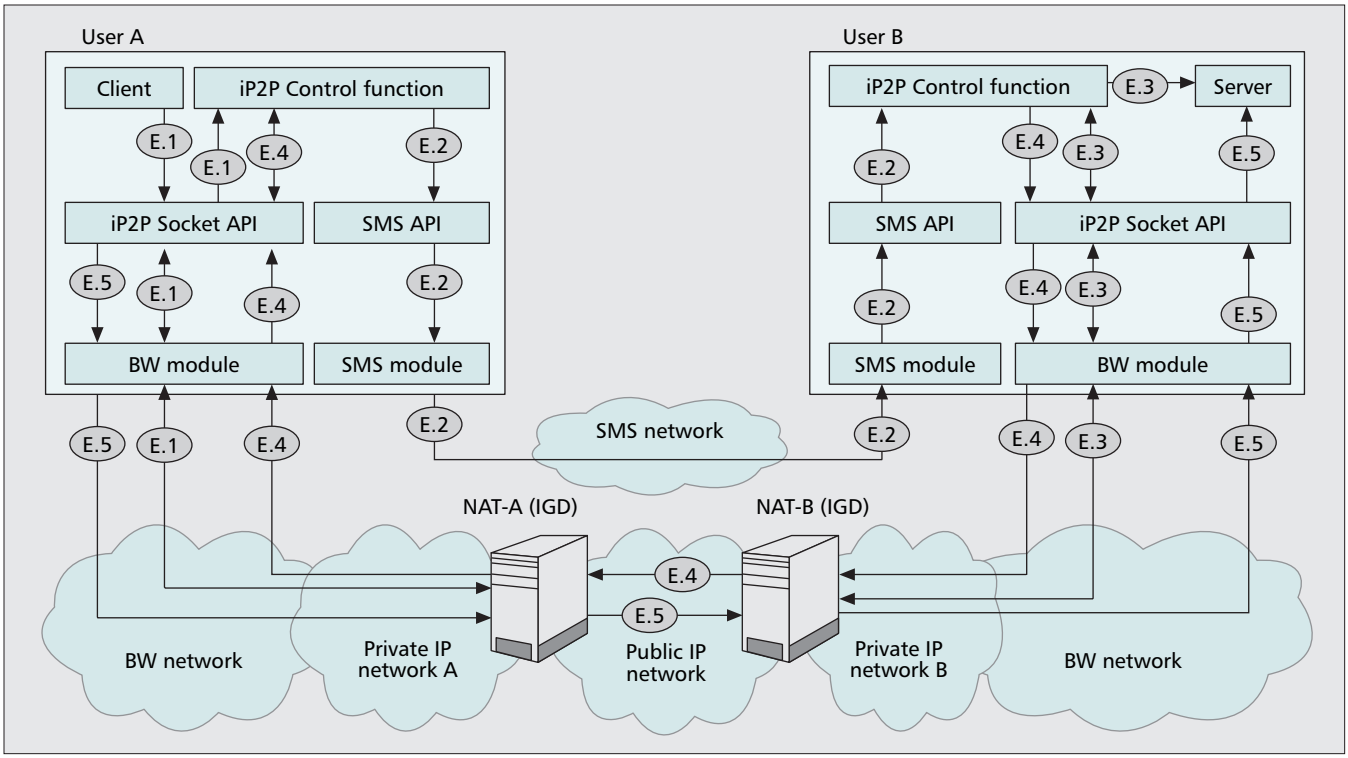


Figure 5. *iP2P connection setup with UPnP (private-to-private).*

replies with an iP2P control message with the App-B public transport address (IP_B:$Port_B$) and the application name to user A through SMS.

Step D.5 App-B sends a SYN packet with a proper TTL value from the App-B private transport address (IP_b:$Port_b$) to the App-A public transport address (IP_A:$Port_A$). When this SYN packet passes through NAT-B, a mapping entry is created in NAT-B as shown in Table 1a. We assume that the SYN packet is dropped before it arrives at NAT-A.

Step D.6 App-A waits for a period (so that the corresponding mapping entry in NAT-B has been created at Step D.5) and sends a SYN packet from its private transport address (IP_a:$Port_a$) to the App-B public transport address (IP_B:$Port_B$). This SYN packet results in the mapping entry creation in NAT-A as shown in Table 1b. Upon receipt of this SYN packet, App-B executes the standard TCP three-way handshake procedure, and the P2P connection is established.

Clearly, iP2P, with the STUNT #2 approach, can solve the NAT traversing issue for TCP as the conventional STUNT #2 approach without the maintenance of the centralized proxy server (see Step C.0), and both $Connection_A$ and $Connection_B$ are eliminated. Furthermore, the called peer is not required to attach to the IP network before the iP2P connection is initiated. Note that the STUNT solution may pose problems. First, setting an appropriate TTL value of the SYN packet (at Step C.7 and Step D.5) might not be trivial. Furthermore, the port prediction server cannot guarantee to provide the precise mapping of the public and the private transport addresses for all the NAT servers. For example, some NAT servers randomly assign the mapping ports, which cannot be predicted.

iP2P Connection Set Up for Private IP Networks: UPnP

Universal plug and play (UPnP) [10] provides another solution to solve NAT traversing for TCP. Usually, a UPnP system consists of several UPnP clients and an Internet gateway device (IGD), which provides Internet connectivity for the protected local area network (LAN). UPnP is a network protocol for automatic discovery and configuration when a device (i.e., a UPnP client) is online. Therefore, the mapping of the public and the private transport addresses in both the UPnP client and the IGD can be established automatically by the UPnP protocol. UPnP can be integrated easily with iP2P to support NAT traversal for the peers in different private IP networks. To support UPnP in an iP2P peer, the iP2P socket API must include the UPnP API functionalities, and the iP2P control function on the mobile device is a UPnP client. In Fig. 5, suppose that both NAT-A and NAT-B support the IGD functionalities. The steps of iP2P connection set up are described as follows:

Step E.1 User A activates the BW module, attaches to the IP network, and obtains the private IP address IP_a through the DHCP. User A starts the client application App-A and executes the DNS resolution procedure with user B's MSISDN. Because user A resides in a private IP network, its iP2P socket API issues a UPnP multicast `M-SEARCH` request to identify the IGD (i.e., NAT-A). Then, it sends a UPnP `GetExternalIPAddress` request to

iP2P can integrate easily with the existing NAT-traversal mechanisms to support mobile devices residing in the private IP networks. The mechanism of iP2P is pending U.S. and Republic of China (R.O.C.) patents.

NAT-A to retrieve the NAT-A public IP address IP_A. The iP2P socket API reserves a local port $Port_a$ for App-A and sends a UPnP `AddPortMapping` request to NAT-A for establishing the mapping entry in NAT-A (i.e., IP_a:$Port_a \leftrightarrow IP_A$:$Port_A$ illustrated in Table 1b). The iP2P socket API passes the public transport address (IP_A:$Port_A$) and the application name to the iP2P control function.

Step E.2 User A's iP2P control function sends an iP2P control message with its public transport address (IP_A:$Port_A$) and the application name to user B through SMS. The iP2P control function then listens on $Port_a$.

Step E.3 Upon receipt of user A's iP2P control message, user B activates the BW module, attaches to the private IP network, and obtains the private IP address IP_b through the DHCP. User B's iP2P control function then starts the corresponding server application App-B, which listens on $Port_b$. Because user B resides in a private IP network, it establishes the mapping entry in its IGD NAT-B (i.e., IP_b:$Port_b \leftrightarrow IP_B$:$Port_B$ illustrated in Table 1a) following the UPnP message exchange as described in Step E.1.

Step E.4 User B's iP2P control function sends its public transport address (IP_B:$Port_B$) and the application name to user A through the IP network by using user A's public transport address (IP_A:$Port_A$). This message can pass through NAT-A because the corresponding mapping entry (IP_a:$Port_a \leftrightarrow IP_A$:$Port_A$) has already been created in NAT-A at Step E.1. Upon receipt of user B's response, user A's iP2P control function passes the App-B public transport address (IP_B:$Port_B$) to the iP2P socket API. Then, the iP2P socket API maps user B's MSISDN to its public transport address.

Step E.5 Because the corresponding mapping entry (IP_b:$Port_b \leftrightarrow IP_B$:$Port_B$) was created in NAT-B at Step E.3, App-A can establish the P2P connection with App-B by using the App-B public transport address (IP_B:$Port_B$).

Compared with the traditional UPnP approach, iP2P does not require the called peer (i.e., user B) to attach to the IP network before the iP2P connection is initiated. Also, no centralized server is required.

Conclusions

This article proposed an innovative P2P system called iP2P, which reuses the UMTS mobility management mechanism and SMS as the control protocol. We implemented the iP2P system on the Windows Mobile 6.0 Professional operating system. The measured data from 200 experiments indicates 21.724 seconds for the average connection set-up time, 2.637 seconds for the average SMS delivery time, and 9.428 seconds for the average WiFi network attachment time.

iP2P has several advantages over the existing P2P systems. First, by reusing the UMTS mobility management mechanism, iP2P provides efficient query without a requirement to implement its own centralized server. Because the UMTS network typically supports millions of users, there is no scalability problem in our approach. Second, iP2P utilizes the SMS push mechanism to establish P2P connections, which can significantly save power consumption in the mobile devices. Third, iP2P allows more flexible IP address allocation because the called peer is not required to attach to the IP network until it receives the SMS-based iP2P control message in the identification phase.

We also showed that iP2P can integrate easily with the existing NAT-traversal mechanisms to support mobile devices residing in the private IP networks. The performance of these mechanisms can be found in [11]. The mechanism of iP2P is pending U.S. and Republic of China (R.O.C.) patents.

Acknowledgment

This work was sponsored in part by NSC 97-2221-E-009-143-MY3, NSC 97-2219-E009-016, Far Eastone Telecom, Chung Hwa Telecom, ITRI/NCTU Joint Research Center, and MoE ATU.

References

[1] D. Bryan *et al.*, "Concepts and Terminology for Peer to Peer SIP," IETF Internet draft, Mar. 2007.
[2] Y.-B. Lin and I. Chlamtac, *Wireless and Mobile Network Architectures*, Wiley, 2001.
[3] Y.-B. Lin and A.-C. Pang, *Wireless and Mobile All-IP Networks*, Wiley, 2005.
[4] T. Pering *et al.*, "CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces," *Proc. 4th MobiSys*, 2006.
[5] J. Rosenberg, "Interactive Connectivity Establishment (ICE)," IETF Internet draft, Oct. 2007.
[6] S. Guha, Y. Taked, and P. Francis, "NUTSS: A SIP-Based Approach to UDP and TCP Network Connectivity," *Proc. SIGCOMM '04 Wksps.*, Aug. 2004, pp. 43–48.
[7] A. Biggadike *et al.*, "NATBlaster: Establishing TCP Connections between Hosts behind NATs," *Proc. ACM SIGCOMM ASIA Wksp.*, Apr. 2005.
[8] J. Rosenberg, "TCP Candidates with Interactive Connectivity Establishment (ICE)," IETF Internet draft, Nov. 2007.
[9] S. Guha and P. Francis, "Characterization and Measurement of TCP Traversal through NATs and Firewalls," *Proc. 2005 Internet Measurement Conf.*, Oct. 2005.
[10] UPnP Forum, "Internet Gateway Device (IGD) Standardized Device Control Protocol," v. 1.0, 2001.
[11] W.-E. Chen, Y.-L. Huang, and H.-C. Chao, "NAT Traversing Solutions for SIP Applications," *J. Wireless Commun. Net.*, 2008.

Biographies

CHIEN-CHUN HUANG-FU (jjfu@cs.nctu.edu.tw) received his B.S.CSIE and M.S.CSIE degrees from National Chiao Tung University (NCTU), Taiwan, in 1999 and 2001, respectively. He is currently a Ph.D. candidate in the Department of Computer Science, NCTU. His current research interests include peer-to-peer networks, design and analysis of personal communications services networks, and mobile ad hoc networks.

YI-BING LIN [F] (liny@csie.nctu.edu.tw) is Chair Professor and Dean of the College of Computer Science, NCTUn. His current research interests include mobile computing and cellular telecommunications services. He is the co-author of the books *Wireless and Mobile Network Architecture* (with Imrich Chlamtac; Wiley, 2001), *Wireless and Mobile All-IP Networks* (with Ai-Chun Pang; Wiley, 2005), and *Charging for Mobile All-IP Telecommunications* (with Sok-Ian Sou; Wiley, 2008). He is a Fellow of ACM, AAAS, and IET.

HERMAN RAO (hrao@fareastone.com.tw) has a B.S. in mechanical engineering from National Taiwan University and a doctorate in computer science from the University of Arizona. He is currently the executive vice president of the Network & Technology Division at Far EasTone Telecommunications Co., LTD, Taiwan. Prior to joining Far EasTone, he worked at Bell Labs as a senior researcher and at AWS as a research director for 10 years. In addition to extensive industry experience in telecommunication and data communication, he was an associate professor at National Tsing-Hua University and National Chung-Cheng University.