國立交通大學

資訊工程學系

博 士 論 文

NAT Inbound 連線, Web 快取, 與行動環境中的 Web 連線切換

NAT Inbound Session Establishment, Web Caching, and Web Session Handoff for Mobile Environments

研 究 生：謝明燈

指導教授：曾建超　教授

中華民國九十四年六月

NAT Inbound 連線, Web 快取, 與行動環境中的 Web 連線切換

NAT Inbound Session Establishment, Web Caching, and Web Session Handoff for Mobile Environments

研 究 生：謝明燈　　Student：Ming-Deng Hsieh

指導教授：曾建超　　Advisor：Dr. Chien-Chao Tseng

國 立 交 通 大 學
資 訊 工 程 學 系
博 士 論 文

A Dissertation Submitted to
the Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in

Computer Science and Information Engineering

June 2005

Hsin-Chu, Taiwan, Republic of China

中華民國九十四年六月

NAT Inbound 連線, Web 快取, 與行動環境中的 Web 連線切換

研究生：謝明燈　　　　　　　　　　　　　　　　　指導教授：曾建超

國立交通大學資訊工程學系博士班

摘　　　要

在本論文中，我們提出了可動態設定的 NAT (DCNAT)、混合式的 Web 快取管理演算法、Web 瀏覽連線換手機制、以及利用網路拓樸與移動方向來進行鍊接層快速換手的機制，以便利用這些機制來支援不間斷的 Web 存取，與推送即時資訊至私有網路內的行動主機。

DCNAT 是一個基於連接埠的 NAT 機制，它可以讓網際網路上的主機主動建立連線與推送即時資訊至 NAT 網路內的主機。DCNAT 包含了一個 Binding Entry Request 程序，藉由此程序，網際網路上的主機在連線至一私有網路內的主機之前，可以要求 DCNAT 路由器來動態地建立關於此一私有網路主機的 NAT binding entry。再者，動態建立 binding entry 的特色使得 DCNAT 能夠有彈性地支援私有網路主機動態開放的連接埠與網路服務之存取。此外，網際網路上的內容提供者也能夠藉由 DCNAT，主動地推送即時資訊給散佈於各個私有 NAT 網路的訂閱者。

為改善 Web 快取的效能，我們提出了混合式的 Web 快取管理演算法。此一混合式 Web 快取管理演算法將快取的儲存空間分成熱區與冷區共兩個區域，熱區採用簡單的 LRU 演算法，冷區採用較複雜的 GreedyDual 系列的演算法。這樣的混合式 Web 快取管理演算法，結合了 SLRU、LRU、GreedyDual 系列的優點，卻只須要在快取儲存空間的一部份(冷區)應用較複雜的 GreedyDual 系列演算法，便可以藉由冷區較周詳的管理演算法來提升快取的整體效能。因而混合式 Web 快取管理演算法可以達到略低於 GDSP 演算法的高文件命中率(DHR)與略低於 LRU 演算法的高位元組命中率(BHR)，但卻只需要

遠低於 GreedyDual 系列演算法的維護代價。

為了讓使用者在更換設備後，可以繼續 Web 瀏覽連線，我們提出一個 Web 瀏覽連線換手系統。此一 Web 瀏覽連線換手系統可以在同質或異質的使用者設備之間切換無狀態或有狀態的瀏覽連線，讓使用者在更換設備後可以進行不中斷且無接縫的 Web 存取。為了能夠在使用者設備之間追蹤與交換各式各樣的連線相關資訊，我們採用的是代理伺服器為基礎的方法，再加上非必須的使用者端的輔助機制。首先，使用者利用其設備向一個稱之為 UAP 的代理伺服器註冊其瀏覽連線；隨後 UAP 便可以追蹤此瀏覽連線並記錄必要的連線資訊，以供換手使用。此外，使用者端的輔助機制可以幫助 UAP 來換手更詳盡的連線資訊。相較於以使用者端為基礎的方法，我們的設計有許多優點，例如：使用者設備需要較少的修改、實用性、與錯誤相容。我們已實作了一 UAP 主機，並在個人電腦(PC)與個人數位助理(PDA)實作了使用者端的程式。實作的結果能夠成功地在個人電腦與個人數位助理換手一有狀態的線上購物瀏覽連線。

為加速在 802.11 無線基地臺之間進行鍊接層換手的過程，我們利用無線基地臺拓樸、行動主機的移動方向、與行動主機的位置來計算換手的候選無線基地臺，並減少候選無線基地臺的數目。此方法另有許多優點：第一，在換手的過程中，行動主機不需要執行耗時的無線基地臺搜尋程序，便能夠直接與無線基地臺取得關聯(association)，從而進一步加快換手的速度；第二，此一計算換手的候選無線基地臺的機制可以與其它現存的快速換手機制(例如：預先驗證與預先金鑰分配)互相配合。

這四個機制一起作用，為行動環境與 NAT 環境的使用者提供了方便的網際網路存取。

關鍵詞：網路位址轉譯、即時資訊推送、快取、瀏覽連線換手、鍊接層快速換手

NAT Inbound Session Establishment, Web Caching, and Web Session Handoff for Mobile Environments

Student：Hsieh Ming-Deng Advisor：Dr. Tseng Chien-Chao

Department of Computer Science and Information Engineering
National Chiao Tung University

ABSTRACT

In this thesis, we propose a Dynamically Configurable NAT (DCNAT), a hybrid web cache replacement algorithm, a web session handoff mechanism, and a Topology-and-Direction-aware link layer fast handoff scheme to support continuous web access and real-time information pushing for mobile hosts that may visit private networks.

With DCNAT, a host situates in Internet can spontaneously establish inbound session and push instant information to hosts beneath NAT networks. DCNAT is a port-based NAT scheme that adopts a Binding Entry Request procedure for an Internet host to request a DCNAT router to dynamically create binding entries, for address:port translation, associated with a private host behind the DCNAT router just before the Internet host originates a communication session to the private host. Furthermore, the dynamic creation of NAT binding entries makes DCNAT very flexible in supporting inbound accesses to the ports/services opened dynamically by the private nodes behind an NAT router. Besides, with DCNAT, a content service provider can push information contents spontaneously to subscribers that are widely spreading and beneath different private networks with NAT.

In order to improve web cache performance, we propose a hybrid web cache replacement algorithm that divides a cache space into two zones, a hot zone and a cold zone, and adopt a simple LRU replacement algorithm for the hot zone and a complex GD-Family replacement algorithm for the cold zone. This hybrid replacement algorithm combines the advantage of SLRU, LRU, and GD-Family algorithms but applies a complex GD-Family replacement algorithm only to a portion of the cache where strict object ordering is much significant to the overall cache performance. Therefore it can achieve a high DHR just slightly lower than that of GDSP and a high BHR just slightly lower than that of LRU while incurs much less maintenance cost, comparing with GD-Family algorithms.

To resume a web browsing session after a user changes the device that he uses to browse Internet, we propose a web session handoff system that can hand over not only stateless but also stateful sessions between homogenous or heterogeneous user devices to enable uninterrupted and seamless web accesses. The proposed system adopts a proxy-based approach and an optional client-assisted scheme to track and hand over a variety of session information from one device to another. In the proposed system, a user device needs to register a session with a User Agent Proxy (UAP) and the UAP then tracks the registered session and records the information necessary to the web session handoffs. In addition to session information tracked by a UAP, the UAP can hand over more comprehensive session information by using the client-assisted scheme. Compared with client-based approaches, our design has several advantages, such as less modification to user devices, practicability, and fault tolerance. We have implemented a UAP on a PC and client programs for both PC and PDA. The implementation can successfully hand over between PC and PDA a stateful session for online shopping applications.

To speed up the process of handoff between 802.11 APs, we propose a Topology-and-Direction-aware fast handoff scheme that resolves a reduced set of candidate APs according to AP topology, MN's its moving trend and/or MN's position. The proposed approach further benefits handoff by allowing 802.11 station reassociates directly with an AP without performing AP discovery during handoff. Besides, the topology-and-movement-aware candidate-AP generation scheme can work in conjunction with the existing fast-handoff schemes that leverage pre-authentication, proactive key distribution, or context transfer.

These four approaches together provide convenient Internet access in mobile environments and NAT environments.

Keywords: NAT inbound session, real-time information pushing, web cache, web session handoff, link layer fast handoff

Acknowledgements

在交大待了這麼多年，終於完成博士學業。首先要感謝我的指導教授—曾建超博士自大學部專題以來多年的指導，讓我能夠完成微處理器相關的碩士論文與與網路相關的博士論文。感謝口試委員：郭耀煌教授、竇其仁教授、周勝鄰組長、王讚彬教授、張明峰教授、曹孝櫟教授細心審查我的論文，並提供寶貴之意見。另外，也感謝繼光輝學長、張鴻駿學弟、蔡清松學弟在研究與論文撰寫的幫助。再感謝多位親切的系辦公室助理小姐多年來在公事上的幫忙。最後，感謝我的家人、實驗室的學弟妹、攝影社的夥伴、與橋藝社的夥伴陪我渡過多年的憂愁與快樂。

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation and goal

In recent years, advances in wireless technologies and capability of mobile devices have made mobile Internet practically feasible. In mobile Internet environments, a person can use a mobile device to access Internet and may switch to another device. Besides, he may move into an NAT network, e.g., a GPRS network or an 802.11 family wireless network that provides a private IP address. Such mobile Internet access challenges researchers to provide mobility support to information acquisition applications and enable inbound session to NAT networks [1]. In the literature, there is much research on terminal mobility or personal mobility, but little on session mobility [2]. Providing session mobility support to applications enables handing over an application session between user devices and therefore enables a user to resume his application after he changes device. Besides, solve NAT inbound session enables information pushing to a user moving into an NAT network. Pushing without a user's probing is a convenient way different from traditional ones, e.g., email or SMS, to delivery real-time information content, e.g., stock information and surveillance alerts, to the user regardless the user's location.

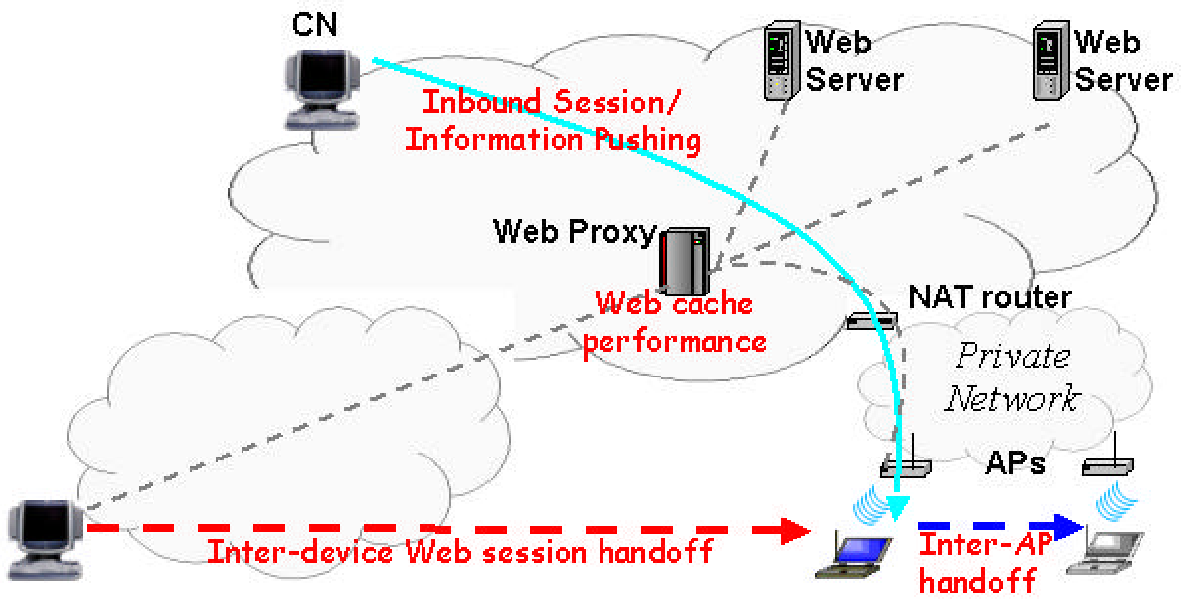


Figure 1-1. Problems overview

Among current information acquisition applications in Internet, web browsing and email are the most important two. However, e-mail is not suitable for real-time information delivery. Therefore this thesis, from the viewpoint of applications, aims at improving web browsing and real-time information delivery in mobile environments. Figure 1-1 illustrates four problems that a person may encounter in mobile environments. These four problems are described as follows:

1. **NAT Inbound session establishment and information pushing to NAT private networks**: Inbound session problem is a notorious problem in NAT environments that prevents an Internet host from initiating a session to a host in an NAT private network. In mobile environments, there will be many NAT private networks since most GPRS and 802.11 family wireless networks provide private IP address to users and use NAT to connect to Internet. Therefore, a mobile node (MN) will probably move into an NAT network and encounter the inbound session problem. Besides, the inbound session problem also impedes Internet content or information servers from pushing real-time information, such as surveillance alerts or stock information, to the hosts in NAT private networks. Pushing information is better than traditional ways such as those use email or SMS since information subscribers need not probe for non-periodically produced information content. In mobile environments, many users that subscribe to the same information content service may be spread over several NAT private networks operated by different ISPs. The information server thus needs to deliver information actively to each NAT private network where one or more subscribers reside. However it is currently impossible for the information servers to push information content to private hosts behind legacy NAT routers.
2. **Web cache performance of web proxy**: Web browsing is an important Internet application that a person would perform in daily Internet access. In current Internet environments, web caching is widely used to reduce network traffic, server load, user-perceived retrieval delay, etc. Previous web cache replacement algorithms may trade document hit ratio with byte hit ratio or require high maintenance cost to achieve

both high document hit ratio and high byte hit ratio. Therefore, we aim to achieve high document hit ratio and high byte hit ratio with less maintenance cost.

3. **Web session handoff**: Web session handoff between user devices provides much convenience. It enables a user to resume browsing the last page that he viewed on a first device without traversing again the hierarchy of a web site's content after he changes to a second device. Besides, handing over history of a web session enables a user to review the pages that he just viewed on the first device. Furthermore, handing over inputted values in HTML form fields exempts users from tedious form refilling. Due to such convenience, web session handoff is useful in real life scenarios. For example, one may use a PC in his office to start browsing for preparing a suddenly-notified meeting and then can hand over his browsing session to a portable computer (a homogeneous device) or a PDA (a heterogeneous device) before he leaves his office or when he moves into a meeting room. Moreover, a session initially on a capability-limited mobile device can be handed over to a more powerful stationary device for purpose of the user's convenience. For example, one may initially view and process stock information with a PDA or a smartphone and then hand over this session to a PC when a PC is available. In the literature, session handoff has been envisioned in first aid and clinical domains [3]. As mobile Internet access becomes popular, there will be more useful and diverse scenarios about web session handoff in the future. Therefore, users need an approach that can hand over session information between user devices rather than re-initiate a session from a second device.

4. **Fast inter-AP handoff**: In mobile environments, there will be 802.11 wireless networks consisting of a lot of 802.11 access points (AP). Inter-AP handoff occurs when an MN moves between the coverage of two wireless access points. A fast inter-AP handoff scheme is therefore required so that a mobile node has sufficient time to perform handoff above link layers, e.g., Mobile-IP (at network layer), SIP (at application layer), and Web (at application layer).

In this thesis, we respectively propose an approach to each of these problems. For the inbound session problem, we propose a DCNAT approach that enables inbound session to dynamically opened ports/services by hosts in NAT private networks. Besides, we extend DCNAT to solve the problem of information pushing to NAT private networks. For web cache performance, we propose a hybrid web cache replacement algorithm to achieve high document hit ratio and high byte hit ratio while spending less maintenance cost. For the web session handoff problem, we propose a proxy-based approach and optional client-assisted scheme to hand over web session information between user devices. For inter-AP handoff problem, we propose a Topology-and-Direction-aware fast handoff scheme. These approaches would jointly contribute to convenient mobile Internet access.

1.2 Thesis organization

The remainder of this paper is organized as follows. Chapter 2 gives detailed description and causes of the problems solved in the thesis. Chapter 3 discusses previous related work on the four problems. Chapter 4 proposes DCNAT, our approach to the NAT inbound session problem and extends DCNAT to fit for the information pushing application. Chapter 5 elaborates our hybrid web cache replacement algorithm. Chapter 6 elaborates our web session handoff system, addresses issues of session tracking in detail, and discusses ways to track and hand over sessions. Chapter 7 elaborates our Topology-and-Direction-aware fast handoff approach. Chapter 8 gives conclusions and future work.

Chapter 2 Background

2.1 Inbound session problem in NAT environments

2.1.1 NAT

Network Address Translation (NAT) [1] is a technique that can translate one IP address to another while providing transparent routing to hosts. It is widely deployed by many Internet services providers (ISP) to solve the IP address depletion problem of IPv4. In an NAT[1] network that uses private IP addresses, there will be an NAT router with two networks interface, an internal interface to the NAT network and an external interface to the public network. The internal interface of the NAT router and hosts in the NAT network (henceforth referred to as internal host) use Internet-non-routable private IP addresses to communicate with each other. On the other hand, internal hosts communicate with hosts external to the NAT network (henceforth referred to as external host) via the external interface of the NAT router, which uses public IP address(s). When packets pass through the NAT router, the NAT router transparently translates an Internet-non-routable private IP addresses used by internal hosts to an Internet-routable public IP address of the external interface. Therefore, a private IP address can be reused by another NAT network and the IP address depletion problem can be solved.

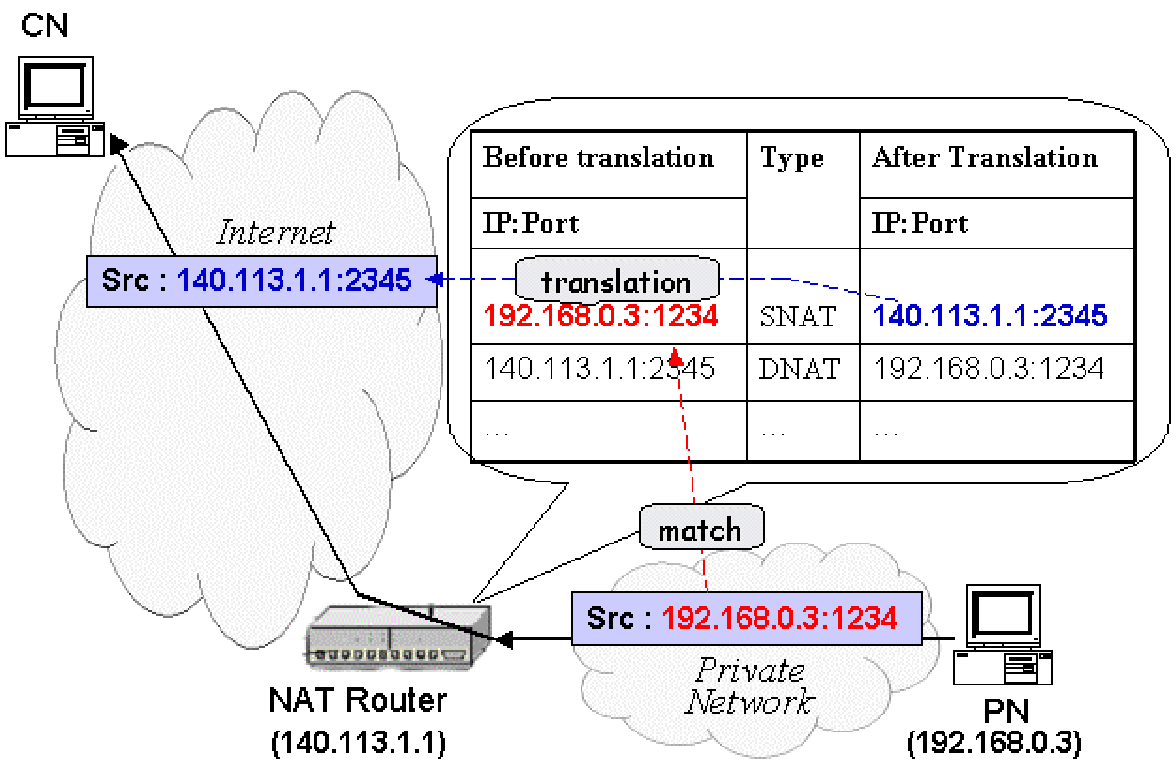
In the following, we first explain two types of traditional NAT: Basic NAT and NAPT (Network Address and Port Translation).

Basic NAT: With Basic NAT, the NAT router reserves a pool of public IP addresses for address translation. When an internal host originates a session to an external host, the NAT router selects an address from the reserved IP addresses and creates an address binding between the private address of the internal host and the selected public address. Afterward, packets of the session can be translated and routed according to the created

[1] Although NAT is restricted to translation between a private address realm and a public address realm, in this thesis we focus only on using NAT to translate between a private address realm and a public address realm. Therefore, we will use NAT network, private network, and NAT private network in interchange throughout this thesis..

address binding. For packets outbound from the NAT network, the NAT router changes the source IP addresses from private addresses to public ones; for inbound packets, the NAT router replaces their destination IP address in reverse from public addresses to private ones.

NAPT: NAPT extends the translation further by also translating the transport identifier, e.g., TCP and UDP port numbers, or ICMP query identifiers. With this extension, transport identifiers of a number of private hosts are multiplexed into transport identifiers of a single public address. Thus NAPT allows multiple hosts to share a single public address for saving public addresses. Therefore most current NAT routers adopt NAPT for address translation.



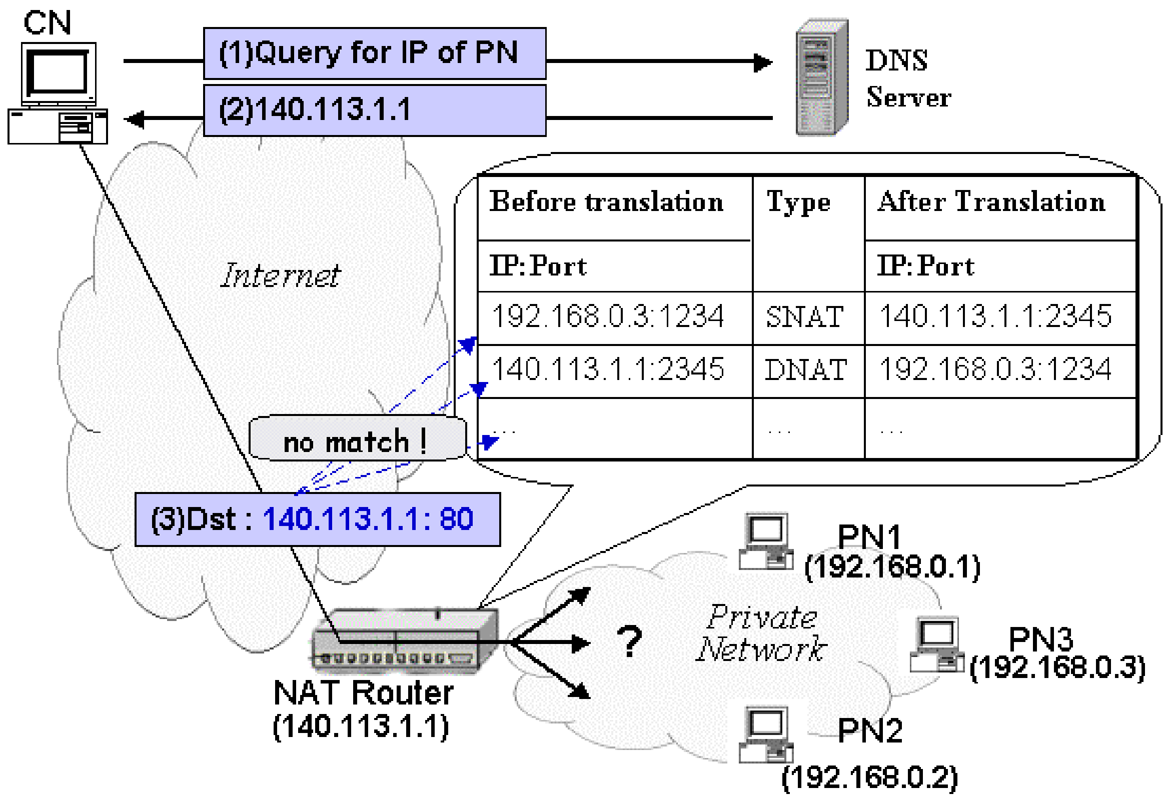
Before translation	Type	After Translation
IP:Port		IP:Port
192.168.0.3:1234	SNAT	140.113.1.1:2345
140.113.1.1:2345	DNAT	192.168.0.3:1234
…	…	…

Figure 2-1. NAPT operation

There are many variations of NAT that lend themselves to different applications. IETF has defined terminology, consideration, and variations of NAT in RFC 2663 [4] and RFC3489 [5]. In reality, each vendor may implement NAT in its own way. For example, Iptables [6], an implementation of NAPT in Linux operating system, classifies binding entries into two categories: Source NAT (SNAT) and Destination NAT (DNAT). In Iptables, an NAT router checks SNAT entries for each outbound IP packet and DNAT entries for each inbound IP packet. Figure 2-1 illustrates the NAPT operations performed by an NAT router for SNAT translation. A host (PN) in a private network with private IP (192.168.0.3) is communicating with another host (CN) in the Internet. PN sends outbound packets with IP:port as 192.168.0.3:1234 which are routed to the NAT Router. The NAT Router then finds that the source IP:port of the received packet matches an SNAT entry in the NAT table, and therefore translates the packet's source IP: port into 140.113.1.1:2345 according to the matched entry. After the translation, the translated IP packets are routed to the Internet according to the routing table of the NAT router. Afterward, when the CN receives translated packets, it can reply PN by simply using the source IP:port of the received translated packets as the destination IP:port of the replied packets. Upon receiving inbound packets from CN, the NAT router knows, by examining DNAT entries, how to replace the destination IP:port with an internal IP:port for the packets. Consequently, PN receives the translated inbound packets destined for PN's internal private IP from the specified port on which PN is expecting to receive the inbound packets.

Besides, RFC3489 [5] further classifies NAT as full cone, restricted cone, port-restricted cone, and symmetric ones. In **full cone NAT**, all packets from the same internal IP address and port are mapped to the same external IP address and port. Furthermore, any external host can then send a packet to the internal host, by sending a packet to the mapped external address and/or port. In **restricted cone NAT**, all packets from the same internal IP address and port are mapped to the same external IP address and port. Unlike a full cone NAT, an external host (with IP address X) can send then a packet to the internal host only if the internal host had previously sent a packet to IP address X. As for **port restricted cone NAT**, it is like a restricted cone NAT, but the restriction includes port numbers. Specifically, an external host can send a packet, with source IP address X and source port P, to the internal host only if the internal host had previously sent a packet to IP address X and port P. In **symmetric NAT**, all requests from the same internal IP address and port, to a specific destination IP address and a specific port, are mapped to the same external IP address and port. If the same internal host sends a packet with the same source address and port, but to a different destination, a different mapping is used. Furthermore, only the external host that receives a packet from a specific port on a specific internal host can then send a packet back to the specific port on the specific internal host.

2.1.2 Inbound session problem



Before translation	Type	After Translation
IP:Port		IP:Port
192.168.0.3:1234	SNAT	140.113.1.1:2345
140.113.1.1:2345	DNAT	192.168.0.3:1234
…	…	…

Figure 2-2. Inbound session problem

However, an external host cannot originate a session to an internal host behind an NAT router unless the NAT router has established NAT binding entry(s) for the internal host. Figure 2-2 illustrates the **inbound session problem**. Assume that there is no binding entry for port 80 (used by HTTP service) on the NAT router's external IP address and an Internet host (CN) wishes to initiate an HTTP session with PN1, a host in the NAT network. CN should first issue a DNS [7-8] request to learn PN1's IP address. The DNS request is then replied by a DNS server with the external IP address of PN1's NAT router. Afterward, CN sends a connection request to port 80 on the NAT router's external IP address. However, when the

NAT router receives the connection request packet, it would find no match in the NAT table and it won't know which host in the private network is the desired destination host.

2.1.3 Information pushing service

Among current mobile Internet connectivity techniques, GPRS and 802.11 family wireless networks are most popularly adopted. GPRS and 802.11 family are superior to traditional dial-up Internet access in providing inexpensive and fast continuous Internet connectivity. With continuous Internet connectivity, mobile devices are kept online so that the mobile devices can be reached by other nodes even the mobile devices have no ongoing connections with any other nodes. Therefore, contents about real-time information, e.g. stock information and surveillance alerts, can be pushed to subscribers without subscribers' probing. Delivering information by pushing is better than traditional ways, e.g., email or SMS, since subscribers will not need to probe for non-periodically produced information. However, it is difficult to push information to mobile users in GPRS networks and 802.11 family wireless networks since many of these networks provide private IP addresses to mobile users and utilize NAT to connect to the Internet.

2.1.4 Our approach -- DCNAT

Many researchers have proposed approaches to resolve the inbound session problem. However, these approaches suffer from wasting IP addresses, being unable to support dynamically open ports by PN, being unable to support lots of PNs, or lack of inflexibility. Therefore, we propose a dynamically configurable NAT (DCNAT) approach to solve the inbound session problem as well as the problem of information pushing to NAT networks. The proposed DCNAT approach uses an add-on NAT Agent on an NAT router and a Binding Entry Request procedure so that an Internet host or its agent can request the NAT router to create dynamic NAT binding entries. After creation of binding entries, packets sent from the Internet host can pass through the NAT router and reach their desired destination host. DCNAT can support general inbound sessions as well as information pushing to a private network. Compared with previous proposals, DCNAT is more scalable and flexible because it can support inbound accesses to lots private hosts and ports/services opened dynamically by private hosts by using just a single or few IP addresses.

2.2 Web caching

Caching has been used for decades as an effective performance enhancing technique in computer systems. As World Wide Web applications becomes popular and contribute to much Internet traffic, web caching is widely used to reduce network traffic, server load, user-perceived retrieval delay, etc. Though the ever decreasing prices of RAM and disks can provide large cache space to achieve higher hit ratio, the optimization or fine tuning of cache replacement is still significant since previous studies have shown that web cache hit ratio grows in a log-like fashion as function of cache size [9-12]. Thus, a better algorithm that increases hit ratios by only several percentage points would be equivalent to increasing cache size by several times. Besides, the growth rate of web content is much higher than that of cache size. The only way to bridge this widening gap is through efficient cache management. Moreover, the benefit of even a slight improvement in cache performance may have an appreciable effect on network traffic, especially when such gains are compounded through a hierarchy of caches.

Many traditional cache replacement algorithms have also been used in web cache replacement, e.g., Least-Recently Used (LRU) and Least-Frequently Used (LFU). However, web caching differs substantially from the traditional ones. Two main features of the web caching that differ from the traditional caching are (1) non-uniformity of the object sizes and (2) non-uniformity of the cost of cache misses (as opposed to the traditional caching where all cache blocks have the same size and require the same amount of time to be retrieved in case of cache misses). These features have led researchers to propose web cache replacement algorithms that take these features into consideration, e.g. GreedyDual and its variants [11,13-14].

Traditional web caching algorithms such as LRU does not perform well in terms of document hit ratio (DHR). On the other hand, GreedyDual and its variants, spending higher maintenance cost, basically perform well in terms of document hit ratio but not well in terms of byte hit ratio (BHR). Although there are algorithms dealing with multiple and complex parameters that may perform well both in terms of DHR and BHR, these algorithms require further higher maintenance cost. In mobile environments, each proxy situated at different network and hierarchy may favor high byte document hit ratio or high byte hit ratio according to bandwidth, traffic, and other network characteristics. Therefore, we propose a hybrid web cache replacement algorithm to improve DHR of web caching and retain high BHR while spending less maintenance cost.

2.3 Web session handoff

Web session handoff intrinsically means handoff of session information maintained by a browser program of a first user device to a browser program of a second device. (Note that the first device is hereafter referred to as *source device* since it is the source providing session information, while the second device is referred to as *target device* since it is the target accepting session information.) The so-called session information consists of any information about a session such as session history, HTTP cookies, web objects maintained by browsers, etc. If complete session information of a session can be handed over, a user will be able to continue browsing seamlessly as if he had not changed to a second device. Therefore, a web session handoff system should be able to track and obtain as much as session information from source devices so that session information can be handed over as complete as possible and a session can be more seamlessly resumed on the target device.

HTTP [15], the basic communication protocol of web browsing, is originally designed to support stateless sessions. However, as web browsing become popular and is widely used for more aspect of information exchange, it is important and necessary to support stateful web sessions [16-17], in which session state persists across multiple HTTP transactions. For example, an online shopping web site should differentiate sessions initiated by different clients and maintain the shopping cart of each client as separate session state. Therefore, many approaches [18-19] have been proposed and extensively used by web servers to enable stateful web sessions. In order to hand over not only stateless sessions but also stateful sessions, a web session handoff protocol/system should be able to track and hand over session state between users devices. For example, BSR [20] adopts a client-based approach that uses a substantially modified browser to save and hand over browser state to/from a BSR repository. However, using client-based approaches to hand over web sessions is not desirable since they require substantial modification to browsers. In client-based approaches, each browser program for various types of user devices needs its own modification to program code. Besides, the inefficient browser-dependent modification is impractical and usually has incompatibility problem when inter-working with unmodified web proxies or web servers.

In this thesis, we propose a web session handoff system that adopts a proxy-based approach and an optional client-assisted scheme in order to track and hand over session information. In the proposed system, we introduce the functionalities of web session tracking and handoff into traditional web proxies to provide a special purpose proxy, called UAP. The UAP can track a session while the user browsing and then can hand over session information to another device. In addition to session information tracked by a UAP, the UAP can hand over more comprehensive session information by using the client-assisted scheme. As a result, the proposed web session handoff system requires less modification to user devices while being able to hand over much of session information. In our experience, proxy-based approaches are practical for web session handoff before a standardized or popular web session handoff protocol is widely recognized.

2.4 Fast inter-AP handoff

Critical to IEEE 802.11 network machinery is the handoff process, which occurs when a mobile node (MN) moves its association from one access point (AP) to another. A standard 802.11 handoff consists of AP discovery, legacy authentication, and reassociation procedures.

Among these procedures, AP discovery by an MN's active scan imposes a significant delay [21]. In active scan [22], an MN repeats the following operations for each channel:

1. The MN switches antenna to the channel and waits for either an indication of an incoming frame or for the *ProbeDelay* timer to expire. If an incoming frame on the channel is detected, the MN knows the channel is in use and can be probed.
2. The MN gains access to the medium using the basic DCF access procedure and sends a *Probe Request* frame.
3. The MN waits for the minimum channel time, *MinChannelTime*, to elapse.
    a. If the medium was never busy, the MN knows there is no network and therefore tries another channel.
    b. If the medium was busy during the *MinChannelTime* interval, the MN waits until the maximum time, *MaxChannelTime*, and processes any *Probe Response* frames sent from APs on the channel.

After all the channels have been tried, the MN selects a target AP, usually the AP with the best received signal strength, from the APs that have responded the MN's probe. Since the number of channels in 802.11 family is at least 11, this AP discovery procedure can take up at least 250 ms. Furthermore, if a security scheme stronger [23-24] than legacy authentication such as open system authentication or WEP [22] was adopted, the time for performing authentication will increase. For example, the IEEE 802.1X [25] framework has been adopted as mandatory part of emerging Robust Security Networks [26] in additional to authentication and cryptographic key management. Given IEEE 802.1X transactions at remote sites, internetwork transmission account for another potentially prohibitive handoff delay.

As a remedy, some researchers have proposed schemes to pre-authenticate MN with APs [27] or proactively distribute key material to neighboring APs [28]. With these schemes, the target AP, upon detecting reassociation, can complete handoff sooner than would otherwise be possible. However, such pre-authentication or key distribution involves the APs in the vicinity of the MN according to some mobility profiles, regardless of the MN's movement, topology information, and whereabouts. As a consequence, a number of APs per MN, e.g., those opposite to the moving direction, are excessively involved. Although another efficient method for AP discovery was developed in [29], these schemes may still suffer from AP discovery.

This thesis proposes a Topology-and-Direction-aware approach to moderating the handoff overhead further in IEEE 802.11 networks. We allow an MN to learn of APs toward which it will roam shortly using its location, moving trend/direction, and the topology information about APs and environment objects. The topology information is provided by some designated server which also provides parameters for AP (re)association so that the MN need not learn these parameters without a probe beforehand. In light of the moving trend or direction, the MN, being able to determine its moving trend and/or position, selects a reduced set of neighbor APs to which pre-authentication messages or security information can be delivered in aforementioned fast handoff schemes. Besides, the MN can in advance select a target AP from the candidate AP set with which the MN will directly reassociate and therefore the time-consuming AP discovery is avoided upon handoff.

Chapter 3 Related Work

3.1 Related work on NAT inbound session problem

Many researchers have attempted to solve the inbound session problem. In this chapter, we briefly describe the mechanisms that have previously been proposed in literature.

3.1.1 Bi-directional NAT

Bi-directional NAT [4] attempts to solve the inbound session problem by adopting an address binding procedure similar to that for the outbound session in Basic NAT. In Bi-directional NAT, an NAT router would reserve a pool of public IP addresses and there is a DNS Application Layer Gateway (DNS_ALG [30]) co-situated with the NAT router. Referring back to Figure 2-2, when an Internet host (*CN*) issues a DNS request to learn the IP address of the designated node (*PN*) and the DNS_ALG receives the DNS request, the DND_ALG will select an unused public IP address of *NAT_R* (named *IP_A*), create NAT binding entries for PN, and inform CN of the selected public address with a DNS reply message.

Bi-directional NAT is a transparent approach since no modification to hosts in public networks or private networks is required. But a drawback of Bi-directional NAT is that an extra public address will be needed for each internal private host that has connection(s) with external hosts on Internet. This drawback conflicts with NAT's original goal in mitigating the IP Address Depletion Problem.

3.1.2 Expanded NAT

Expanded NAT [31] tackles the Inbound Session Problem by pre-configuring NAT binding entries to enable inbound accesses to some pre-defined services on internal hosts. Therefore only the services that have already been defined on specific hosts can be accessed by external hosts in Internet. Moreover, Expanded NAT has a Port Collision Problem since it can map a specific well-known service port, such port 21 for ftp, only to a particular internal host. No two distinct internal hosts can offer the same service using the same port on an NAT router.

3.1.3 Tunnel Approach

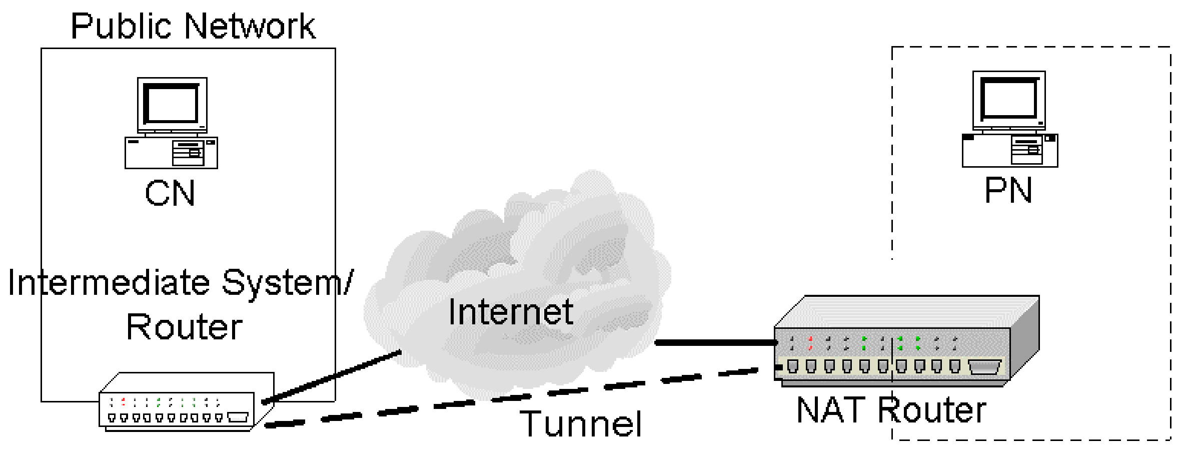


Figure 3-1. System architecture of Tunnel Approach

Figure 3-1 illustrates the system architecture of the Tunnel Approach [32]. The Tunnel Approach assumes that an intermediate system, which is situated on the route to/from CN to/from the Internet, can intercept packets originating from or designated to *CN*, and an IP-in-IP tunnel has been established between the Intermediate System and the NAT router of PN. Before *CN* initiates a communication session with *PN*, it issues a DNS request to inquire *PN's* IP address. This DNS request will be intercepted by the Intermediate System. Since the DNS query is for a host PN within a private network that is reachable through the pre-configured tunnel, the Intermediate System chooses an IP address (*IP_External*) as the external IP address of PN seen by CN, and informs the *NAT router* to establish binding entries between *IP_External* and *PN's* private IP. It should be noted that the Intermediate System could use private IPs for IP_External addresses. After establishing NAT binding entries, the intermediate system returns *IP_External* as PN's IP in a DNS reply to *CN*. *CN* can then use *IP_External* as the destination address for the packets to PN. The packets that are sent to *IP_External* will be intercepted by the Intermediate System. The Intermediate System will encapsulate the packets and send the encapsulated packets to the NAT router through the IP-in-IP tunnel. When the NAT router receives the encapsulated packets, it de-capsulates the packets, performs address translation from *IP_External* to PN's internal IP, and routes the de-capsulated inbound packets to PN.

Although the Tunnel Approach requires no modifications to hosts in both public and private networks, IP-in-IP tunnels must have been established between the Intermediate System *en route* and the intended private network. It is clearly impractical to have an Intermediate System *en route*, and establish a tunnel between the Intermediate System and each possible private network beforehand.

3.1.4 NAT binding inquiry

In contrast to Bi-directional NAT, Expanded NAT, and Tunnel Approach, which are transparent approach to PNs, many non-transparent approaches have been proposed, e.g. UPnP IGD[33] and STUN[34]. These non-transparent approaches rely on PN, aware of NAT, actively sending some messages to acquire the mapped public IP:port of PN and informing CN of the mapped IP:port. These schemes may be appropriate for applications that can learn a CN's IP:port from a centralized server or cooperatively distributed servers, e.g., SIP-based VoIP. However they are not well-suited for the real-time content pushing services since PN can not foresee when a real-time push server is ready to push messages to PN, and inquire PN's own mapped IP:port beforehand. Furthermore, in these schemes, PN will occupy a port on an NAT router even if there are no ongoing sessions, which may reduce the available number of ports on an NAT router when there are many PNs in the private network.

3.2 Related work on web caching

Many traditional cache replacement algorithms have also been used in web cache replacement, e.g., Least-Recently Used (LRU) and Least-Frequently Used (LFU). However, web caching differs substantially from the traditional ones. Two main features of the web caching that differ from the traditional caching are (1) non-uniformity of the object sizes and (2) non-uniformity of the cost of cache misses (as opposed to the traditional caching where all cache blocks have the same size and require the same amount of time to be retrieved in case of cache misses). These features have led researchers to propose web cache replacement algorithm that take these features into consideration, e.g. GreedyDual and its variants [11,13-14].

3.2.1 Performance metrics

Different performance evaluation measurements will produce different competition results. There are three categories of commonly used performance metrics for evaluating performance of web caching:

- **Document Hit Ratio (DHR)** refers to the percentage of hits in terms of the request times. **Byte Hit Ratio (BHR)** refers to the percentage of hits in terms of byte requested. There is a tradeoff between these two metrics. For example, an algorithm that prefers to keep 10 small objects probably achieves higher DHR than the algorithm that keeps only one objects of the same total size.
- **Request Response Time**: Real world tests found that response time is not as stable as being measured in the synthetic environments [11,35]. Though it is not appropriate to be an algorithm metric, the response time is still a dependable metric to measure performance improvement.
- **Outbound Network Traffic**: All misses will cause outbound network traffic for retrieving data from original web servers or other web caches. Unlike response time, the outbound traffic is a static metric that in most case equal to the object size.

Among these performance metrics, DHR and BHR are the most basic and are the most popularly adopted since other metrics are usually affected by DHR or BHR. For example, request response time is affected by DHR since web cache can responds on document hit, without requesting documents from original web servers or other web caches, faster than on document miss. Outbound network traffic is affected by DHR and BHR since more document misses or more byte misses cause a web cache requesting more from original web servers or other cache proxies.

3.2.2 LRU and SLRU

LRU, in principle of capturing temporal locality in web traffic, ranks objects according to each object's last reference time. The existing of temporal locality in web traffic has been proven by much analysis on web traffic. For example, [10] shows that the probability of an object's inter-access time is inversely proportional to the object's inter-access time. Figure 3-2 illustrates logical flow of LRU. In LRU, an object is inserted at cache head on cache miss, pushed toward cache tail by other requested objects, promoted to cache head if requested again, and removed when it reaches cache tail. Such operations are simple and of a maintenance cost of constant order. Due to its simplicity, LRU becomes the most widely used replacement algorithm in popular web cache systems, e.g., Squid [35].

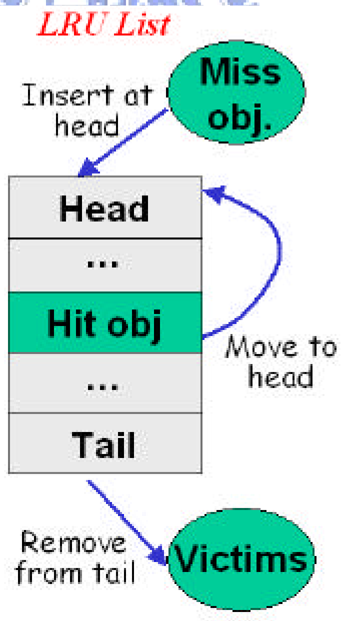


Figure 3-2. Logical flow of LRU

Since pushing an object from cache head to cache tail needs long time, inserting an object at cache head may be unnecessary before the cache knows the object is popular or not. Therefore, Rassul Ayani *et al* proposed Segmented LRU (SLRU) [36] in which a missed object is fetched not into cache head. Figure 3-3 illustrates logical flow of SLRU. SLRU divides cache space into two segments, each of which operates according to LRU. A missed object is fetched into the head of the lower segment and becomes the most recently used (MRU) object of the lower segment. An object in the lower segment is promoted to the head of the upper segment after several times of hit and becomes the MRU object of the upper segment. Victims of the upper segment are moved to the head of the lower segment and victims of the lower segment are removed from cache. The evaluation results in [36] shows that SLRU improves LRU in terms of both DHR and BHR.

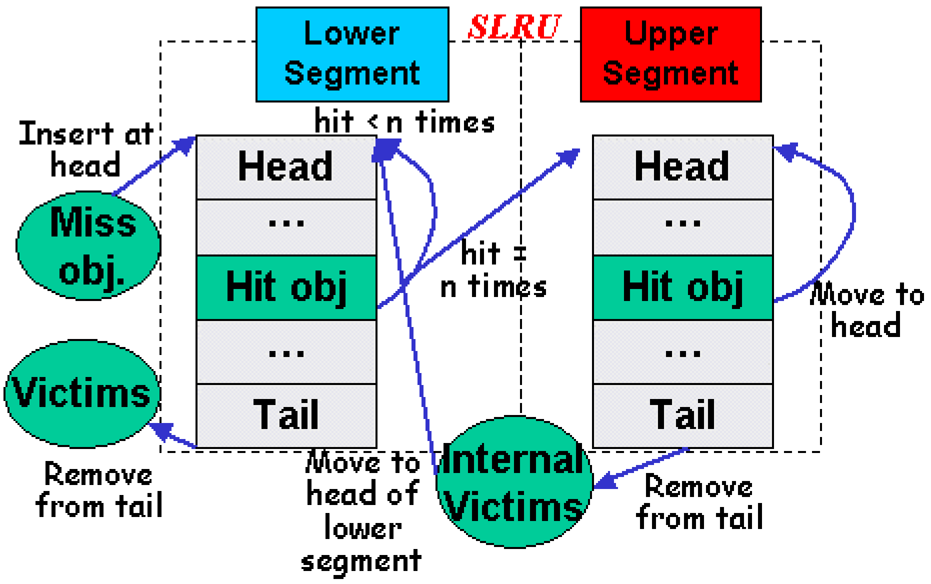
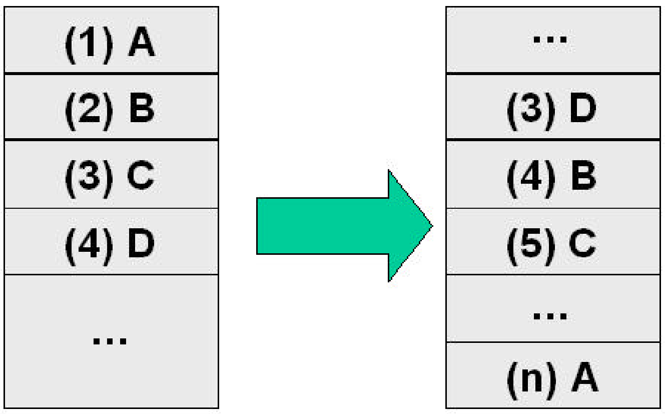


Figure 3-3. Logical flow of SLRU

The simple recency-based strategy of LRU works well for short-term popularity. Observe that a recently hot object is very probably requested again and therefore should not be removed soon. Besides, the ranking of a hot object relative to other objects will change drastically since the object is very probably requested again and re-ranked. Figure 3-4 shows an example in which relative ranking of the four objects: A, B, C, D at cache head when *time1* (Figure 3-4(a)) changes drastically when any of the four objects reaches cache tail when *time2* (Figure 3-4(b)). Therefore, it is unnecessary for a cache to spend lots efforts to maintain strong short-term popularity ranking for popular objects. From this point of view, LRU manages short-term popularity in a right way.



(a) Ranking at *time1* (b) Ranking at *time2*

Figure 3-4. Long-term behavior of objects

On the other hand, the simple recency-based strategy of LRU does not suit for maintaining long-term popularity. Observe that object's recency, a probable indication of the object's short-term popularity, does not have strong relationship between the object's long-term popularity. For example, a pure LRU would remove the least-recently-used object, which has been requested 1,000 times, even though the secondly least-recently-used object has been requested only once. From the viewpoint of probability, a cache, except having very small space, had better remove objects according to long-term popularity rather than short-term popularity. From this point of view, LRU is not proper for determining victims of a cache.

3.2.3 GreedyDual and its variants

GreedyDual (GD) algorithm [37] is originally proposed to deal with variable-cost uniform-size page caching problem. Cao and Irani extend GD into GreedyDual-Size (GDS) [11] to deal with variable-size web objects. Arlitt *et al* further proposed GDSF [38] to take object frequency into consideration. According to GDSF, Jin and Bestavros proposed GDSP (Popularity-Aware GreedyDual-Size) [14], which maintains object frequency as popularity profile. Besides, there are numerous algorithms [39-40] varied from GD. In [13], Jin and Bestavros summarize some variants into GreedyDual* (GD*). In GreedyDual*, each object *p* is associated a utilization value, $H(p) = L + u(p)^{1/b}$, where L is the global inflation value representing recency, $u(p) = (\frac{f(p) \times c(p)}{s(p)})$ is evaluated according to p's access frequency function $f(p)$, retrieval cost function $c(p)$, and size function $s(p)$, and b is the short-term temporal correlation factor $(0 \le b \le 1)$. On retrieval or on a hit of an object p, $H(p)$ is re-evaluated. On eviction of an object p, L is set to the $H(p)$. In general, each variant of GD is equivalent to GD* with specially defined $f(p)$, $c(p)$, $s(p)$, and b. Besides, some variants of GD further use runtime statistics to perform probability estimation and may dynamically adjust the key function $H(p)$ or function parameters, e.g. CERA [39]. All variants of GD use the following algorithm:

```
$L = 0$
For each request for object $p$ do
    If $p$ is in cache then $H(p) = L + u(p)^{1/b}$
    else fetch $p$
        While there is not enough free space for $p$
            do $L = min\{ H(q) \mid$ all objects $q$ in cache$\}$
                Evict the $q$ with minimum $H(q)$
        $H(p) = L + u(p)^{1/b}$
```

According to the above definition of utilization function and the above algorithm, most variants of GD favor objects of small size than large objects. Therefore GD and its variants can significantly outperform LRU in terms of DHR. As for the comparison between BHR of GD-Family and LRU, previous research has shown different results. Generally speaking, a GD variant with very high DHR often has lower BHR than LRU while a GD variant with proper parameters or utilization function can trade off some DHR for outperforming LRU in term of BHR.

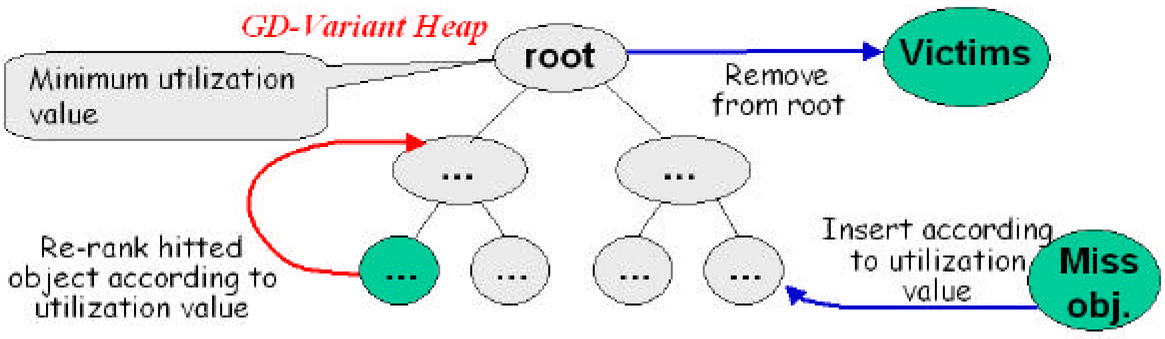

GD-Variant Heap
root
Minimum utilization value
Victims
Remove from root
...
Re-rank hitted object according to utilization value
Insert according to utilization value
Miss obj.

Figure 3-5. Logical flow of GD-Family

Since GD-Family evict the object with minimum utilization value, GD-Family often use a priority queue or a min heap, whose maintenance complexity is $O(\log n)$, to rank objects according to each object's utilization value. Figure 3-5 illustrates the logical flow of GD-Family. In comparison with LRU, the comprehensive strategy used by GD-Family is better for maintaining long-term popularity of objects and determining victims. For example, when selecting a victim from a least-recently-used object that have been accessed for 1,000 times and a secondly least-recently-used object that have been accessed once, GD-Family would comprehensively select the secondly least-recently-used object while LRU would select the least-recently-used object. However, the comprehensive strategy used by GD-Family is not unnecessary for maintaining short-term popularity as described in Section 3.2.2. Furthermore, GD variants that dynamically adjust the utilization function or function parameters may incur further higher maintenance cost if the adjustment requires global re-evaluation of utilization values of all objects. Therefore, GD variants are not widely used in popular web cache system due to higher complexity although they outperform LRU.

3.3 Related work on session mobility and web session handoff

3.3.1 Previous work on session mobility

In the literature, much research on mobility focus on terminal mobility, e.g., Mobile IP [41] and its variants [42]. Mobile IP and many of its variants deal with delivering packets via the home agent and foreign agents to a mobile terminal. In [42], by looking-up the location of a mobile terminal and directly establishing a connection to the mobile terminal, packets can be delivered by a correspondent node without passing through the home agent or foreign agents of the mobile terminal. On the other hand, personal mobility becomes more and more important [43-49] because it realizes a convenient scenario that allows a user to use various devices and to switch between them. Netchaser [43] uses agent programs residing on user devices and servers to achieve personal mobility. MPA [44-45] aims to contact a user regardless of devices available at hand and usable communication protocols. The iProxy/iMobile system [47] is a cross-platform middleware that allows client applications on various devices with various underlying protocols to access web services. VHE [48] in UMTS aims to provide global service portability that enables a user to tailor his service set and to receive personalized services, regardless of his location, access network or device type. In general, most systems support personal mobility by using a performance enhancing middleware server [49] situated between mobile devices and correspondent nodes to perform protocol conversion, content adaptation, and content caching. These middleware servers can store and use information about users preference, devices capability, and personal service profiles to adapt contents for mobile devices accordingly.

There is little research on handing over between user devices application sessions. iMASH [3] implements a system to fulfill this concept and uses it in a special-purposed healthcare domain. Besides, web session handoff can be realized by using process migration, Virtual Network Computing (VNC) [50], or Remote Desktop provided by Microsoft Windows XP operating system. However, protocols for process migration are too complex and heavy to suit application handoff. Furthermore, VNC and Remote Desktop are intrinsically different from application handoff since they are designed for remote console that outputs the screen of a first device to a second device and sends user input to the first device for processing. The transmission of screen data is a weakness since it may consume much network bandwidth. Besides, they cannot work well between heterogeneous user devices.

To hand over web sessions, Browser State Repository (BSR) Service [20] has implemented a BSR repository server and a BSR plug-in for Microsoft Internet Explorer (IE)

5.0. The BSR plug-in of a first device takes a snapshot of a browser's state and saves the snapshot at a BSR repository server, and then a browser on a second device can retrieve the saved snapshot to resume the saved state and session. Note that the first device is hereafter referred to as *source device* since it is the source providing session information, while the second device is referred to as *target device* since it is the target to accept saved snapshot. In its current implementation, BSR service only supports Microsoft's desktop OS and does not show the ability to hand over web sessions between heterogeneous devices (though it envisions such scenarios). Besides, BSR is a client-based approach and therefore requires much effort for modification to user devices.

3.3.2 Approaches for web session handoff

A web session handoff system should be able to track and obtain session information from source devices so that session information can be handed over as much as possible and a session can be more completely resumed on the target device. There can be three alternatives to web session handoff: client-based, server-based, and proxy-based approaches. As described in [20], server-based approaches are restricted by the information that a client sends to a web server and are not compatible to web servers that do not support web session handoff. Client-based and proxy-based approaches can hand over more session information than server-based approaches, but require modification to clients and proxies respectively. Since these two kinds of approaches require no modification to web servers, session handoff can be performed transparently to web servers and therefore server compatibility problem needs not be concerned.

The main advantage of client-based approaches is that complete and precise session information can be obtained since a modified client can locally monitor complete session information. However, each client-based approach is currently restricted to user devices that have specially modified browsers or specialized client programs for each approach. It is impractical and inefficient to modify all existing browser programs for all types of user devices unless a standardized or popular web session handoff protocol is widely recognized.

In proxy-based approaches, a proxy should record all HTTP requests and responses passing through the proxy and perform additional treatment to track session information. Proxy-based approaches have two advantages over client-based approaches:

1. **Practicability**: Proxy-based approaches have high practicability since they require less or no modification to user devices and therefore are easy to support more browsers and devices. Although client-based approaches can hand over complete and precise session information, they require in-depth knowledge about browsers and substantial modification to user devices. For example, the modification to user devices in BSR [20] is made directly to Microsoft's IE 5.0. In contrast, an unmodified user device in proxy-based approaches can still hand over a session without losing much session information. Besides, a user device in proxy-based approaches can be slightly modified without in-depth knowledge about browsers to hand over more session information. Thus client-based approaches cannot work properly when modified devices are un-available to users. Besides, users may suffer from the problem that their favorite browsers do not have a modified version that supports web session handoff.
2. **Fault tolerance to unreachable source device**: During handoff, a source device may be unreachable due to lack of battery or wireless connection. In this case, a proxy-based approach can still hand over a session because this session has been tracked by a proxy while the user is browsing. As a result, session handoff in a proxy-based approach fails only when the tracking proxy is unreachable. On the contrary, client-based approaches store session information in source devices or outer repositories (e.g. BSR repository), and therefore session information will not be stored or handed over if source devices, outer repositories, or network connections to them fail. In summary, proxy-based approaches are tolerant to the failure due to unreachable source devices. Thus, proxy-based approaches are more robust than the client-based approaches in terms of fault tolerance, especially in wireless and mobile environment.

On the contrary, compared with client-based approaches, proxy-based approaches have two disadvantages:

1. The additional treatment generates extra load to proxies.
2. Session information tracked by proxy-based approaches might be incomplete since some session information may not pass through a proxy and will not be tracked by the proxy.

To overcome these disadvantages, we adopt in our design a proxy-based approach with optional assistant information from user devices. The assistant information includes session information that cannot be tracked by a proxy, but can be collected by a slightly modified browser or an optionally installed client program in a user device. With the combination of proxy-based and client-assisted scheme, the proposed approach elaborated in the following sections outperforms the existing approaches.

3.4 Related work on fasth Handoff for 802.11 networks

3.4.1 Pre-authentication and Frequent Handoff Region

Much research adopts performing pre-authentication [27,51-52] to APs in order to speed up handoff procedure. With pre-authentication, an MN performs authentication to one or more APs to establish security association between the MN and APs before an MN hands over and therefore the security information for the MN, such as key material, can be proactively distributed to the APs. As a result, when the MN enters the coverage of a pre-authenticated AP, the MN need not perform authentication to the pre-authenticated AP and the AP can directly use the proactively distributed key to communicate with the MN. Handoff latency is therefore reduced, especially in networks that adopt remote authentication procedure, e.g. RADIUS [23] and DIAMETER [24].

However, performing pre-authentication to all neighbor APs may be unnecessary. Therefore, some research uses predictive approach [27,51-52] to drive a reduced set of APs to which pre-authentication is performed. S. Pack and Y. Choi propose in [27,51] a fast handoff scheme that performs pre-authentication to a reduced set of neighbor APs, called Frequent Handoff Region (FHR). In this scheme, an MN entering the coverage of an AP, performs authentication procedures to the current APs as well neighbor APs selected by a FHR selection algorithm, which takes into account users' mobility patterns, service classes, etc.

3.4.2 Neighbor Graph

In [28-29,53], A. Mishra *et al.* use a data structure, the Neighbor Graph, to dynamically capture the mobility topology of a wireless network as a means for discovering candidate APs [29] and proactively distributing key [28]. The Neighbor Graph records the channel used by each AP and APs' reassociation relationship, which exists between two APs if an MN can perform reassociation from one of the two APs to another AP but may not exist between two APs that are geographical neighbor. Figure 3-6 shows an example AP topology of a door environment and the corresponding Neighbor Graph. In this environment, there are four APs in passages and one AP in a room. The dashed lines in the left of Figure 3-6 show the potential paths of motion and the reassociation relationship between APs. As shown in the example, AP_A and AP_C are geographically neighbor but does not have reassociation relationship. With the knowledge of the Neighbor graph, an MN can perform pre-authentication and proactive key distribution only to a reduced set of candidate APs that have reassociation relationship with the MN's current AP. Therefore, those neighbor APs that are not candidate will not incur the overhead for pre-authentication and proactive key distribution. Besides, the MN, upon handoff, can restrict the AP discovery procedure only to the channels used by the candidate APs, not to 11 or more channels. Furthermore, the MN, knowing from the Neighbor Graph the number of candidate APs at each channel, can advance the termination of AP discovery procedure for a channel when the MN have received probe responses from all candidate APs at the channel. Therefore, the latency for AP discovery procedure can be greatly reduced.

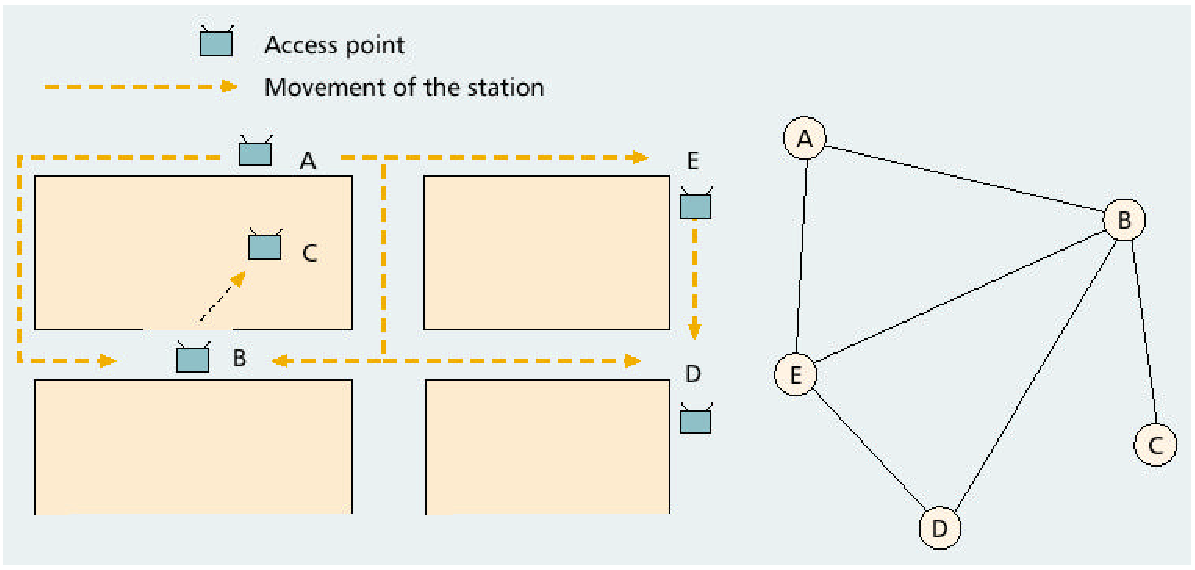


Figure 3-6. An example AP topology and corresponding Neighbor Graph

Chapter 4 Dynamically Configurable Network Address Translation (DCNAT) and Information Pushing

In this chapter, we first describe a dynamically configurable NAT (NAPT, more precisely), henceforth referred as DCNAT, approach to the inbound session problem. DCNAT is originally proposed in [54]. We then describe our improvement to DCNAT and extend DCNAT to fit for information pushing service.

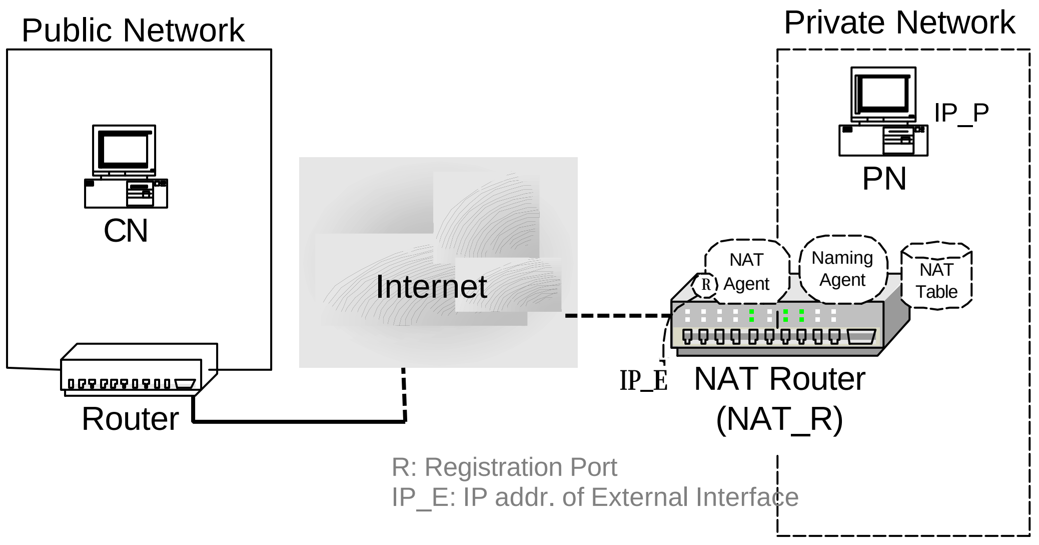


Figure 4-1. System architecture of DCNAT

4.1 DCNAT

4.1.1 System architecture and protocol flow

Figure 4-1 shows the system architecture and major functional components of DCNAT. Similar to a traditional NAT, DCNAT incorporates a Naming Agent for intercepting and processing identifier-to-IP queries to hosts inside a private network. It replies the external IP (henceforth referred to as *IP_E*) of the NAT router, rather than the private IP (henceforth referred to as *IP_P*) of the intended private host *PN,* for the identifier-to-IP queries originated by external hosts outside the private network. The Naming Agent is a DNS_ALG if the unique identifier of *PN* is a domain name. However, if the unique identifier is a telephone number or an NAI [55], the Naming Agent may be an ENUM [56] or other directory services.

In addition, DCNAT uses an **NAT Agent** for session registration process, data port selection, and NAT table management. The detailed function of the NAT Agent will be explained later.

In addition to the two agents, DCNAT defines three additional ports:

- **Service port**: a port on which PN is expecting data from *CN*.
- **Data port**: a port on *NAT_R*'s external interface, selected dynamically by NAT Agent for a specific session from *CN* to *PN*.
- **Registration port**: a port on *NAT_R*'s external interface that NAT_R opens for Session Registration Request messages from the Internet.

DCNAT assumes that *CN* is a DCNAT-enabled proxy or host, and knows the unique identifier of *PN* it intends to communicate with. Furthermore, *CN* knows the Registration port on the NAT Router for perform Binding Entry Request procedure, and the service port on which *PN* expects to receive inbound packets.

Before *CN* sends packets to *PN*, it first must issue a Binding Entry request, containing *PN*'s identifier and desired service port, to *NAT_R*. Upon receiving the Binding Entry request, NAT Agent on *NAT_R* creates NAT binding entries for *PN*, and replies to *CN* with the external IP of *NAT_R* and the data port associated with the newly created binding entry(s). After the Binding Entry Request procedure, *CN* knows which address and data port it should use for sending packets to *PN*. *NAT_R* can then translates the destination IP and port of the packet sent by *CN* from "external IP of *NAT_R*:data port" into "Private IP of *PN*:service port" and can route the translated packet to *PN*.

Figure 4-2 illustrates the protocol flow for establishing an inbound TCP session with DCNAT. The steps are explained as follows:

1. *CN* issues a DNS request in order to query *PN*' s IP address. This packet is routed to *NAT_R* according to DNS [7] mechanism.

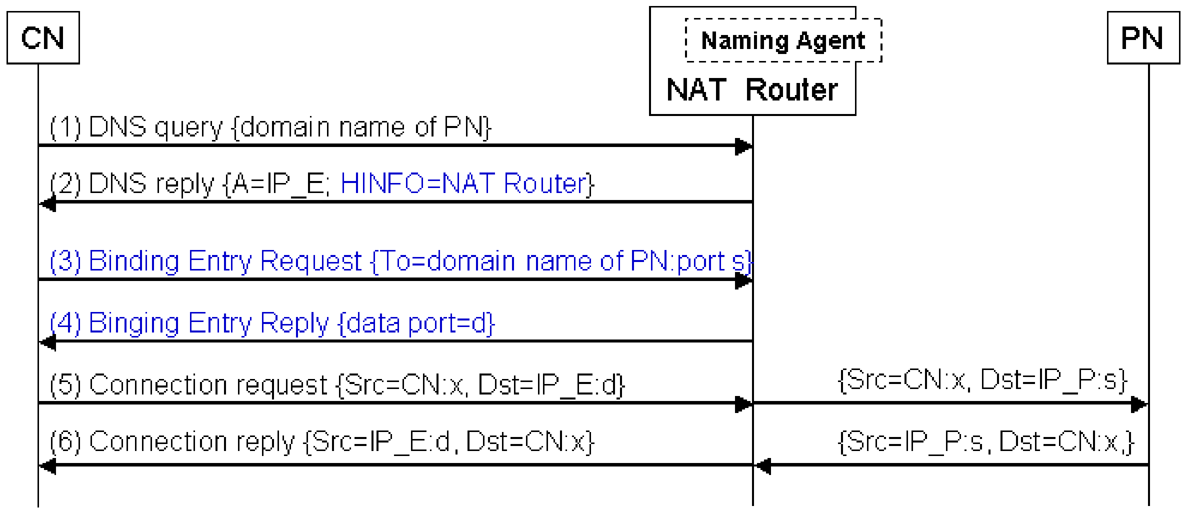

CN
Naming Agent
NAT Router
PN
(1) DNS query {domain name of PN}
(2) DNS reply {A=IP_E; HINFO=NAT Router}
(3) Binding Entry Request {To=domain name of PN:port s}
(4) Binging Entry Reply {data port=d}
(5) Connection request {Src=CN:x, Dst=IP_E:d}
{Src=CN:x, Dst=IP_P:s}
(6) Connection reply {Src=IP_E:d, Dst=CN:x}
{Src=IP_P:s, Dst=CN:x}

Figure 4-2. Protocol flow of DCNAT

2. Naming Agent in *NAT_R* intercepts the DNS request and discovers that it is an inquiry for a host within its private network. Therefore, Naming Agent sends *CN* a DNS reply containing a DNS resource record of type **A** [8] indicating *NAT_R*'s external IP address (henceforth referred to as *IP_E*) and a DNS resource record of type **HINFO** [8] indicating that *IP_E* belongs to an NAT router.

3. Upon receiving the DNS reply, *CN* discovers from the HINFO resource record of the DNS reply that *PN* is a host behind an NAT router with *IP_E*. Therefore *CN* performs a **Binding Entry Request procedure** by sending a Binding Entry Request message to *NAT_R*'s external IP address (*IP_E* acquired from the DNS reply). The Binding Entry Request message specifies the designated *PN* and the desired service port on which *PN* expects to receive inbound packets.

   When the Binding Entry Request message arrives at *NAT_R*, the NAT Agent in *NAT_R* performs the following steps:

   3.1 NAT Agent selects an unused port of *IP_E* as the data port for the Binding Entry Request.

   3.2 NAT Agent creates two entries in the NAT table: a DNAT entry for the binding between "*IP_E*:data port" to "*IP_P*:service port" for inbound traffic and an SNAT entry for the binding between "*IP_P*:service port" to "*IP_E*:data port" for outbound traffic (NAT Agent learned the service port from the Binding Entry Request message and *IP_P* via normal DNS operation internally within the private network). Note that a Binding Entry Request message can carry multiple destination PNs. In this condition, the NAT router creates binding entries between CN and each PN.

4. NAT Agent sends a Binding Entry Reply message to inform *CN* of the result, containing IP_E and data port.
5. Upon receiving the Binding Entry Reply message, *CN* sends a connection request to the data port on *IP_E* if the registration succeeds.
   5.1 Upon reception of the connection request packet, *NAT_R* translates the destination IP and port of the packet from "*IP_E*:data port" into "*IP_P*:service port" according to the NAT table. The translated packet is then routed to *IP_P* (*PN*).
6. PN receives the packet on the service port and replies message to *CN* using the service port as the source port.
   6.1 When the replied packets arrives at *NAT_R*, *NAT_R* translates the source IP and port of the packets from "*IP_P*:service port" into "*IP_E*:data port" according to the NAT table. Finally, the translated packets are then routed toward *CN*.

4.1.2 Modification to NAT routers

In order to support DCNAT, an NAT router should be modified. First, the Naming Agent should be modified so that the Naming Agent will include in DNS reply an HINFO DNS resource record in addition to an A DNS resource record. The value of HINFO DNS resource record is "**NAT Router**" indicating that the IP address in the A DNS resource record belongs to an NAT router.

Besides, the NAT Agent in an NAT router should be modified to support the following functions:

- ***Processing Binding Entry Requests***: The NAT Agent opens a port at the NAT router's external interface waiting for Binding Entry Request messages. When a Binding Entry Request arrives at the NAT router, the NAT Agent examines the request, performs data port selection and NAT table manipulation if the Binding Entry Request is granted, and issues a Binding Entry Reply (containing request status, lifetime, data Port, etc) to the originator of the Binding Entry Request message.

- ***Data port selection***: At the external interface of an NAT router, a pool of ports is reserved as data ports for inbound sessions. When a Binding Entry Request arrives at the NAT router, the NAT Agent selects an unused port from the reserved ports as the data port for this Binging Entry Request.
- ***NAT table manipulation***: The NAT Agent creates two NAT binding entries in the NAT table (as described in protocol Step 4.2). The NAT binding entries have a timer and if the timer expires then NAT Agent deletes the corresponding entries. The lifetime of binding entries are extended if packets of the associated session arrive continuously.

4.1.3 Modification to originating sides

Besides modifications to an NAT router, a DCNAT-enabled host should be modified to perform the following functions in order to originate inbound sessions to a private network.

1. Determine whether the destination address is an NAT router or a normal host: The host should examine the DNS reply to check if the reply contains an extra HINFO DNS resource record indicating that the IP address in the A DNS resource record belongs to an NAT router.
2. Perform the Binding Entry Request procedure.
3. Redirect data packets: The host should connect to the data port indicated in the Session Registration Reply instead of the original service port.

Although DCNAT requires modification to CN, the modification can be retrained to some specific hosts. The modification would not be an issue for new Internet applications that would like to pass through NAT routers since these applications can be developed by incorporating DCNAT functionalities. For example in real-time information pushing services, only the push server needs to be equipped with DCNAT functionalities, and the content server of an Internet service provider need not be aware of the DCNAT functionalities.

4.1.4 Use Port-Restricted Cone NAT in DCNAT

In DCNAT, a CN may connect a data port different from the original service port. This phenomenon is due to using Full Cone NAT [5] in the original DCNAT (henceforth referred to as Full Cone DCNAT). In Full Cone NAT, address translation is performed if the destination address and/or port of an inbound packet match a binding entry or the source address and/or port of an outbound packet match a binding entry. Therefore, if an NAT router can map its port 23 to port 23 on two hosts, PN1 and PN2, the destination address of inbound packets desired for port 23 on PN1 and PN2 will both be the external address of the NAT router and port 23. This makes the NAT router unable to distinguish the desired destination of these inbound packets. As a result, an NAT router using Full Cone DCNAT cannot map its one port to more than one PN. In the above example, port 23 on either PN1 or PN2 cannot be mapped to port 23 on the NAT router and will be mapped to another port. Due the possibility of this port mapping, the applications on CNs should be able connect to another data port instead of original service port (as mentioned in Section 4.1.3). This requires modification to application programs or CNs.

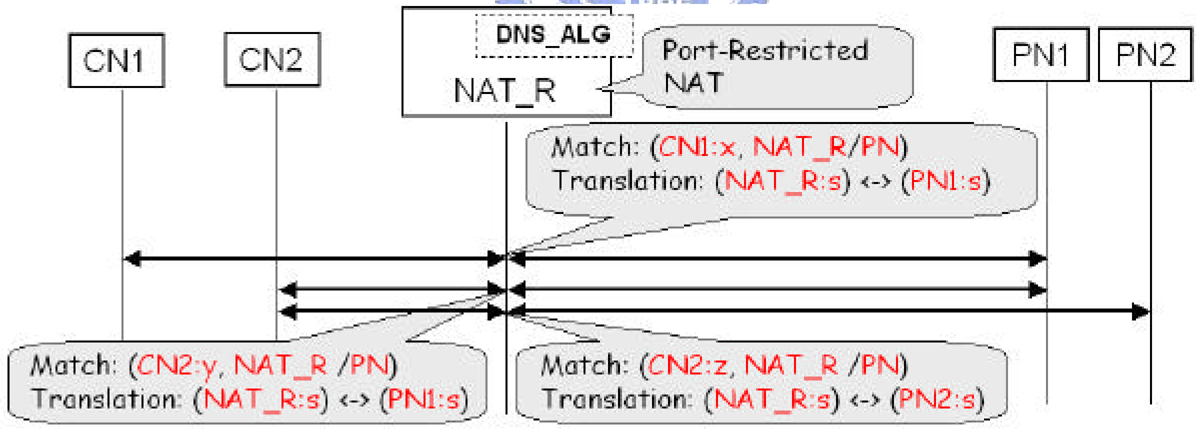

CN1
CN2
DNS_ALG
NAT_R
Port-Restricted NAT
PN1
PN2
Match: (CN1:x, NAT_R/PN)
Translation: (NAT_R:s) <-> (PN1:s)
Match: (CN2:y, NAT_R /PN)
Translation: (NAT_R:s) <-> (PN1:s)
Match: (CN2:z, NAT_R /PN)
Translation: (NAT_R:s) <-> (PN2:s)

Figure 4-3. Use Port-Restriction Cone NAT in DCNAT

To avoid connection to another data port and reduce modification to application programs, we introduce Port-Restricted Cone NAT [5] to DCNAT (henceforth referred to as Port-Restricted Cone DCNAT). In Port-Restricted Cone NAT, inbound packet is translated if the source address, source port, and destination address match a binding entry and outbound packet is translated if the source address, destination address, and destination port match a binding entry. Therefore, an NAT router can map its one port to more than one PNs. Figure 4-3 shows an example. In this example, both CN1 and CN2 have connections to port *s* on PN1. Since the inbound packets sent by CN1 and CN2 have different combinations of source address, source port, and destination address, the NAT router can correctly multiplex and translate inbound packet destined to its port *s* to PN1's and PN2's port *s*. Similarly, the outbound packets can also be correctly multiplexed and translated according to the source address, destination address, and destination port. Besides, if CN2 has simultaneous connections to port *s* on PN1 and PN2, the NAT router still can correctly multiplex and translate packets since the packets to/from PN1 and PN2 have different combinations of source address, source port, destination address, and destination port. As a result, by using Port-Restricted Cone DCNAT, applications on CNs need not connect to another port instead of original service port. Besides, applications on CNs can utilize Port-Restricted Cone DCNAT without modification since the related modification to perform the other two functions mentioned in Section 4.1.3 can be implemented in CN's system libraries or auxiliary programs.

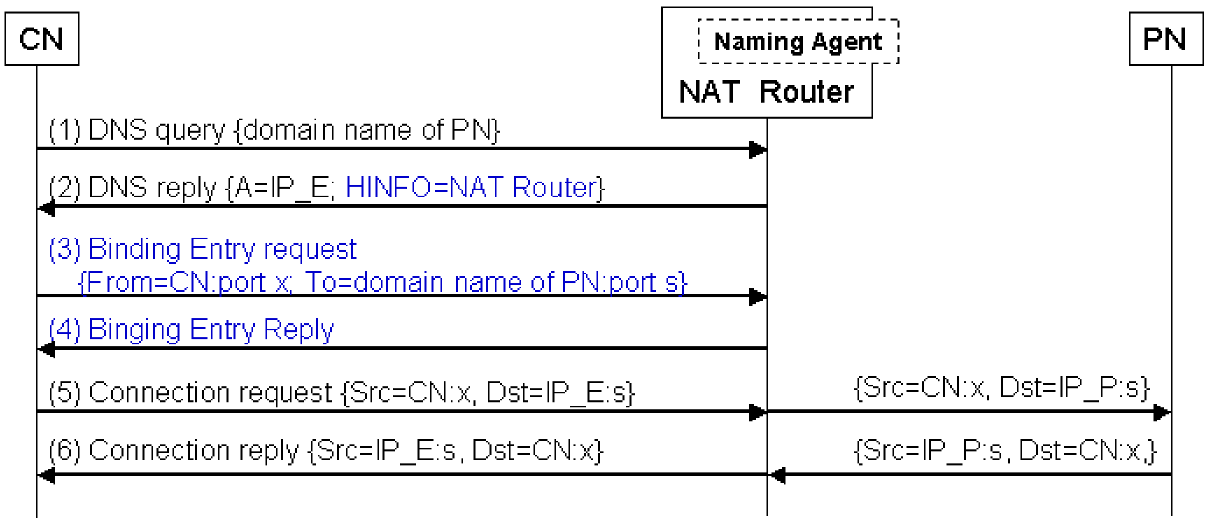

CN
Naming Agent
NAT Router
PN
(1) DNS query {domain name of PN}
(2) DNS reply {A=IP_E; HINFO=NAT Router}
(3) Binding Entry request
{From=CN:port x; To=domain name of PN:port s}
(4) Binging Entry Reply
(5) Connection request {Src=CN:x, Dst=IP_E:s}
{Src=CN:x, Dst=IP_P:s}
(6) Connection reply {Src=IP_E:s, Dst=CN:x}
{Src=IP_P:s, Dst=CN:x}

Figure 4-4. Protocol flow of Port-Restricted Cone DCNAT

Figure 4-4 shows the protocol flow of Port-Restricted Cone DCNAT. This protocol flow differs that shown in Figure 4-2 in Steps 3 to 6. In Step 3, the Binding Entry Request sent from CN contains an extra field for indicating the source address and port from which the CN will initiate a connection. With the extra field and the desired destination field, the NAT router can then establish Port-Restricted Cone binding entry(s) without mapping the desired port of the desired PN onto another port on the NAT router. In Step 4, the NAT router reply CN with a Binding Entry Reply without needing to contain a mapped data port. In Steps 5 and 6, CN connects to the desired service port of PN via the same port number of the NAT router.

4.1.5 Addition and removing of binding entries

In Bi-directional NAT and Tunnel Approach, an NAT router creates binding entries when a CN performs DNS lookup. On the other hand, DCNAT adopts a Binding Entry Request procedure independent of DNS lookup. Coupling binding entry creation with DNS lookup is intrinsically not suitable for either Full Cone DCNAT or Port-Restricted Cone DCNAT. In Full Cone DCNAT is used, CNs can learn the mapped data port for its desired service port from Binding Entry Reply message but cannot learn from DNS reply messages. In Port-Restricted Cone DCNAT, in which port mapping is avoided, is used, an NAT router can learn from Binding Entry Request message the port with which a CN uses to communicate with a PN but cannot learn from DNS lookup messages. As regards this point, one may argue that if CN can send DNS lookup packet and connection packet from the same port on CN, the Binding Entry Request procedure can be omitted. However, most UNIX or Windows network application programs utilize system library to perform DNS lookup and therefore DNS lookup packet is sent from a port different from that used by an application program. Therefore, the Binding Entry Request procedure cannot be replaced by DNS lookup.

Besides, using DNS lookup to create binding entries has the drawbacks of creating useless binding entries when a CN performs DNS lookup without any connecting action. On the contrary, DCNAT will not create useless binding entries when a host performs DNS lookup. Unfortunately, useless entries could still exist when a malicious program performs Binding Entry Request procedure without originating any connections. A simple remedy to the creation of useless binding entries is to initialize a binding entry a short lifetime and extend the lifetime when the first data packet matching that entry arrives, or another DNS lookup or registration request resulting in the same binding entry arrives.

Moreover, all approaches mentioned in this paper might suffer from Denial-Of-Service (DOS) or Distributed- Denial-Of-Service (DDOS) attack, which are difficult to solve, in the form of continuous DNS lookup or requesting for creating useless binding entries. Even though there are no perfect solutions to DOS and DDOS attacks, DCNAT is more robust than other approaches since DCNAT decouples creation of binding entry from DNS lookup. With this decoupling, DDOS registration attack to NAT router is more difficult to perform since a malicious host is difficult to find hosts willing to perform Binding Entry Request procedure for its distributed attack. On the other hand, a malicious host is easy to find hosts in the current Internet environment willing to perform DNS lookup.

In comparison with addition of binding entries, removing of binding entries should also be taken care. RFC2663 [4] briefly discusses the considerations about the time to remove NAT binding entries in an NAT router. Besides the consideration discussed in [4] and the prevention from creation useless binding entries discussed in the above paragraphs, a CN can reasonably request an NAT router to remove binding entries that is previously created on the CN's request.

4.1.6 Proxy-based applications with DCNAT

This subsection describes how a proxy-based application can utilize DCNAT. DCNAT is well suited to proxy-based applications since the application proxies can perform DCNAT functions transparently on behalf of client hosts. Thus hosts without modification can originate inbound sessions into private networks.

In proxy-based applications, application layer proxies, such as web proxies for HTTP, are situated between communication peers. Such applications have the following properties:

1. An originating host originates a connection request to a proxy using the application protocol.
2. The request carries the identification of the destination host as the payload of the application protocol.

3. The proxy determines the location of the destination host by certain lookup scheme such as directory service or DNS.
4. The proxy establishes a connection to the destination host for the originator hosts and inter-works for the two hosts.

With DCNAT-enabled proxies and NAT routers, hosts without modification may originate connection to the hosts in a private network behind an NAT router.

4.2 Information pushing with DCNAT

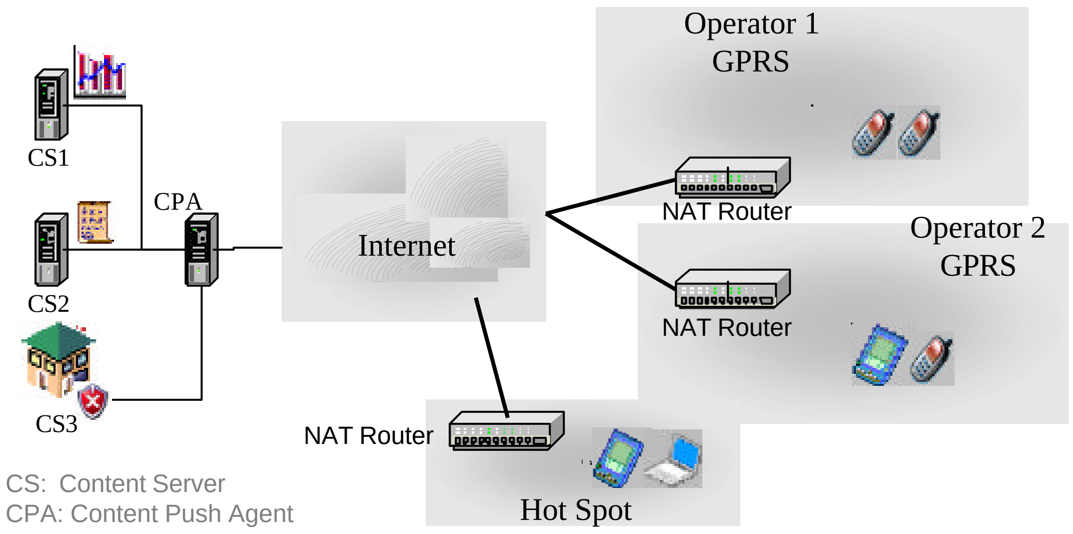


Figure 4-5. Information pushing scenario

Figure 4-5 shows an application scenario of information pushing. In this scenario, CSs (Content Server) are producers of various paid or unpaid public information service content. CS1 produces stock information, CS2 produces news, and CS3 produces home surveillance alert. The CPA (Content Push Agent) is the deliverer for sending the produced information content through NAT routers to subscriber nodes (SN) spread over several networks. Although a CS capable of performing a CPA's functionalities also works, we assume that the functionalities of CS and CPA are deployed at separated node. Such functionality separation allows many simple or private CSs sharing a public CPA.

In practical condition, SNs can be in NAT networks or public networks and move between these networks. Since there is little difficulty in delivering information to SNs in public networks, this thesis focuses on delivering information to SNs in NAT networks.

4.2.1 Sketch protocol flow

The proposed information pushing protocol works on the assumption as follows:

1. An NAT router receives only one copy of produced information service content regardless of the number of SNs in the NAT network subscribing the information service content.

1.1 An NAT router can uses either in-LAN multicast or multiple unicast to send information service content SNs behind the NAT network. Using multicast can greatly reduce network traffic in comparison with use of multiple unicast if the number of subscribers in the NAT network is large. Besides, using in-LAN multicast has two advantages. First, delivery of information content would not be blocked by Internet routers not supporting multicast. Second, the complex mechanism of Internet multicast [57-58] need not be tackled.

1.2 Each SN receives a specific information service content on a service port at its private IP address or an IP multicast address. The multicast address (if used) and service port for receiving different information service content can be different and be assigned by a CPA or a CS when SNs subscribe to information services content.

2. Each SN uses its unique identifier, e.g. NAI [55] and E.164 number [56] to subscribe to information service content and therefore CSs or CPAs know the unique identifier of each SN. Besides, CSs, CPAs, or NAT routers can perform adequate query to learn each SN's current IP address, network, or NAT router by using each SN's identifier (will be described in Section 4.2.3).

3. Each CPA knows the registration port on an NAT router to which the CPA can perform Binding Entry Request procedure as if an Internet node does in DCNAT.

Since the proposed information pushing approach is based on DCNAT, the definitions and usages of the three terms: service port, registration port, and data port, are the same as those in

DCNAT.

Figure 4-6 illustrates the sketch message delivery protocol flow. This protocol flow incorporates the Binding Entry Request procedure of Full Cone DCNAT in which a CPA should request an NAT router to establish binding entries before pushing information content. The sketch protocol flow can be extended to fit for different scenario with unicast, multicast, or different subscribers mobility handling mechanism. For purpose of simplicity, the sketch protocol flow does not tackle subscriber mobility management and shows only one NAT router (NAT_R) while the protocol flow of multiple NAT routers can be easily deduced. The steps are explained as follows:

1. The CS produces and sends an issue of a specific information service to the CPA. Besides, the CS may also send the CPA a list of SNs that may otherwise be maintained by the CPA.

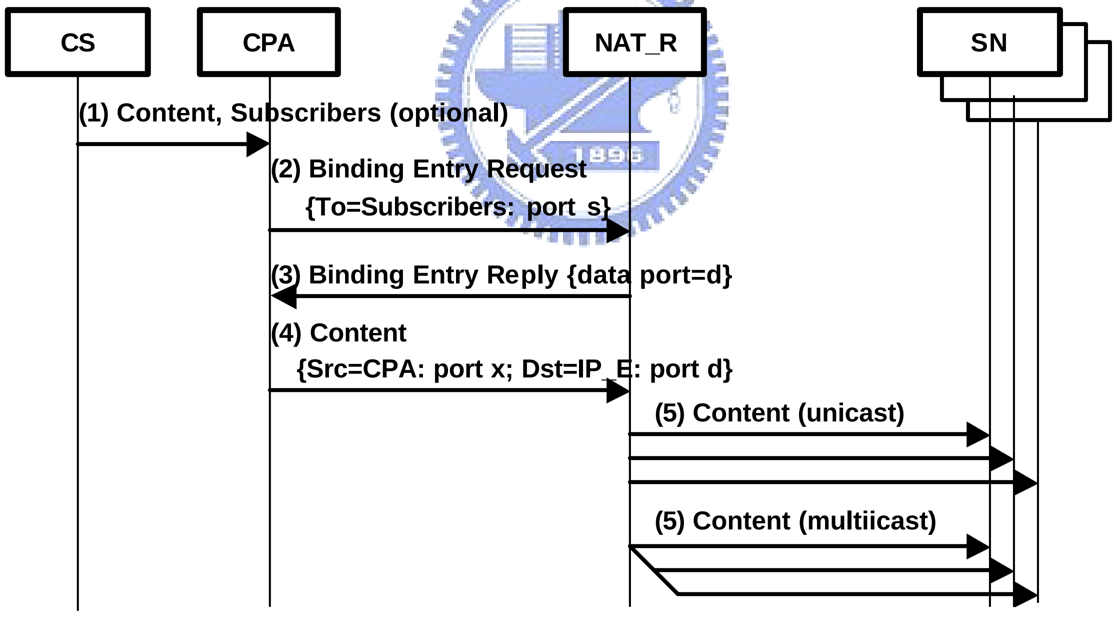

CS
CPA
NAT_R
SN
(1) Content, Subscribers (optional)
(2) Binding Entry Request {To=Subscribers: port s}
(3) Binding Entry Reply {data port=d}
(4) Content {Src=CPA: port x; Dst=IP_E: port d}
(5) Content (unicast)
(5) Content (multiicast)

Figure 4-6. Message delivery protocol flow

2. According to the mechanism mentioned in the subsection 4.2.3, the CPA decides that one or multiple binding entries on the NAT_R should be established before sending the information content. Therefore the CPA then sends a **Binding Entry Request** message to "NAT_R's external IP address (IP_E) : registration port", which contains

the following two elements:

  a. **A destination multicast address** (if all SNs receive the specific information service content at the same multicast address) or **a list of identifiers of each SN** [55-56] (if each SN receives the specific information service content at its private IP address).
  b. **A destination service port**.

3. Upon receiving the **Binding Entry Request** message, the NAT_R selects an unused port on its external interface (named data port) for receiving the oncoming information content and creates binding entries according to the destination type:
  a. **If the destination is a list of identifiers of each SN, two binding entries for each SN**, one for the inbound direction and the other for the outbound direction (if necessary), will be established. In this case, the NAT_R performs adequate query to resolve each SN's IP address according to the SN's identifier and then establishes binding entries for mapping between "IP_E: data port" and each SN's "private IP address : service port".
  b. **If the destination is a multicast IP address, total two binding entries**, one for the inbound direction and the other for the outbound direction (if necessary), will be established. In this case, NAT_R establishes binding entries for mapping between "IP_E: data port" and "destination multicast IP address: service port".
4. The CPA sends the information content to "IP_E: data port".
5. According to the binding entries created at Step 3, the NAT_R multicasts the information content in the private network to "multicast IP address: service port" or perform multiple unicast to send the information content to each SN's "private IP address: service port".

4.2.2 Multicast address and port assignment

For purposes of differentiating between different information service content and delivering a specific information service content to desired subscribers, each SN should receive different issues of a specific information service content on the same multicast IP address (if used) and the same port number. In the proposed approach, the content delivery CPA or the content producing CS of a specific information service is responsible for assigning this information service a multicast IP address (if used) and a port number for receiving the information service content. The multicast IP address (if used) and the receiving port number are then learned by SNs when SNs subscribes to this information service.

Since there may be many CPAs and CSs on the Internet and a CPA may deliver content on behalf of many CSs, there should be a coordinating mechanism to avoid duplicated assignment by CSs and CPAs. In such a coordinating mechanism, CPAs should coordinate between each other to avo id duplicated or conflicted assignment. Furthermore, if CSs are capable of the assignment, their delivery CPA should coordinate between these CSs. Besides, if a coordinating mechanism will be widely adopted over the Internet, this mechanism will require globally or officially recognized multicast address allocation and therefore we do not propose a complete coordinating mechanism in this paper but just discuss the assignment policy.

As for the assignment policy, several factors may be taken into consideration, e.g., differentiating the SNs of each information service and saving multicast IP addresses and service ports. In order to differentiating SNs, each private information service (e.g. home security information) or each paid information services can be assigned an exclusively used address and an exclusively used port. On the contrary, in order to save multicast IP address and service ports, many information services can be assigned the same address with a shared or non-shared service ports. If many information services are assigned the same multicast IP address with a shared service port, the followings should be taken care:

1. On each SN, the application waiting on the service port should be able to process content of all subscribed information services received from the service port.
2. Since each information service will have its group of SNs, in order to keep content secret to non-SNs that receive the content maliciously or accidentally, a CPA or a CS can deliver content in a ciphered form, which can only be deciphered by the desired SNs.

4.2.3 Subscribers mobility

In order to handle user mobility for users moving between different NAT networks or different operators' networks, there should be a mechanism for CSs or CPAs to locate subscribers. However, existing approaches or proposals about user mobility management such as Mobile IP [41] and its variants may not suit information pushing service. In Mobile IP and its variants without route optimization [59], a correspondent node (CN) always knows the home address of a mobile node (MN) and therefore the communication path between the CN and the MN should initially pass through the MN's home agent. As a result, if the mobility management of Mobile IP or its variant is applied in the targeted environments of the proposed information pushing approach, a CPA or a CS will always know the home address of an SN instead of the SN's current NAT router and therefore the CPA or the CS will not be able to request the SN's current NAT router to create binding entry(s) for the oncoming information content.

Due to the above problem and the fact that Mobile IP and its variants are not widely deployed, a practicable and easy mechanism is that mobile each SN's actively register its current IP address, NAT router, or network to CSs, CPAs, or certain directory servers. The registration of an SN's IP address should be adopted when the SN uses a public IP address, otherwise the SN should register its NAT router or network, from which CSs/CPAs can deduce the SN's NAT router or with which CSs/CPAs can query a directory server to learn the SN's NAT router.

4.2.4 Illustration scenarios

This subsection illustrates two information pushing scenarios and their protocol flows. Each of the example protocol flow is extended from the sketch protocol flow shown in Figure 4-6 by including subscriber mobility handling mechanism. These scenarios are not meant for covering all combination of unicast, multicast, and subscriber mobility handling mechanisms but for illustrating the practicability of the protocol flow shown in Figure 4-5 in various information pushing services

4.2.4.1 epaper delivery

Figure 4-7 illustrates an example epaper delivery scenario and protocol flow. In this scenario, each SN subscribes to a specific information content at a CPA, registers its NAT router to this CPA, and an NAT router uses multicast to deliver information content to SNs. The subscription at a CPA and the registration to the same CPA is a reasonable approach widely adopted by most public epaper service provider. The protocol flow is describe as follow:

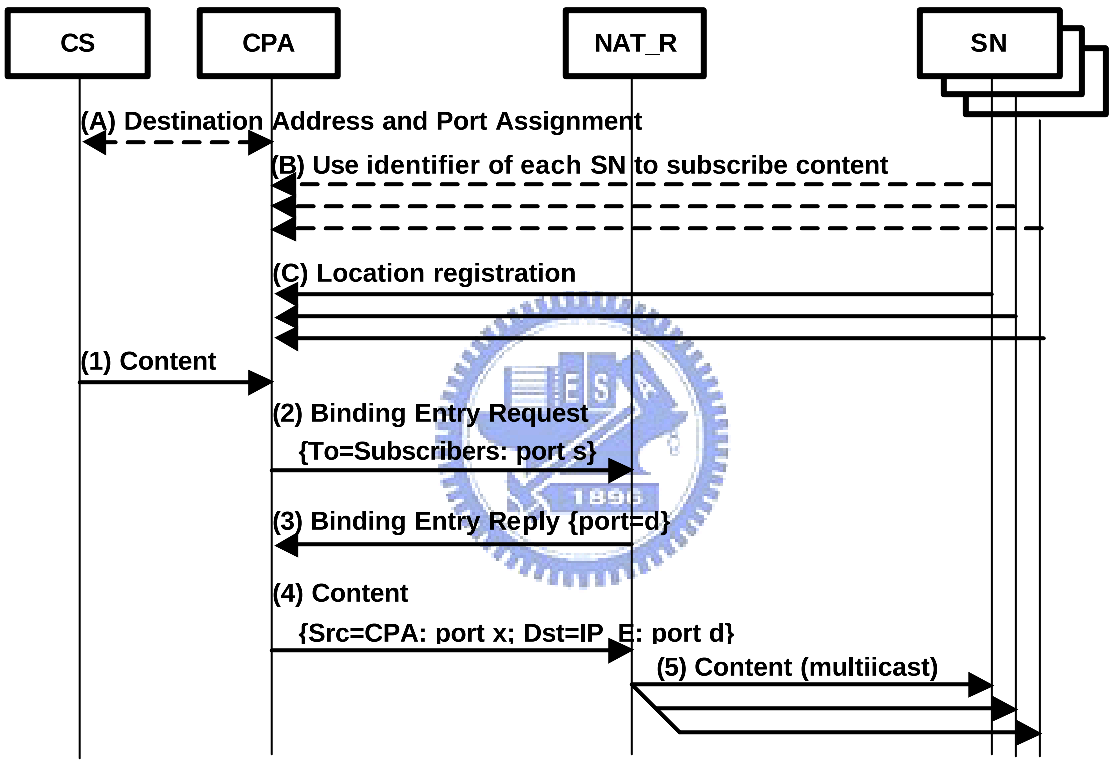

CS
CPA
NAT_R
SN
(A) Destination Address and Port Assignment
(B) Use identifier of each SN to subscribe content
(C) Location registration
(1) Content
(2) Binding Entry Request
{To=Subscribers: port s}
(3) Binding Entry Reply {port=d}
(4) Content
{Src=CPA: port x; Dst=IP_E: port d}
(5) Content (multiicast)

Figure 4-7. ePaper delivery protocol flow

A. CPA assigns CS a destination multicast address and port for SN to receive a specific information content. The assignment may be performed in out-of-band communication.

B. Each SN uses its identifier to subscribe to the specific information content at the CPA and therefore CPA can maintain a list of subscribers. The subscription may also be performed in out-of-band communication. At this time, each SN learns the destination multicast address and port for receiving the information content.

C. Each SN registers its location to the CPA when moves into another network.

1. The CS sends the CPA an issue of information content without subscriber list, which are maintained by the CPA.

2 and 3. The CPA, which knows the NAT router (NAT_R) of each SN at Steps B or C, requests the NAT_R to create binding entries.

4 and 5. The CPA sends the NAT_R one copy of the information content without subscriber list and then the NAT_R multicast information content to SNs in its local NAT network according to the binding entries created at Step 2.

4.2.4.2 home surveillance alerts delivery

Figure 4-8 illustrates an example home surveillance alerts delivery protocol flow. The delivery of home surveillance information allows home members being aware of their home environment event (e.g. thief break-in) or household appliances status. In this scenario, a CS situated at a house either produces or collects home environment event or household appliances status, a CPA is provided by a network operator or a security service firm. However, subscription, location registration, and information delivery in an NAT networks are performed in ways different from those in e-paper delivery. In this scenario, each SN subscribes to a specific information content at a CS, registers its NAT router to this CS, and an NAT router uses multiple unicast to deliver information content to SNs. The subscription at a CS and the registration to the same CS is reasonable for home surveillance alerts since only home members, whose number is usually small, care home surveillance alert information and the information usually is not open for public subscription. The protocol flow is describe as follow:

A. CPA assigns CS a destination multicast address and port for SN to receive a specific information content. The assignment may be performed in out-of-band communication.

B. Each SN uses its identifier to subscribe to the specific information content at the CS and therefore CS can maintain a list of subscribers. The subscription may also be performed in out-of-band communication. At this time, each SN learns the destination multicast address and port for receiving the information content.

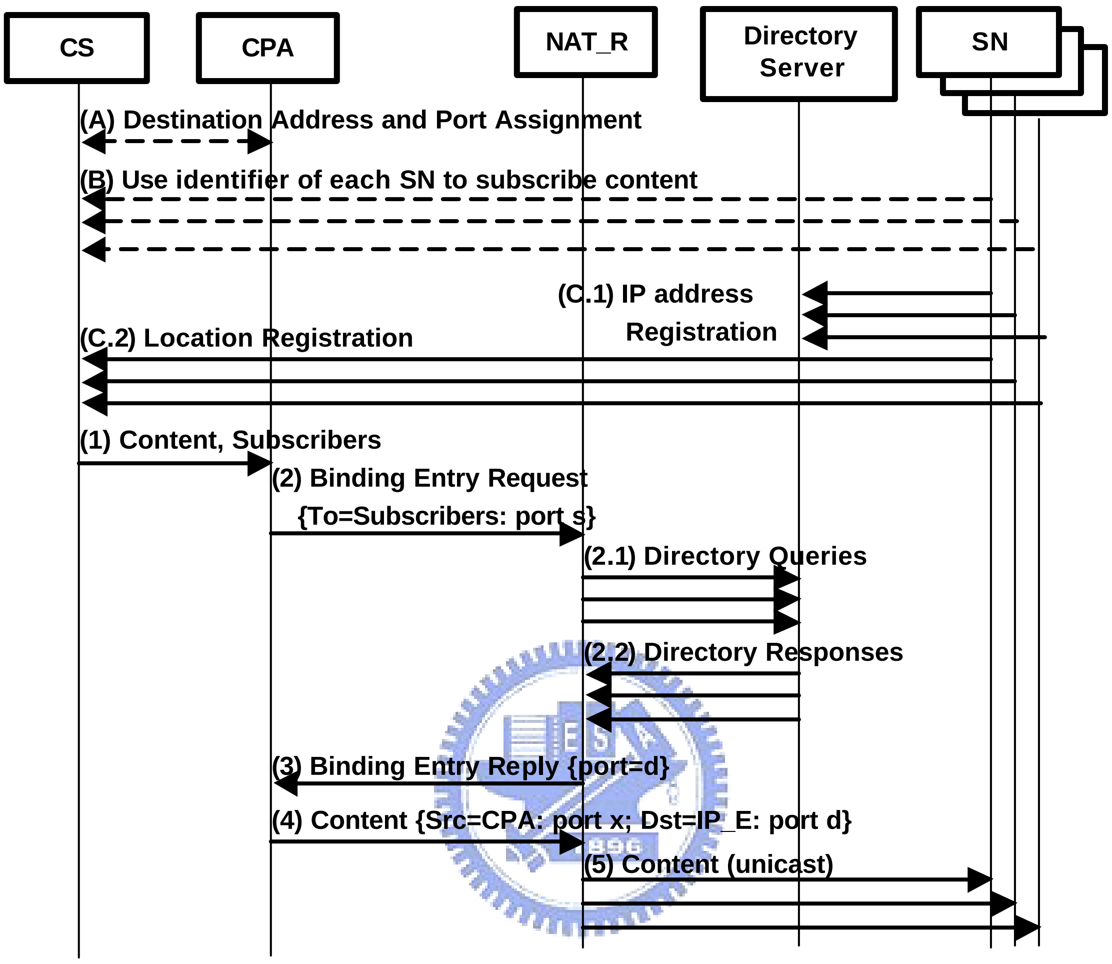

CS
CPA
NAT_R
Directory Server
SN
(A) Destination Address and Port Assignment
(B) Use identifier of each SN to subscribe content
(C.1) IP address Registration
(C.2) Location Registration
(1) Content, Subscribers
(2) Binding Entry Request {To=Subscribers: port s}
(2.1) Directory Queries
(2.2) Directory Responses
(3) Binding Entry Reply {port=d}
(4) Content {Src=CPA: port x; Dst=IP_E: port d}
(5) Content (unicast)

Figure 4-8. Surveillance alerts delivery protocol flow

C. Each SN registers its IP address to a local directory service server. If the directory service is DNS [7], the registration procedure can be combined with the procedure of SN's acquisition of IP address such as DHCP [60] and Dynamic DNS. Besides registering to the CS, each SN registers its location to the CS when the SN moves into another network.

1. The CS sends its produced information content along with subscribers list to the CPA.

2. The CPA sends a **Binding Entry Request** message to "NAT_R's external IP anddress : registration port", which contains a list of SNs' identifiers and the service port on which SNs receive information content.

3. Upon receiving the **Create Binding Entry Request** message, the NAT_R selects a data port on its external interface and queries directory service server to lookup each SN's IP address (Steps 2.1 and 2.2). The type of directory service server queried depends on the type of each SN's unique identifier. If SNs can use different types of identifiers, the NAT_R should query directory servers of different types. Besides, the NAT_R and a directory server may require multiple query-reply exchanges for query multiple identifiers. After completion of all queries, the NAT_R creates binding entries between "NAT_R's external IP address : data port" to each SN's "private IP address: service port".

4 and 5. The CPA sends the NAT_R one copy of the information content without subscriber list and then the NAT_R multicast information content to SNs in its local NAT network according to the binding entries created at Step 2.

4.3 Performance model and comparison

4.3.1 Qualitative comparison

In this subsection, we compare Bi-directional NAT, expanded NAT, and Tunnel Approach with DCNAT in terms of the number of needed IP addresses, extra processing, setup cost, and scalability. As mentioned previously, NAT Binding Inquiry schemes are not suitable for real-time information pushing services since they rely on NAT inquiry messages sent actively by the private nodes at the connection termination endpoints.

Table 4-1 summarizes the qualitative comparison of the four approaches to NAT Inbound Session Problem. In this table, scalability is judged according to two point-of-views. The first is the connection capacity that an NAT router offered to private hosts in an NAT network. The second is the number of NAT networks that a CN can connect to. Bi-directional NAT uses a public IP address for each NAT session so that its scalability is limited to the number of available public IP addresses. Expanded NAT has the port collision problem and does not support dynamically opened services. Therefore Expanded NAT might only work for small networks with pre-configured services. The Tunnel Approach can assign private IPs for PNs' external addresses because an IP-in-IP tunnel has been established between Intermediate System and the designated NAT router. Therefore it does not need any extra public IP address for NAT traversal and does not have the port collision problem. In addition, it requires no modifications to hosts in both public and private networks. However before a host can communicate with a private host, an IP-in-IP tunnel must have been established between the intermediate system *en route* and the intended private network. It is obviously impractical to establish tunnels toward all possible private networks beforehand. In DCNAT, before CN commences an inbound session to PN, CN registers with the destined NAT Agent two binding entries between "*IP_E*:data port" and "*IP_P*: service port" for PN. Therefore by examining the data port appearing in the destination port, the NAT router can distinguish inbound sessions destined to different internal hosts. Because the NAT router will redirect "*IP_E*:data port" to "*IP_P*:service port", PN can just listen on a service port for inbound messages. Therefore a DCNAT-enabled CN can initiate an inbound session to PN and multiple PNs can open the same service port without port collision. Besides, similar to NAT/NAPT, DCNAT is very scalable since it can support many sessions with only a single public IP address.

Table 4-1. Comparison of four NAT approaches

	Bi-directional NAT	**Expanded NAT**	**Tunnel Approach**	**DCNAT**
# of Public Addresses Needed	1 for each host	1	1	1
Extra Setup Cost	NAT Agent	Pre-configuration for known services	IP-in-IP tunnel, Intermediate System, NAT Agent	NAT Agent
Dynamic Opened Service	Yes	No	Yes	Yes
Modification to Host	No	No	No	Yes
Scalability	Limited to the number of available public IP	Good, but with port collision	Works where an IP-in-IP tunnel exists	Best

Although DCNAT requires some modification to CNs, previous experience on Internet applications has shown that modification to applications for supporting new features, such as passive mode ftp and WWW proxies, is acceptable. Furthermore, for the applications that have proxies inter-work between clients and servers, only the proxies, e.g., the CPAs in the proposed information pushing service, need to be DCNAT-aware.

4.3.2 Numerical results

This subsection presents the performance comparison between DCNAT and Bi-directional NAT in terms of service blocking probability. The performance of Expanded NAT and Tunnel Approach are not discussed here because their deployment are restricted by port collision, pre-configuration, or the number of predefined tunnels. We use the *M/G/n/n* loss system [61] to model the service blocking probability of an NAT router. In the *M/G/n/n* loss system, inbound session requests can be regarded as the service arrivals whose inter-arrival time is of exponential distribution and whose service duration is of general distribution, and *n* represent the number of allowed services and the offered connection capacity in an NAT router. From [61], the blocking probability, p_b, of *M/G/n/n* can be formulated by Erlang Loss Formula, as follows.

$$p_b = \frac{r^n / n!}{\sum_{k=0}^{n} \left(r^k / k! \right)}$$

where ?=?/µ is the traffic load, ? is the arrival rate, and µ is the service rate.

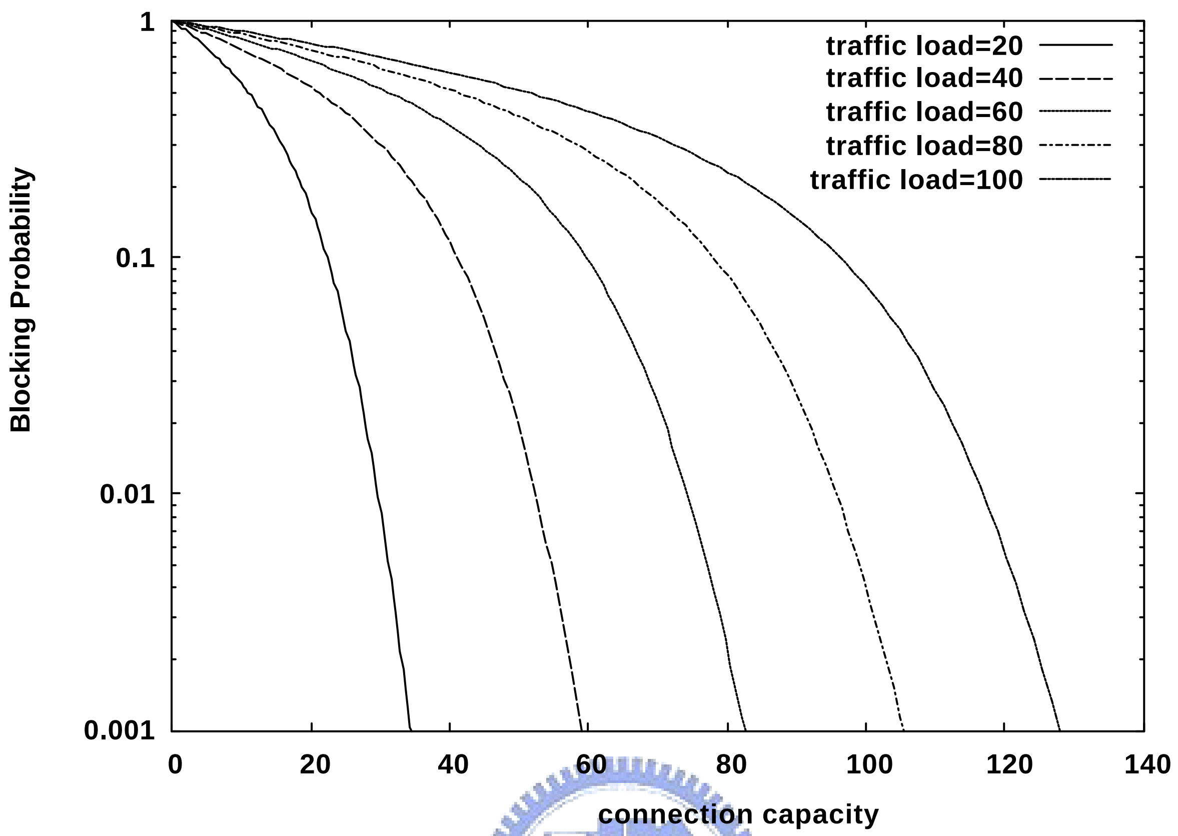


Figure 4-9. Blocking probability of *M/G/n/n* system

Figure 4-9 shows the relation between the offered connection capacity of an NAT router offers and the blocking probability under various traffic loads. As shown in this figure, in order to have a blocking probability lower than 1%, an NAT router needs to have capabilities of 30, 53, 75, 96, 117 connections, respectively, for traffic loads of 20, 40, 60, 80, and 100. For a blocking probability lower than 0.1%, an NAT router must have capacities of up to 35, 60, 83, 106, and 128 connections, respectively, for traffic loads of 20, 40, 60, 80, and 100.

One important point should be clarified. In Full Cone DCNAT, the connection capacity of an NAT router, having one public IP address, is exactly the same as the number of available ports. This is because that Full Cone DCNAT translation is based on IP address and port of the private side and therefore the NAT router can map its one port number to only one service on one private host. In Bi-directional NAT, though an NAT router performs translation according to only IP address of the private side, the connection capacity of the NAT router is not only the number of available public IP addresses. This is because that one private host can offer multiple services to Internet by using one public IP address. Therefore the connection capacity of a Bi-directional NAT router can be estimated as the number of available public IP addresses multiplied by the average number of services offered by a single host.

Generally speaking, the number of services offered by a host is small, usually several services at most. Therefore Bi-directional NAT would require a tremendous amount of public IPs when traffic load is high. On the contrary, DCNAT can offer more than 60,000 simultaneous connections by using just one single IP address. Moreover, if Port-Restricted Cone DCNAT is used, the connection capacity will be more.

4.4 Summary

This chapter proposes a DCNAT approach to the inbound session problem for a private network behind an NAT router. DCNAT can be generalized to fulfill general inbound sessions for private networks behind NAT routers and also be applied for real-time information pushing. For proxy-based applications, such as web browsing or real-time information pushing, only the proxies and NAT routers need to be equipped with DCNAT functionalities. DCNAT is a port-based NAT scheme that uses a Binding Entry Request procedure to create NAT binding entries so that dynamically inbound sessions into private networks are enabled, while also avoiding the port collision problem. With the dynamic creation of NAT binding entries, DCNAT provides lower blocking probability and better scalability by using just one or several public IP addresses.

Chapter 5 Hybrid Web Cache Replacement Algorithm

5.1 Hybrid algorithm combining LRU and GD-Family

As stated in section 3.2, the simple strategy used by LRU is suitable for maintaining short-term popularity while the comprehensive strategy used by GD variants is suitable for maintaining long-term popularity. Besides, giving an object highest cache ranking may be unnecessary before the cache knows the object is popular or not. Therefore, we propose a hybrid algorithm that combines the advantage of LRU, GD-Family [11,13-14], and SLRU [36]. The hybrid algorithm, similar to SLRU, also divides cache space into two zones, called hot zone and cold zone. Unlike SLRU, the cold zone adopts GD-Family to manage cold objects while the hot zone adopts LRU to manage hot objects.

Figure 5-1 shows the logical flow of the proposed hybrid algorithm. The logical flow is summarized as follow:

A. Cache Miss: The missed object is inserted into the cold zone according to its utilization value (A1 of Figure 5-1). If the cold zone does not have enough free space for the missed object, objects with minimum utilization values are removed for making space (A2 of Figure 5-1).

B. Cache Hit in cold zone: If the hitted object has been hitted in cold zone for less than *n* times (*n* is a system parameter), this object is re-ranked according to its re-evaluated utilization value. Otherwise, this object is promoted to the head of the hot zone (B1 of Figure 5-1). If the hot zone does not have enough free space for this promoted object, objects at the tail of the hot zone are removed from the hot zone (B2 of Figure 5-1) and inserted into the cold zone according to its re-evaluated utilization value (B3 of Figure 5-1).

C. Cache Hit in hot zone: The hitted object is moved to the head of the hot zone (C of Figure 5-1).

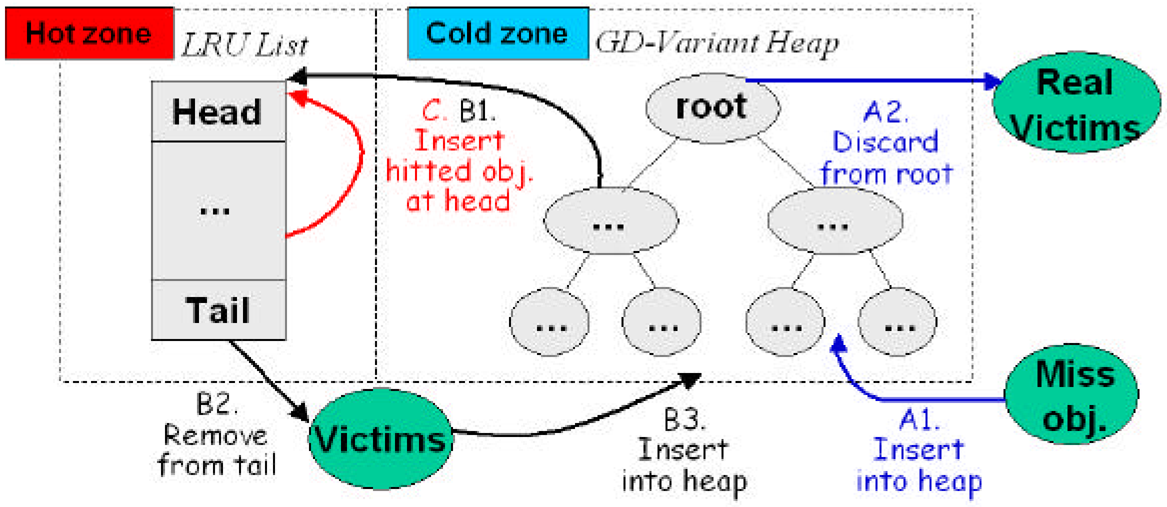


Figure 5-1. Logical flow of the hybrid algorithm

5.1.1 Maintaining object's utilization value or parameters for utilization value at LRU zone

In this hybrid algorithm, objects may be moved back and forth between two zones that use different algorithms that should maintain different parameters. Therefore we should investigate if an object's parameters for one zone should be maintained when the object is at another zone.

For an object moved from LRU zone to GD-Family zone, GD-Family zone need not maintain any extra parameter since pure LRU does not have any parameter. On the contrary, GD-Family algorithm has extra parameters that pure LRU does not have. Therefore, for an object moved from GD-Family zone to LRU zone, LRU zone should maintain utilization value or parameters for utilization value, e.g., recency, frequency, and cost. Otherwise, when an object is moved from LRU zone back to GD-Family zone, GD-Family will know only the object's size and cannot correctly re-evaluate the object's utilization value. Besides, the LRU zone of the hybrid algorithm should update the object's utilization value or utilization value's parameter when the object is hitted in LRU zone. Therefore, the LRU of our hybrid algorithm requires extra storage to maintain utilization value or utilization value's parameters. Despite the extra maintenance, the maintenance cost of LRU is still of constant order.

Moreover, some web cache algorithm should maintain parameters or statistics of all or portion of objects that are removed from cache space. In this aspect, the hybrid algorithm need not maintain object parameters or statistic data for these removed objects.

5.2 Performance evaluation

In this subsection, we present the results of the trace-driven simulations that we have conducted to evaluate the performance of our hybrid algorithm. As stated in section 3.2, other performance metrics are usually functions of DHR and BHR. Therefore, our simulations evaluate the performance in terms of only DHR and BHR. In our trace-driven simulations, we use ten traces during October 5-11, 2004 from NLANR [62]. Table 5-1 shows the characteristics of these traces. In this table, "Number of one-timer" means number of objects that are requested only once, and DHR∞ and BHR∞ are the upper bound of the ratios that are evaluated with infinite cache size. Among the ten traces, we present evaluation results of BO2, SD, UC, and SV traces for representatives. BO2 trace is chosen due to its median average request size and low traffic. SD trace is chosen due to great traffic. UC trace is chosen due to its largest number of requests, smallest average request size and highest DHR. SV trace is chosen due to its largest average request size and lowest DHR.

5.2.1 Effects of combining hot LRU and cold GDSP

This subsection compares three algorithms: GDSP [Shu99], LRU, and the hybrid algorithm to see effects of combining hot LRU and cold GDSP. GDSP, a member of GD-Family algorithms, uses a utilization function as $H(p) = L + (\frac{f(p)}{size})$ with $f_{i+1}(p) = 1 + f_i(p) \times 2^{-t/T}$ in which t is the total number of accesses between (i+1)-th and i-th accesses to object p, and T, set as 700000, is the constant for controlling aging rate. The hybrid algorithm allocates 50 percentage of cache space for hot LRU zone and 50 percentage of cache space for cold GD-Family zone, whose utilization function is the same as that of GDSP. This hybrid algorithm is therefore referred to as HY5050 according to the space ratio of the zones.

Table 5-1. Statistics of ten NLANR traces

Traces	**BO2**	**SD**	**UC**	**SV**	**NY**	**RTP**	**PA**	**PB**	**SJ**	**BO1**
Number of Requests	2,369,703	3,523,585	7,194,834	2,014,677	2,675,402	5,428,951	919,124	2,815,620	4,046,330	2,611,844
Number of Distinct Requests	1,248,637	1,714,987	1,376,750	1,320,566	1,531,000	1,710,839	549,145	1,512,566	1,980,746	1,333,331
Size of Requests	28.56G	59.08G	36.91G	48.53G	34.88G	75.2G	9.83G	35.86G	55.16G	27.40G
Number of one-timer	1,052,390	1,420,643	1,044,386	1,171,779	1,301,744	1,201,486	460,440	1,210,199	1,639,272	1,128,774
Size of Distinct Requests	14.96G	20.83G	16.89G	22.16G	18.87G	19.83G	5.84G	15.64G	24.31G	16.58G
Average Requests Size (Byte)	12053	16766	5130	24090	13039	13865	10699	12735	15655	10492
DHR∞ (%)	47.3	51.3	80.9	34.5	42.8	68.49	40.3	46.3	51.1	48.9
BHR∞ (%)	48.1	65.2	55.5	54.6	46.2	74.41	40.7	53.8	57.1	40.1

Figure 5-2 shows the simulation results of the three algorithms. In terms of DHR, all the four traces have similar relative results: GDSP slightly outperforms HY5050 by 1% and significantly outperforms LRU. In terms of BHR, the four traces behave somewhat different. In BO2 and SD traces, LRU outperforms HY5050 by no more than 2%; LRU's BHR is between those of GDSP2 and the hybrid algorithm; GDSP1 fall behind others substantially when cache size is small but improves to fall behind HY5050 by 1% when cache size is large enough to accommodate large objects. In UC trace, BHR of the four algorithms from high to low are: GDSP2, LRU, HY5050, and GDSP1 while each differs its immediate former by around 1%. Note that BHR of GDSP1 in UC trace does not substantially fall behind that of HY5050 since the average request size of UC trace is small in comparison with other traces. In SV trace, BHR of the four algorithms from high to low are: GDSP2/LRU, HY5050, and GDSP1 while each differs its immediate former by around 1.5% for small cache, HY5050 outperforms GDSP1 by 5-10% for large cache, and GDSP2/LRU outperforms HY5050 by 5-10% for large cache. The large difference in HY5050's BHR and LRU's BHR in SV trace may be attributed to the largest average request size and the lowest DHR. In case of large request size and low DHR, a missed object inserted into GD-Family zone is more probably removed soon before it is requested again.

From the simulation result, we find that HY5050 can achieve a high DHR, slightly lower than that can be achieved by GDSP. We can reasonably infer that simple cache algorithm such as LRU is enough to manage hot objects. Moreover, HY5050 can achieve a high BHR, approximately 1-2% lower than that of LRU except for traffic with large request size and low DHR. That is, HY5050, the combination of LRU and GDSP, preserves advantages of both LRU and GD-Family. To see why HY5050 has such effects, Figure 5-2 also shows the DHR and BHR of HY5050's hot zone, which are labeled as $HY5050_{hot}$. We can see that HY5050 has at least 80% of document hits (relative to total document hits) in hot LRU zone and at least 90% of byte hits (relative to total byte hits) in hot LRU zone. This result agrees with characteristics of web traces analyzed by previous research and explains why the hybrid algorithm can achieve a BHR slightly lower than that achieved by LRU.

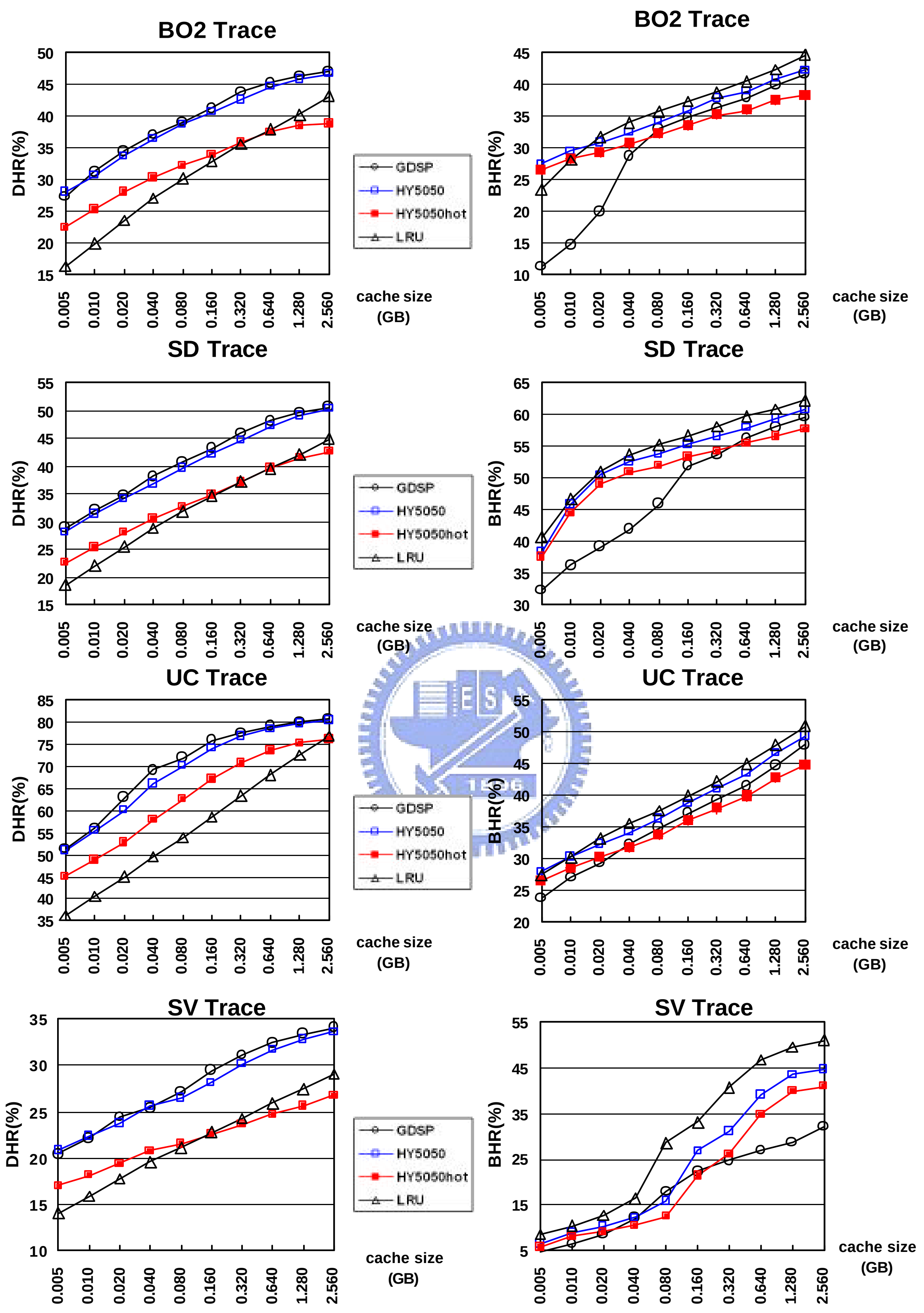


Figure 5-2. Effects of combining LRU and GDSP

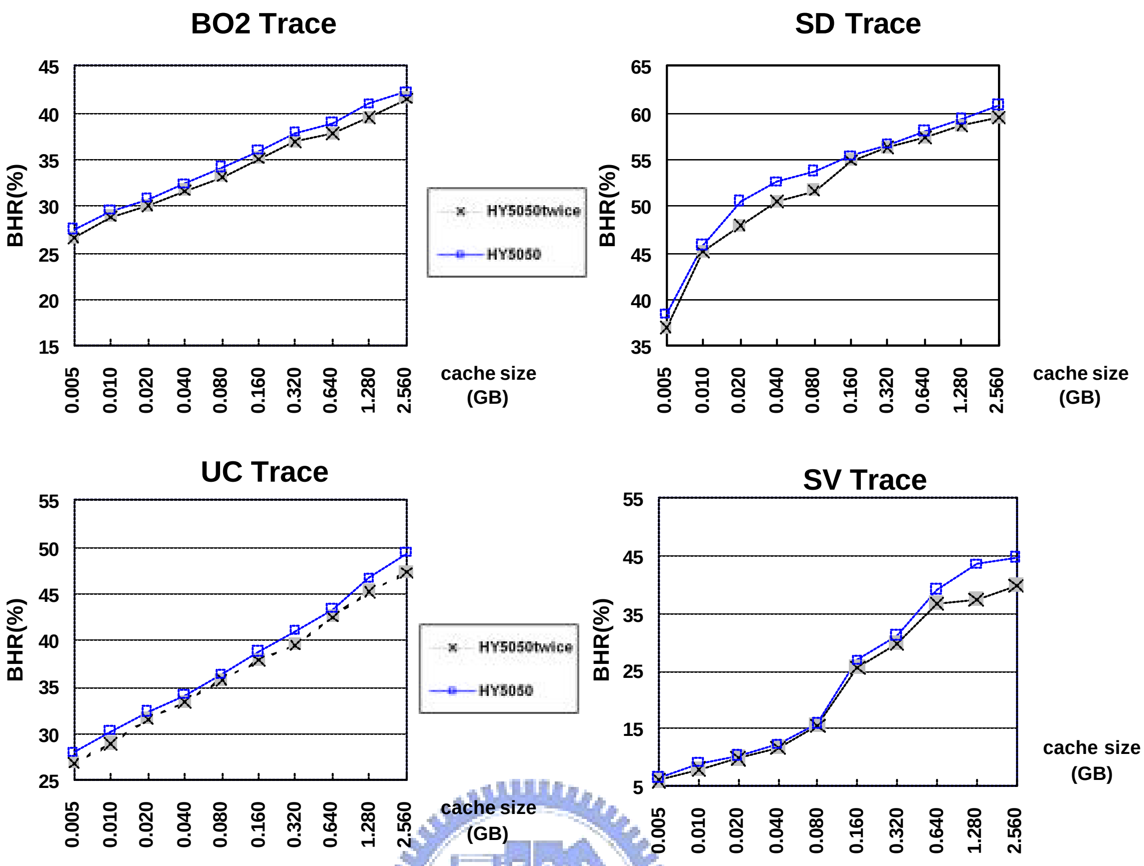


Figure 5-3. Effects of promoting an object to hot zone after twice hits

5.2.2 Effects of promoting an object to hot zone after twice hits

This subsection compares two variants of the proposed hybrid algorithms, one is HY5050 and the other is HY5050$_{twice}$ (also labeled as HY5050twice). HY5050$_{twice}$ is a variant of HY5050 that promotes an object from the cold zone to the hot zone when the object is hitted in the cold zone for the second time. The simulation results show that HY5050$_{twice}$ improve DHR over HY5050 very slightly. This improvement may be attributed to filtering of less popular objects: promoting an object after multiple times of hit can filter less popular objects and therefore more popular objects, which have higher probability of being requested again, can be kept in cache. However, the improvement is too little to be shown in a normal scale figure. Therefore, we do not show DHR of HY5050$_{twice}$ as a figure for saving space. On the other hand, the difference in BHR between HY5050$_{twice}$ and HY5050 is perceivable in a normal scale figure. Figure 5-3 shows the BHR of HY5050$_{twice}$ and HY5050. We can see that

$HY5050_{twice}$ degrades BHR by 1-2%. This degradation may be attributed to filtering of large objects: promoting an object after multiple times of hit would increases the possibility of removing a large object from GD-Family zone and therefore decrease the possibility of hitting large objects.

5.2.3 Advantage of cold GDSP over cold LRU

This subsection compares three algorithms: LRU, SLRU [36], and HY5050 to see the effects of using LRU and GDSP to manage cold objects. Figure 5-4 shows the simulation results of the three algorithms. In terms of DHR, all the four traces have similar relative results: the improvement over LRU by HY5050 is at least twice that by SLRU. In terms of BHR, SLRU outperforms LRU by about 1% except BO2 trace with small cache, which outperforms LRU by up to 5%. Since HY5050 and SLRU differs only in the cache algorithm used for managing cold zone, the difference in DHR between HY5050 and SLRU is therefore mainly incurred from performance difference between HY5050's cold zone and SLRU's cold zone. We can conclude from the simulation results that GDSP indeed manages cold objects better than LRU. Moreover, we can reasonably infer that comprehensive cache algorithms, such as GD-Family, manage non-popular or non-recently requested objects better than LRU.

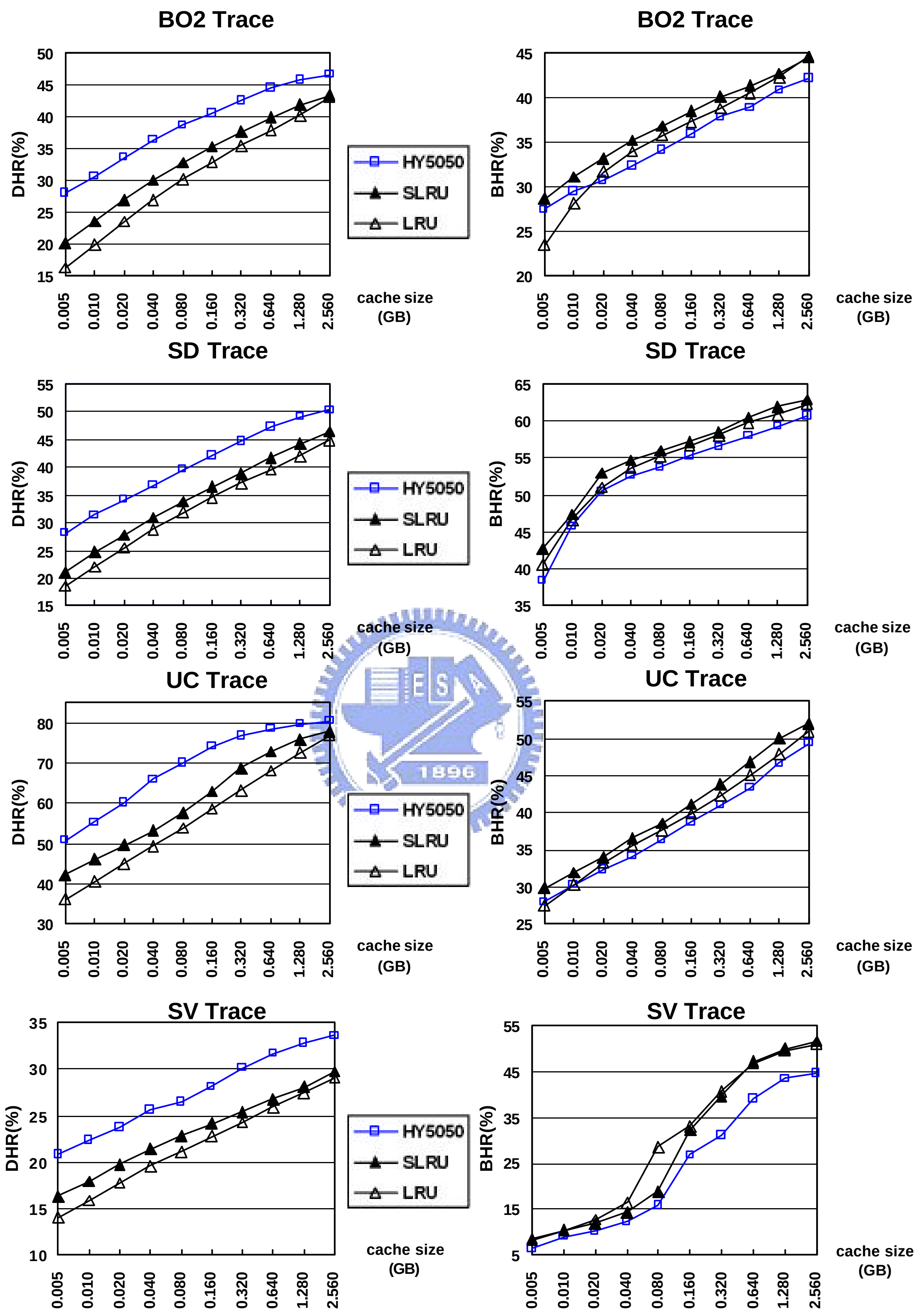


Figure 5-4. Effects of cold GDSP over cold LRU

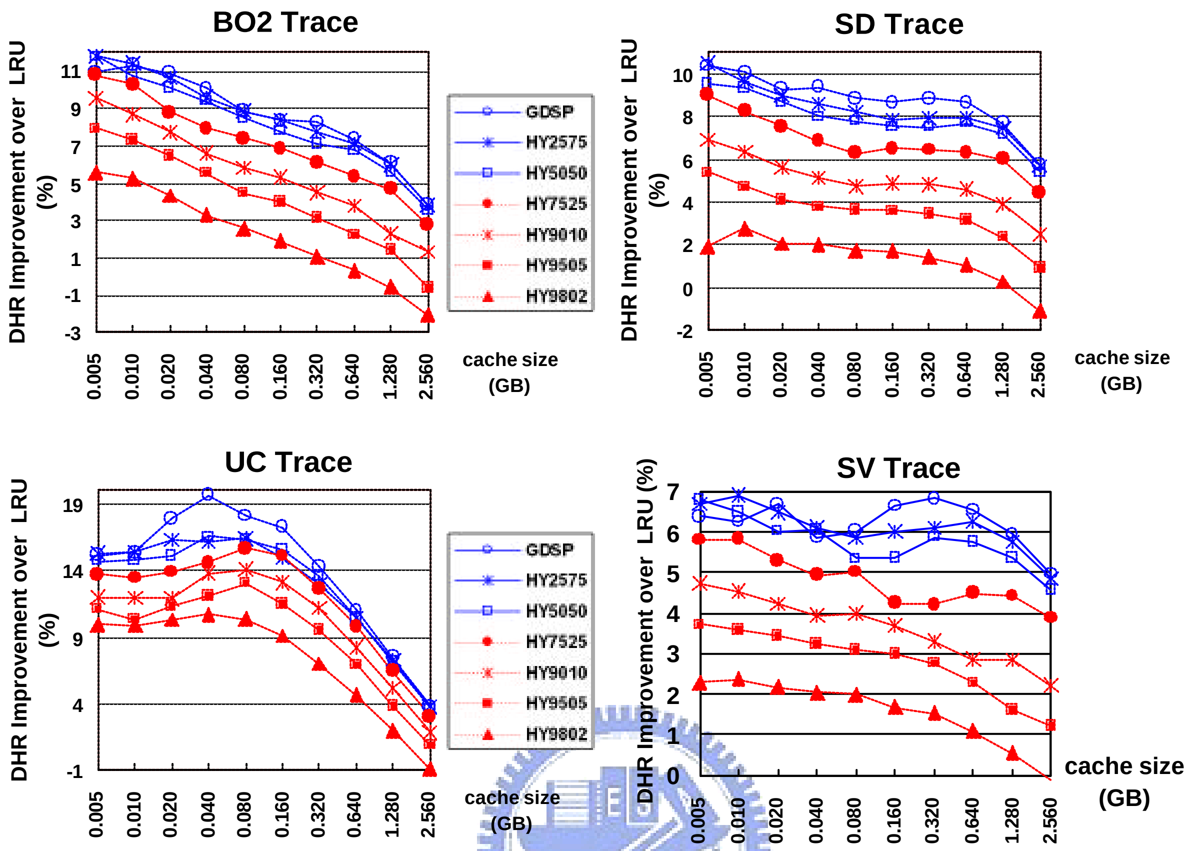


Figure 5-5. Effects of increasing space ratio of cold zone

5.2.4 Effects of Increasing space ratio of cold zone and hot zone

In this subsection, we study the impact on hit ratios of increasing space ratio of cold zone and hot zone.

Figure 5-5 shows DHR of GDSP and six variants of the hybrid algorithm: HY2575, HY5050, HY7525, HY9010, HY9505, and HY9802. Each of these variants is referred to as HY*nnmm* with the first two digits *nn* representing the space ratio used by hot LRU zone and the last two digits *mm* representing the space ratio used by cold GDSP zone. For example, HY2575 means 25% for hot LRU zone and 75% for cold GD-Family zone. Furthermore, GDSP can be regarded as a variant of the hybrid algorithm that allocates all space for cold zone. Note that Figure 5-5 is presented in the form of difference in DHR between LRU and each variant of the hybrid algorithm for purpose of clarity since DHR of each variant may differ very slightly. From Figure 5-5, we can see that DHR increases as space ratio of cold zone increases. This effect agrees with GD-Family's characteristic of being good at DHR. However, the DHR increasing effect weakens as space ratio of cold zone goes on increasing. That is, the increase in DHR made by per percentage of space decreases as space ratio of cold zone goes on increasing. This is attributed to temporal locality in web request streams. Previous analysis [9,12] on web request streams have shown that most re-requests for an object occurs in a short time interval and therefore most hits occur in hot zone (as shown in Figure 5-2). According to this characteristic, increase in DHR benefited by increasing space ratio of cold zone will be greatly cancelled out by decrease in DHR incurred by decreasing space ratio of hot zone. As a result, the weakened increasing effect and the high maintenance cost of GD-Family, $O(\log n)$, discourage allocating too great space ratio for cold zone.

Figure 5-6 shows BHR of GDSP and seven variants of the hybrid algorithm: HY2575, HY4060, HY5050, HY7525, HY9010, HY9505, and HY9802. Note that Figure 5-6 is also presented in the form of difference in BHR between LRU and each variant of the hybrid algorithm for purpose of clarity. From Figure 5-6, we can see that BHR increases as space ratio of hot zone increases from 0% to 50% (by looking at GDSP, HY2575, HY4060, and HY5050). This effect agrees with LRU's characteristic of being good at BHR. On the contrary, BHR decreases as space ratio of hot zone increases from 50% to 100% (by looking at HY5050, HY7525, HY9010, HY9505, and HY9802). The latter effect is attributed to small cold zone. When cold zone is too small, cold zone of the hybrid algorithm is more probably unable to keep a misses and newly inserted object, especially a large object long enough before the object is requested again and promoted to hot zone. Therefore, BHR decreases when cold zone is too small to accommodate sufficient temporal locality in a web request stream. Besides, the BHR increasing effect (when hot zone is between 0% to 50%) and the BHR decreasing effect (when hot zone is between 50% to 100%) weaken as space ratio of hot zone approaches 50%. That is, the increase in DHR made by per percentage of space decreases as space ratio of cold zone goes on increasing. This is also attributed to temporal locality in web request streams.

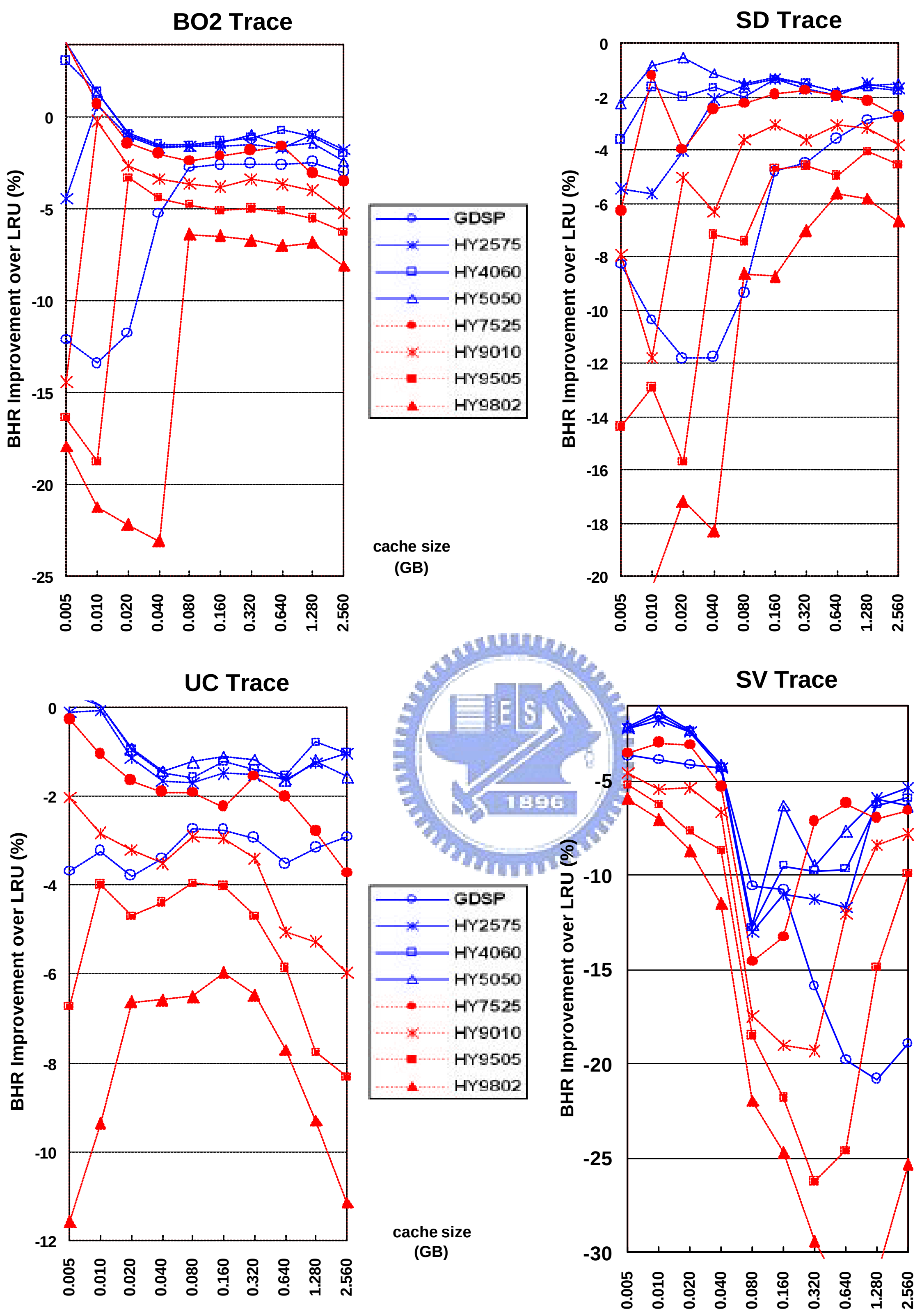


Figure 5-6. Effects of increasing space ratio of hot zone

From the simulation results shown in Figure 5-5 and Figure 5-6 and the weakened increasing or decreasing effects when space ratio of the two zone goes on increasing or decreasing, we can conclude that allocate 50% of space for each zone is a good choice. Such choice can not only retain both high DHR and high BHR but also reduce maintenance cost, compared to GD-Family. Based on this 50%-50% segmentation, a cache using the hybrid algorithm can slightly adjust the space ratios of the two zones according as the cache favors DHR or BHR.

5.2.5 Increasing number of zones

We further evaluate a three-zone hybrid algorithm consisting of an upper LRU zone, a middle LRU zone, and a lower GD-Family zone. The evaluated three-zone hybrid algorithm is equivalent to a two-zone hybrid algorithm consisting of a hot LRU zone and a cold two-zone hybrid zone. From another point-of-view, this three-zone hybrid algorithm is equivalent to a two-zone hybrid algorithm consisting of a hot SLRU zone and a cold GD-Family zone except that a missed object is inserted into the hot SLRU zone. The proposal of this three-zone hybrid algorithm is inspired by the fact that SLRU outperforms LRU in terms of both DHR and BHR. Therefore, we evaluate the performance of this three-zone hybrid algorithm to see if DHR and BHR can be further improved.

We do not show the simulation as figures for saving space since balancing of the space ratio of the three zones is more complex than in the two-zone hybrid algorithm. According to the characteristics of LRU, GD-Family, and multi-zone hybrid algorithms, increasing the space ratio of hot zone increases BHR but decrease DHR. Besides, increasing the space ratio of cold zone increases DHR but decrease BHR. Moreover, increasing the space ratio of middle zone increases BHR and slightly decreases DHR. This is attribute to the higher BHR of the middle LRU zone than that of a middle GD-Family zone. Furthermore, the combined space ratio of middle zone and cold zone significantly determine if a newly fetched object or an object removed from hot zone will be removed from the cache soon. Therefore, none of the three zones can be allocated a small space ratio otherwise the performance will significantly fall behind that of a two-zone hybrid algorithm.

According to the simulation results, we find that three-zone hybrid algorithm can slightly outperform two-zone hybrid algorithm only with limited space ratio allocation. However, it may not be efficient and practical to spend time finding such proper space ratio allocation while the improvement is slight. Furthermore, diving cache space into too many zones increases maintenance overhead for moving objects between zones. As a result, we conclude that two-zone hybrid algorithm is enough.

5.3 Summary

From the simulation result, we can see that the proposed hybrid algorithm can improve DHR of LRU by using a GD variant that is good at DHR while preserving most byte hit of LRU. The resulting DHR is slightly lower than that can be achieved by GDSP and the resulting BHR is slightly lower than that of LRU. Besides, the combination of hot LRU zone and cold GDSP zone, we have combined other algorithms in the proposed two-zone hybrid algorithms. For example, a GD variant whose utilization function is $H(p) = L + (\frac{f(p)}{log_{10}(size)})$, cold zone will not excessively favor small objects and therefore the resulting BHR is slightly higher than that of the combination of hot LRU and cold GDSP, but the resulting DHR is degraded more greatly. According to these simulation results, we infer that it maybe be a good idea to combine an algorithm very good at BHR with another algorithm very good at DHR as long as these two algorithms do not conflict.

Compare to a pure GD-Family, the combination of hot LRU and cold GDSP can achieve approximately the same hit ratios with less maintenance cost. Table 5-2 shows the maintenance cost of LRU, GD-Family, and the proposed two-zone hybrid algorithm. The first three rows show the data structure used to maintain objects and the maintenance cost for five simple operations. The last two rows show the maintenance cost of cache hit and cache miss, which are consisted of several simple operations. In comparison with the maintenance cost of

Table 5-2. Maintenance costs of LRU, GD-Family, and proposed hybrid algorithm

	LRU	**GD-Family**	**Hybrid**	
			Hot LRU	**Cold GD-Family**
Search	Hash table: $O(1)$ *	Hash table: $O(1)$ *	Hash table: $O(1)$ *	
Delete/Insert /Re-Rank	List/Stack: $O(1)$	Heap/Priority Queue: $O(\log n)$ (worst case)	= LRU	=GD-Family
Update object statistics	$O(1)$	$O(1)$	$O(1)$	
Hit	$O(1)$: Search: $O(1)$ Delete: $O(1)$ Update: $O(1)$ Insert: $O(1)$	$O(\log n)$: Search: $O(1)$ Update: $O(1)$ Re-Rank: $O(\log n)$	*Hit in hot zone=* $O(1)$ (= LRU) *Hit in cold zone=* $O(\log n)$: Search: $O(1)$ Delete for cold zone: $O(\log n/2)$ Delete for hot zone: $O(1)$ Update: $O(1)$ Insert for hot zone: $O(1)$ Insert for cold zone: $O(\log n/2)$	
Miss	$O(1)$: Search: $O(1)$ Delete: $O(1)$ Update: $O(1)$ Insert: $O(1)$	$O(\log n)$: Search: $O(1)$ Delete: $O(\log n)$ ** Update: $O(1)$ Insert: $O(1)$	$O(\log n)$ (=GD-Family)	

*: Depends on probability of hash collision

**: May perform *V* times, where *V* depends on number of victims

GD-Family, the proposed hybrid algorithm has reduced cost when cache hits in hot zone, equal cost when cache misses, and extra maintenance cost when cache hits in cold zone. This extra cost, of $O(\log n)$, is for moving objects (hereafter referred to as internal victim(s)) from the hot zone to the cold zone in order to make space for promoting a hitted object of the cold zone to the hot zone. Note that each time the hybrid cache will move different number of internal victims since web objects are of different size. For example, if a promoted object is larger than the object at the tail of the hot zone, the number of internal victims will be more than one. On the other hand, if a promoted object is smaller than the object at the tail of the hot zone, removing the object at the tail of the hot zone will make extra free space and therefore the next promoting may be done without removing any internal victims. Table 5-3 shows the average number of internal victims in our simulations with different cache size. The average number of internal victims is slightly smaller than one as cache size is small and decreases as cache size increases. Such increase may due to popular objects are often of small size.

Table 5-3. Average number of internal victims

Traces Size(GB)	BO2	SD	UC	SV	NY	RTP	PA	PB	SJ	BO1
0.005	0.99	1.00	0.99	0.99	1.00	1.00	0.99	1.00	1.00	0.99
0.01	0.99	0.99	0.98	0.98	1.00	0.99	0.99	0.99	1.00	0.99
0.02	0.98	0.99	0.98	0.97	0.99	0.98	0.97	0.98	0.99	0.98
0.04	0.97	0.97	0.97	0.95	0.97	0.98	0.93	0.97	0.99	0.97
0.08	0.93	0.95	0.96	0.92	0.95	0.97	0.87	0.94	0.97	0.94
0.16	0.85	0.93	0.90	0.88	0.89	0.92	0.76	0.89	0.95	0.89
0.32	0.76	0.89	0.81	0.76	0.79	0.86	0.49	0.82	0.93	0.79
0.64	0.55	0.77	0.65	0.60	0.60	0.76	0.20	0.67	0.85	0.59
1.28	0.22	0.57	0.44	0.49	0.31	0.59	0	0.42	0.76	0.30
2.56	0	0.21	0.15	0.28	0	0.36	0	0.09	0.42	0

However, such extra cost does not occur frequently since less than 20% of total document hits are in cold zone for small cache and the percentage of total document hits in cold zone gradually reduce as cache size grows. Therefore, the proposed hybrid algorithm does not incur extra maintenance cost frequently. For example, if the DHR in cold zone is H_c=0.05 and the DHR in hot zone is H_h=0.3, the proposed hybrid algorithm has 5% of chance to incur extra cost and 30% of chance to save cost. Compared with a GD-Family algorithm having a DHR=0.37, the relative chance of saving cost will be 0.3/0.37=81%, a significantly great chance. As a result, we conclude that the hybrid algorithm provide high DHR and BHR efficiently.

Chapter 6 Web session handoff system

This chapter describes our web session handoff system. Section 6.1 presents the system overview. Section 6.2 discusses ways to track sessions and addresses issues of session tracking in detail. Section 6.3 elaborates our session handoff protocol including ways to hand over session information. Section 6.4 shows our current implementation.

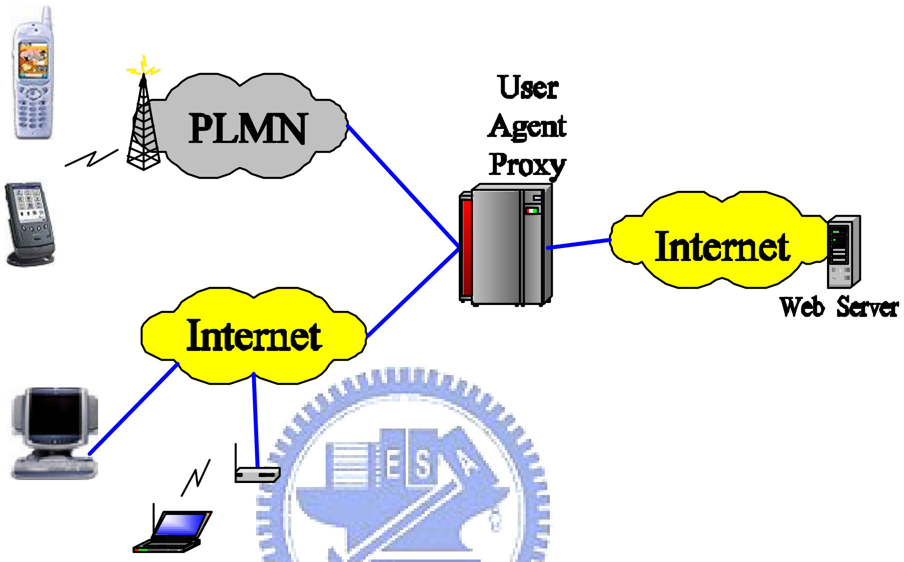


Figure 6-1. System topology

6.1 System overview

Figure 6-1 illustrates the system topology of our design. This design is extended from our previous work [63-64]. The topology is similar to the traditional web access topology that consists of user devices, proxies, and web servers. In this topology, we introduce a special purpose web proxy, called User Agent Proxy (UAP), to replace a normal web proxy. In our design, user devices and the UAP are able to perform handoff-related functionalities in addition to traditionally functionalities of web proxies. The UAP, situated between user devices and web servers, provides user authentication, session tracking, session handoff, and content adaptation functionalities for fulfilling web session handoff. As for user devices, they can be unmodified or slightly modified in comparison with normal user devices. The slight modification to user devices is to plug in a small client program that performs handoff-related functionalities such as registering sessions, collecting information for assisting the UAP in tracking sessions, and handing over sessions. User devices without modification still can perform session handoff as well, while the modification to user devices supports handoff of more session information, e.g., un-submitted user input. Intrinsically, the idea of this design is different from BSR service because a BSR repository server is not involved when a user device is browsing and does not provide functionalities of web caching and session tracking.

6.1.1 Proxy session handling

In our design, a UAP will track the session information of each proxy session for session handoff. A proxy session, in our definition, is different from a normal web session established between a user device and a web server. A proxy session starts when a browser process on a user device is authenticated with the UAP, remains when the user hands over his session to another device, and terminates when the authentication expires or the user logouts. According to this definition, a proxy session thus may contain multiple traditional sessions between several web servers in parallel.

In our design, processing of a proxy session is divided into 3 stages: registration, browsing and tracking, and session handoff:

- **Registration**: In this stage, a user authenticates himself and his current device with a UAP to register his device location, profiles, and sessions. For the user's convenience, he is allowed registering a new session by using another browser process on the same or another device while keeping previously registered sessions.
- **Browsing and tracking**: In this stage, an authenticated user device (can be a source device or a target device) sends browsing requests to the UAP, and then the UAP not only processes the browsing requests as a traditional proxy does but also tracks the proxy session. If the UAP cannot satisfy the requests with its cached objects, the UAP will forward the requests to original web servers. However the UAP needs to add user profiles as CC/PPex [65-67] headers in the requests sent by the browsers not supporting CCPP [65]. Furthermore, it needs to add in the requests sent from a target

device or in the responses replied from web servers the session information, such as cookies, which has not been handed over to the target device. On the other hand, session tracking is performed when the user device and the UAP exchange browsing requests and responses. Therefore, no extra messages are exchanged between the user device and the UAP for session tracking. The basic treatment of session tracking is to record all HTTP requests from a registered browser and to track their corresponding responses for session information. Besides, the UAP can perform further treatment to learn more session information. The description of such further treatment about proxy session tracking will be elaborated in Section 6.2.

- **Session handoff**: In this stage, the UAP, one or more source devices, and the target device co-operate to hand over the proxy session from a selected source device to the target device. First, the user switches to the target device and chooses to hand over a proxy session among those registered ones in the UAP. The UAP then combines necessary session information tracked at the UAP with assistant information collected at the source device (if available), and hands over the combined session information to the target device. Note that in this stage not all session information is handed over to the target device. Therefore, the UAP will hand over more session information to the target device when necessary. After handoff of the session information, the user can continue browsing with the same information that he viewed at the source device. The issues and protocol design of session handoff will be elaborated in Section 6.3.

6.1.2 UAP architecture

As shown in Figure 6-2, UAP provides the following functional components:

1. **User Authenticator**: User Authenticator authenticates source or target user devices and browser processes before proceeding to other UAP functionalities. It stores users' authentication information in **User DB**.
2. **Profile Manager**: Profile Manager stores user profiles such as user preference, device

capabilities, and locations of user devices in **User Profile DB.**

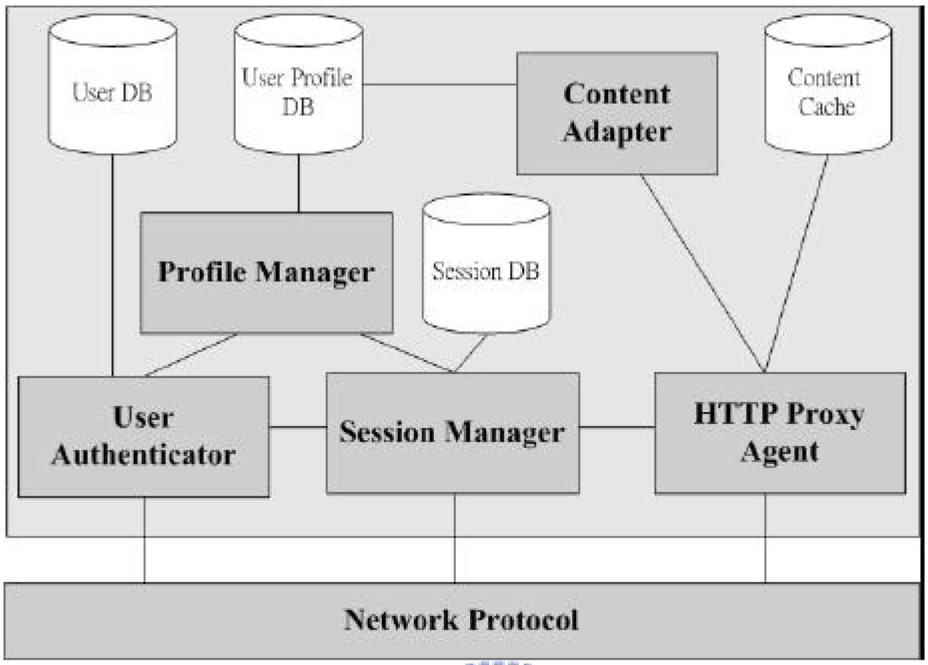


Figure 6-2. UAP architecture

3. **HTTP Proxy Agent**: HTTP Proxy Agent behaves like a traditional web proxy. It receives browsing requests from registered browsers and responses from web servers, asks Session Manager to deal with the requests and the responses, obtains the requested objects from web servers or **Content Cache**, and then replies user devices with objects adapted by **Content Adapter**.
4. **Session Manager**: Session Manager tracks proxy sessions and adjusts requests and responses during browsing and tracking stage. In addition, Session Manager will be also involved during session handoff stage. During browsing and tracking stage, Session Manager analyzes requests and responses received by HTTP Proxy Agent to track and record session information in **Session DB**. Session Manger also adjusts requests and responses by adding necessary session information in the requests and responses from/to target devices. Moreover, if browsers do not support CCPP [65-66], Session Manger will add user profiles as CC/PPex headers [67] in the requests as well. On the other hand, during session handoff stage Session Manager inter-works with source devices and target devices to accomplish session handoff.

5. **Content Adapter**: Content Adapter adapts web content to fit a user's profile during session handoff and browsing-and-tracking stages. If web servers can perform all the necessary adaptation, UAPs' functionality of content adaptation can be optional. Otherwise, multiple passes of adaptation by UAPs and web servers would be possible.

The detailed inter-working between these components in the registration, browsing and tracking, and session handoff stages is similar to that in our previous work [64], which can be found in Section 3.4 of [64].

6.2 Proxy session tracking

This section discusses ways to track a proxy session with modified and unmodified user devices respectively and also addresses issues of session tracking in detail. Before going into proxy session tracking, we first clarify usage of the following terms in this paper:

- **Session information:** It consists of any information about a session such as session history, HTTP cookies, web objects maintained by browsers, etc. Intrinsically, web session handoff means handoff of session information. Session information is called Browser State in BSR [20]. In this paper, we use the term session information to avoid confusion between session state and browser state.
- **Session state:** Session state is a subset of session information. It is the information used by web servers for identifying or recording a stateful session, e.g., HTTP cookies and other information discussed in Section 6.2.2.

Proxy session tracking in our design includes three operations: session history tracking, session state handling, and un-submitted form fields tracking.

6.2.1 Session history tracking

Session history denotes the chronological sequence of pages browsed by a particular browser process. It can be accessed by a browser's "*back*" and "*forward*" commands/buttons.

Handoff of session history enables users to traverse web pages in target devices in the original sequences as they have traversed in source devices. Therefore, it is the basis of session information.

The most intuitive way to obtain session history is from source devices. However, most browsers maintain session history in memory, which cannot be directly and easily accessed without in-depth knowledge about these browsers. Thus obtaining session history requires a modified browser or a program snooping a browser's session history. On the other hand, a UAP can track session history by recording requests and corresponding responses in the requested order. From the two solutions, we choose the latter in order to minimize modification to user devices.

The following sub-subsections further discuss three conditions about session history tracking and their respective treatments.

6.2.1.1 Multiple browser processes on a user device

For a user device executing multiple simultaneous browser processes, a UAP should be able to differentiate requests from different browser processes to separately track each proxy session. If persist connection is used (RFC2616 [15], sec8.1), the simplest way to this differentiation is according to each request's source TCP port. However, a browser's connection to a UAP may occasionally terminate and the browser process will then use another TCP port to connect to the UAP. Thus the source port is not sufficient for differentiating requests.

As a result, we propose a solution that requires modification to user devices and uses *session ids* to identify proxy sessions. Figure 4-3 illustrates the flow of using session ids. Initially, browser1 on the user device sends the UAP its first request without proxy session id. Since this request contains no session id, the UAP assigns a unique proxy session id (id1) to the proxy session for this request. Afterward, upon receiving the first request's response from server1, the UAP adds in the response the session id as an extension HTTP header and relays this response to browser1. Browser1 will then tag all subsequent requests with this id so that the UAP can recognize these requests are sent from the same browser process. Afterward, when browser2 on the same device sends its first request without proxy session id to the UAP, the UAP will assign another unique proxy session id to the proxy session for the request. As a result, different browser processes are assigned different proxy session ids, and therefore proxy sessions can be easily differentiated. In our implementation, we make no modifications to the browsers. Instead, we plug in a client program to intercept and store session ids when a user device receives responses. Consequently, our client program appends session id in the requests when a user device sends requests.

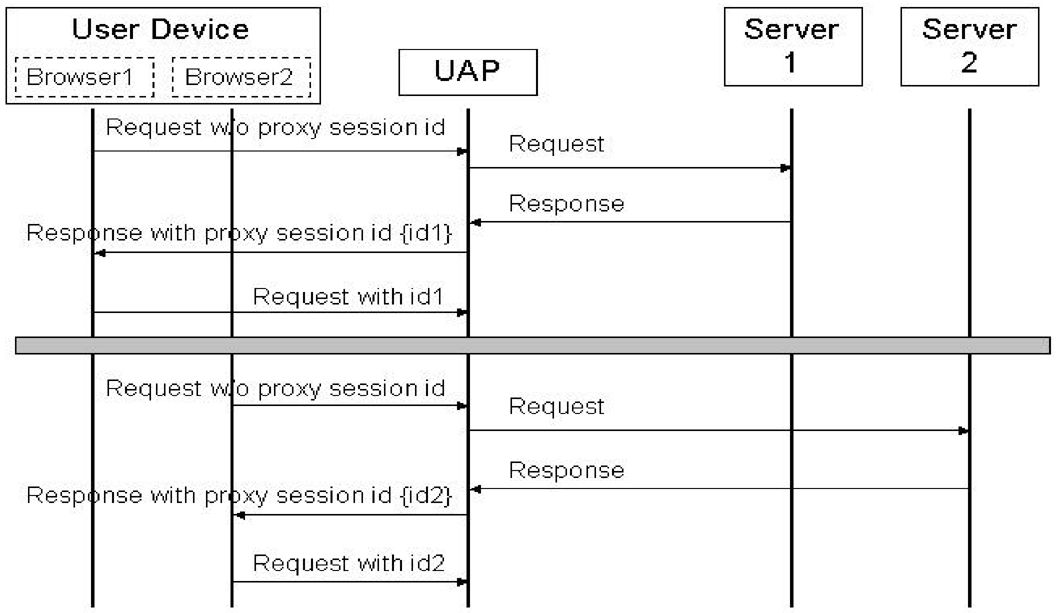

User Device
Browser1
Browser2
UAP
Server 1
Server 2
Request w/o proxy session id
Request
Response
Response with proxy session id {id1}
Request with id1
Request w/o proxy session id
Request
Response
Response with proxy session id {id2}
Request with id2

Figure 6-3. Identification of proxy session

6.2.1.2 HTML document containing other objects

HTML documents may contain other objects, e.g., frames or pictures. For such documents, a browser will send additional requests for the contained objects. In this case, the UAP should exclude these objects from session history. For example, after browsing three objects: A, B, and C sequentially, the session history should be “A, B, C” even though object B contains another two objects: B1 and B2. If a UAP has no information about B’s structure and just records all requested objects into session history, the UAP will maintain an incorrect session history as “A, B, B1, B2, C”. Unfortunately, to recognize the requested document’s structure requires parsing of the document, which will increase load to the UAP [64]. To alleviate load of UAPs, our client program will send a UAP assistant information indicating whether a requested object should be recorded into session history.

6.2.1.3 A request with a message body

An HTTP request may contain a message body (RFC2616 [15], sec4.3). For example, the message body can carry an uploaded file or user input in a form that used PUT method. The meaning and usage of the message body depend on the HTTP method and the requested URL of the request. Therefore, the message body should be included in the request if the UAP will re-request content from the original server after session handoff. Due to the specification of HTTP [15] and CC/PP [65], such recording and re-requesting can occur in our system only when the UAP can not perform content adaptation to fit for the target device and the HTTP method in the request is idempotent[2]. Table 4-1 summarizes the impact of a UAP's adaptation capability and HTTP methods on the treatment of documents and message body recording.

Table 6-1. Treatments for requests with a message body

Impacts / Condition	UAP record message body	Treatment of documents
UAP can adapt content	No	Hand over adapted cached documents
UAP can not adapt content; Method is idempotent	Yes	Hand over re-requested documents from web servers
UAP can not adapt content; Method is not idempotent	No	Hand over un-adapted cached documents

6.2.2 Session state treatment

Originally, HTTP is designed to support only stateless sessions. However, as web

[2] An HTTP request with non-idempotent HTTP method (RFC2616 [15], sec9.1.2) should not be unintentionally re-requested to an original web server since re-performing such a request may result in unwanted update to server contents. According to the specification of HTTP 1.1 [15], the methods GET, HEAD, PUT, DELETE, OPTIONS, and TRACES are idempotent and the method POST is not idempotent. In other words, requests using POST method should not be unintentionally re-performed.

browsing become popular and is widely used for more aspect of information exchange, stateful sessions with session state persisting across multiple HTTP transactions become important and necessary. With stateful sessions, a web client can send requests with necessary session state so that a web server can identify a session and deal with these requests according to the session state.

Since a stateful session is established between a source device and a web server, the target device initially will not have the session state sent between the source device and the web server. A UAP thus must track session state so that the target device can continue the stateful session after session handoff. Among the approaches [16-19] proposed to support stateful sessions on the original HTTP, our design deals with session state information supported by three extensively used approaches that use hidden form fields, cookies, and URL rewriting. Note that using cookies [17] is different from the other two since transmission of cookies in HTTP is a special portion for stateful sessions under the definition in RFC2965 [17] while the other two approaches can serve for other usage as well as stateful sessions. The following sub-subsections describe the three approaches and their respective treatment in our design.

6.2.2.1 Hidden form fields and URL rewriting

To enable a stateful session, a web server can use hidden form fields (invisible in browser) [17] or URL rewriting [19] to embed session state in web documents. After browsing such web documents, a client thus can send browsing request with necessary session state included either in the requested URL or in the request's message body. Consequently, the web server can identify the session and deal with the request according to appropriate session state.

The following shows the detail of submitting a form containing hidden form fields or using URL rewriting and the corresponding treatment in our design:

A.1 **A form using GET method and containing hidden form fields**: When submitting such a form, a client includes the hidden form fields as part of the requested URL. Consider submitting an example HTML document, which uses GET method and includes a form containing three form elements: the hidden field “session”, the select field “Year”, and the button “Submit”, as shown in Figure 6-4. The client will submit a request for **http://nctu.edu.tw/F1.cgi?session=001&Year=2003**. Note that the part “session=001” is the embedded session state.

```
<html>
<form name="F1" action="http://nctu.edu.tw/F1.cgi" method="GET">
              <input type="hidden" name="session" value="001">
              <select name="Year" size=1>
                 <option value="2003">2003</option>
                 <option value="2004">2004</option>
               </select>
              <input type="submit" value="Submit">
</form>
</html>
```

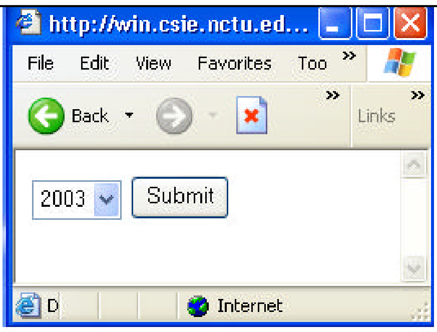


Figure 6-4. Form example using GET method and hidden form fields

A.2 **Rewriting URL**: A web server can use URL rewriting to encode session state as part of a requested URL. URL rewriting is a server-side technique that translates the path part of a requested URL to another path. For the example in Figure 6-4, if the web server translates its path “session001/F1.cgi” to “F1.cgi?session=001”, the server can send a client a different HTML document that embeds session state in the form action field as “http://nctu.edu.tw/session001/F1.cgi” without using hidden form field “session”. When submitting such a form, the client sends the server a request for “**http://nctu.edu.tw/session001/F1.cgi?Year=2003**” and the web server can automatically translate the requested URL to the real one “**http://nctu.edu.tw/F1.cgi?session=001&Year=2003**”.

As a result, the UAP performs no special treatment and can automatically record session state embedded by URL rewriting or in hidden form fields of a form using GET method since the UAP should record requested URLs for tracking session history. Besides, the UAP performs no special action and can hand over session state embedded in these two ways when handing over a session's history.

B. **A form using POST method and containing hidden form fields**: When submitting such a form, a client appends the hidden form fields in the message body of the request. Consider the same example in Figure 4-4, if the form method is POST, the client will submit a request for "http://nctu.edu.tw/F1.cgi" with a message body containing only one line "**session=001&Year=2003**".

As a result, tracking of session state embedded in this way needs the same treatment for a request's message body as stated in Section 6.2.1.3. The handoff of recorded message body will be described in Section 6.3.3.

6.2.2.2 Cookies

To enable a stateful session by using cookies, a web server associates session state with cookies or encodes session state as cookies and then sends the cookies to a client. Afterward, the client will send requests containing these cookies so that the server can deal with the requests according to the session state.

Transmission of cookies in HTTP is a special portion for stateful sessions under the definition in RFC2965 [17]. In a stateful session, when answering a client's request, a server will assign the client cookies which are associated with certain domains and some paths in these domains. The assigned cookies are then sent to the client by using "Set-Cookie2" or "Set-Cookie" HTTP header in the server's response and are stored in the client. Afterward, when the client wishes to access objects in domains and paths which have been associated with client's stored cookies, the client will send requests containing the associated cookies in "Cookie" HTTP header in requests.

As a result, to track a stateful session using cookies, a UAP should handle "Set-Cookie2" or "Set-Cookie" headers in web servers' responses, record cookies in the Session DB, and hand over recorded cookies to the target device when handing over the session. However, sending all cookies of a session to the target device may be unnecessary and slow for a mobile device. As a result, in our design the UAP will not send any cookie to the target device upon session handoff. Instead, the target device obtains necessary cookies during browsing and tacking stage. In this stage, the target device sends requests without cookies, and the UAP should append cookies in these requests and the corresponding responses if necessary. The detail will be described in Section 6.3.3.

6.2.3 Form fields tracking

Handoff of user input in form fields provides users convenience to avoid tedious form refilling. Tracking of user input includes two aspects as follows:

- **Submitted form**: As stated in Section 6.2.2.1, a browser includes user's input in form field either in the requested URL or the message body, depending on the method (GET or PUT) used by the form. Therefore, a UAP should obtain user input from the requested URL of a request for GET method and from the message body of a request for PUT method. The obtained user input is the form of "*field_name=input_value*" (for one input field) or "*field_name1=input_value1&field_name2=input_value2&…*" (for multiple input fields). Note that the HTTP requet does not contain the URL of the original HTML documentin in which the user fills the input. Therefore, the UAP parses the obtained user input into individual fields and then uses these fields and the requested URL to find out the HTML document in which the user fills the input. Afterward, the UAP combines user input into corresponding fields according to each field's name.
- **Un-submitted form**: Tracking of user input in un-submitted forms is not so easy. In normal browsing operation, input values of form fields are sent to a UAP only after form submission and therefore the UAP has no way to receive and track user input in

un-submitted form fields without assistance from modified user devices. In our design, we adopt a client program to snoop the maintained objects in browsers for the current-viewed HTML files, to find input fields, and to collect these fields' inputted values. The collected data will then be requested by the UAP during session handoff stage. The detail operation will be described in Section 6.3.1.

6.2.4 Violation of RFC's Cache-Control directives

In our design, UAPs may store responses and cookies for the purpose of session tracking. However, this storing operation may violate the specifications in RFC2616 and RFC2965 if a response is specified as non-cacheable by using "private", "no-cache", or "no-store" directives in "Cache-Control" header (RFC2616 [15], sec14.9). These directives are originally used for preventing reveal of private information when proxies reply with cached data to subsequent requests from other user devices. Thus, in order to protect the private information such as personalized data and session state, a user device and a UAP should establish security association. Fortunately, there have been many application layer, network layer, and link layer approaches that can establish the required secure association [68-72]. Besides, a UAP must separately maintain private information and must not provide the private information of a user device to other users devices. Moreover, the UAP should delete the private information when a proxy session terminates.

6.3 Stateful session handoff

Figure 6-5 shows the message flow of our session handoff protocol. In this protocol, the participants at a user device can be an unmodified browser or our client program (if it has been installed in the user device). Assume that several source devices have ongoing registered sessions tracked by the UAP. The steps of the session handoff protocol in Figure 6-5 are described as follows:

1. The target device sends a handoff request to the UAP after registering to the UAP.

2. The UAP replies a list of the user’ s registered sessions to the target device.
3. The target device requests to hand over a selected session.
4. If the source device is unreachable or without installing our client program, this step is skipped. Otherwise, from the selected source device the UAP obtains partial information of the selected session collected by our client program.
5. The UAP sends the target device necessary session information that the UAP tracks as well as that collected by our client program. In the current state, our implementation only hands over the session history in order to avoid handing over unnecessary information that is not immediately used by the target device. If necessary, other part of the session information will be sent to the target device together with subsequent browsing responses.

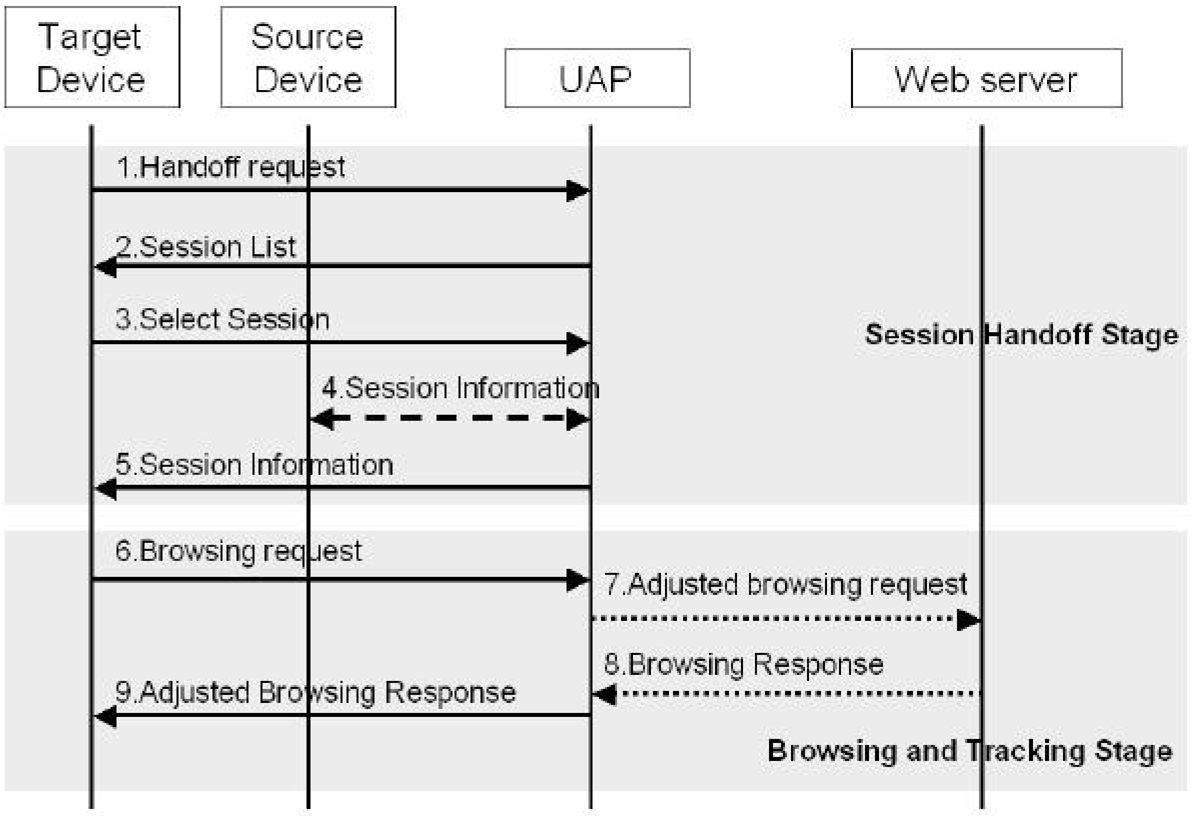


Figure 6-5. Session handoff protocol message flow

6. The target device sends the UAP a browsing request for a page in the session, e.g., the last-viewed page.

7. If necessary, the UAP adjusts the browsing request by adding the user profile as CC/PPex [65-67] header and session information containing the message body and cookies that are previously recorded by the UAP and associated with objects accessed in this browsing request. The adjusted request is then sent to the web server.
8. The UAP receives a response from the web server.

(Steps 7 and 8 can be optional if the UAP can perform content adaptation by using cached files.)

9. The UAP adjusts the response by appending necessary cookies and relays the adjusted response to the target device.

Note that Steps 6 to 9 will repeat for every browsing request sent by the target device.

The following subsections describe flow of handing over user input in form fields, web pages, recorded cookies, and recorded message bodies. These flow are similar to those in chapter 4 of [64], which also gives examples for illustration.

6.3.1 Handoff of user input in form

As stated in Section 6.2.3, the way to tracking user input in submitted forms and un-submitted forms are different. User input in submitted forms is tracked by the UAP upon receiving requests from user devices during browsing and tracking stage, even user devices are not modified. However, user input in un-submitted forms can be tracked and handed over only using modified user devices. In our design, tracking and handoff of user input in un-submitted forms are assisted by our client program. The detailed operation is as follows:

1. **During browsing and tracking stage at a source device:** The client program snoops the maintained objects in browsers and collects user input in un-submitted form fields as shown in the left of Figure 6-6. Besides, the UAP checks requested URL and received message body to obtain user input in submitted forms, parses the obtained user input into individual fields and then uses these fields and the requested

URL to find out the HTML document in which the user fills the input.

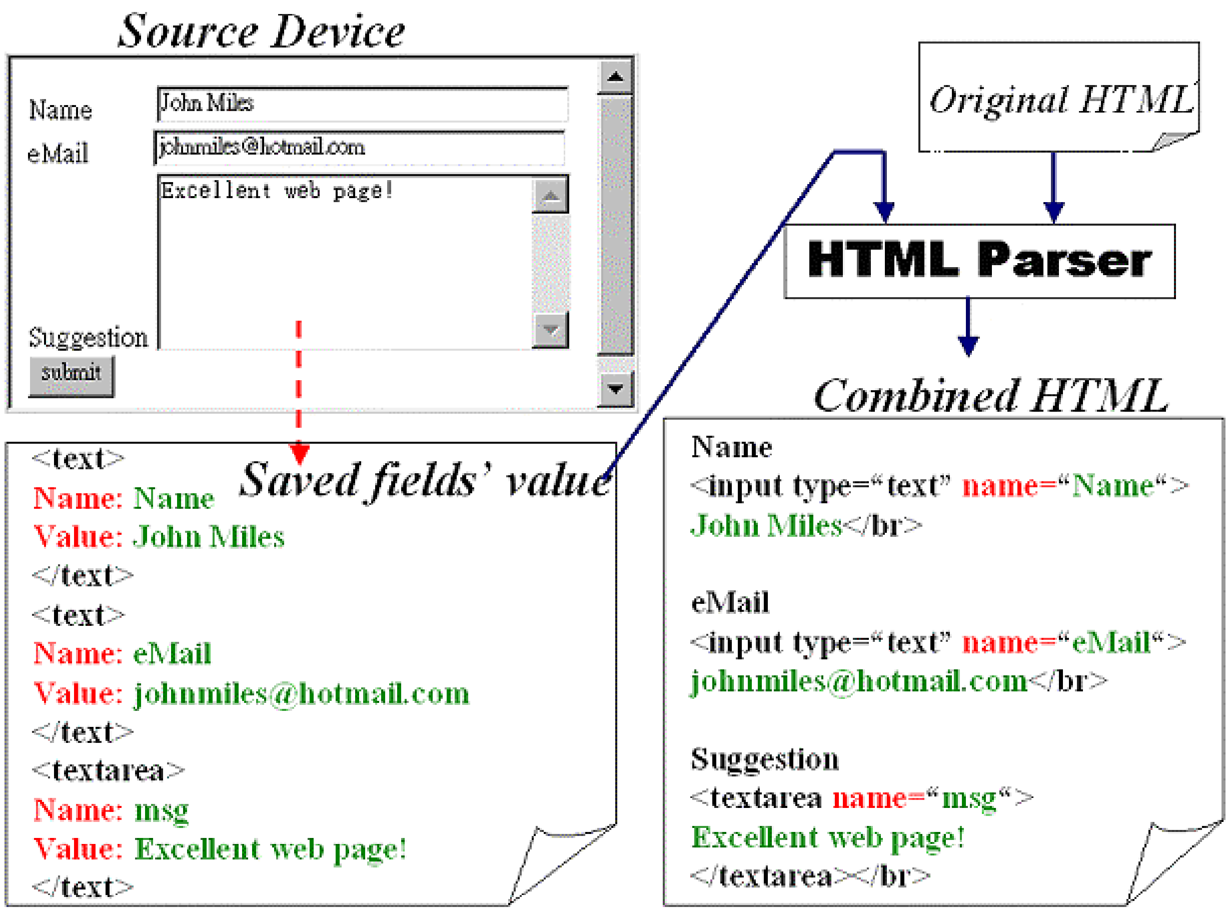


Figure 6-6. Combination of user input

2. **During session handoff stage at a target device:** The UAP obtains user input in un-submitted forms from our client program at the source device as part of session information during Step 4 of Figure 6-5, while user input in submitted forms has already been tracked the UAP.
3. **During browsing and tracking stage at the target device**: When the target device requests an HTML document containing a form (Step 6 of Figure 6-5), the UAP combines user input into the HTML document before replying the target device (Step 9 of Figure 6-5). The combination flow is shown in the right of Figure 6-6. The UAP obtains the HTML document from the UAP's Content Cache or the original web server, parses the HTML document, and combines user input into

corresponding fields according to each field's name.

Note that the program code for snooping different browsers' maintained objects is browser-dependent. In the current state, we implement only the client programs for Microsoft's IE of desktop Windows and Pocket PC.

6.3.2 Handoff of a web page

Intuitively, a web page can be handed over in form of its content or URL. Handing over real content data is a convenient and fast choice since the browser on the target device is unnecessary to send another browsing request in addition to the handoff request. However, in this way our client program must save the content data of the page and objects contained in this page as local files, parse the page, change references of the contained objects in the page into locally saved files, and notify the browser to open the locally saved file of the page. As a result, the browser will display the locally saved file's name as the page's URL instead of the original network URL. Furthermore, the extra parsing and reference changing will generate extra load to user devices.

On the contrary, handing over URL is unnatural and slower since a browser needs to send additional requests to obtain real content data from the UAP. These requests result in extra messages exchanged between a user device and the UAP. However, this way makes the browser on a target device display the original network URL as the page's URL and frees the target device from the extra parsing and reference changing. Thus, our design adopts handoff of a web page in the form of its URL.

In our design, handoff of a web page occurs only at the end of session handoff stage, while the content of this page is requested during browsing and tracking stage. At Step 9 of Figure 6-5, the UAP sends the target device the list of URLs in the session history and also redirects the target device to open the last-viewed page that we reasonably assume the user would like to view first after session handoff. With the redirection, the user device will automatically request the last-viewed page from the UAP without the user's instruction (at Step 6 of Figure 6-5).

6.3.3 Handoff of recorded cookies and message bodies

As we have stated previously, to avoid handing over session information that is not immediately used, our session handoff protocol does not send a target device cookies and message bodies associated with a session and recorded by a UAP during session handoff stage. Instead, we send a target device session information associated with a particular object only when the target device requests this object.

The handoff of recorded cookies and message bodies of a session is performed during the browsing and tracking stage of Figure 6-5. At Step 6, the target device sends a browsing request that may not contain the cookies and message body necessary for accessing the requested object. At Step 7, if necessary, the UAP appends in the browsing request the cookies and message body for accessing the requested object and forwards the adjusted browsing request to the web server. At Step 8, the web server replies the requested object to the UAP. Steps 7 and 8 can be optional if the UAP can perform content adaptation by using cached files. At Step 9, the UAP appends in the received response the cookies that it appends at Step 7 and that are associated with the object in the received response, and then forwards the adjusted response to the target device. Afterward, subsequent requests from the target device will contain these cookies. Therefore, the UAP need not append these cookies again until the session is handed over again. It should be noted that at Step 9 a recorded message body is not included in the adjusted response since the standard HTTP protocol cannot afford to send a message body in the reverse direction. Although the target device cannot obtain the message body, the target device is probably able to proceed to browse without the message body since the same browsing request of the target device can be satisfied by the cache of the target device or the UAP.

6.3.4 Comparison with BSR

Table 6-2 summarizes differences between BSR [20] and our approaches with and without modification to user devices. Our approach adopts a proxy-based approach that is inherently different to the client-based approach adopted by BSR. Besides the basic differences between proxy-based approaches and client-based approaches, our approach has two main advantages over BSR service:

- **Handoff Action**: BSR service requires source devices to explicitly save browser state. In contrast, our design has the flexibility that a user can perform session handoff when he leaves his source device without performing a saving action.
- **History and Cookies Handoff**: The style of handoff performed by BSR service requires transmission of much session information and page content from a source device to a BSR repository server and then from a BSR repository server to a target device. Such transmission may be unnecessary since a user may not re-visit all the web pages of a session. Although BSR service provides adjustable limit on the history size, such limitation may be impractical since a user may need different limits for different sessions. In order to reduce network traffic, our protocol only hands over URLs of a session so that a user can follow the URLs to re-visit pages of the session.

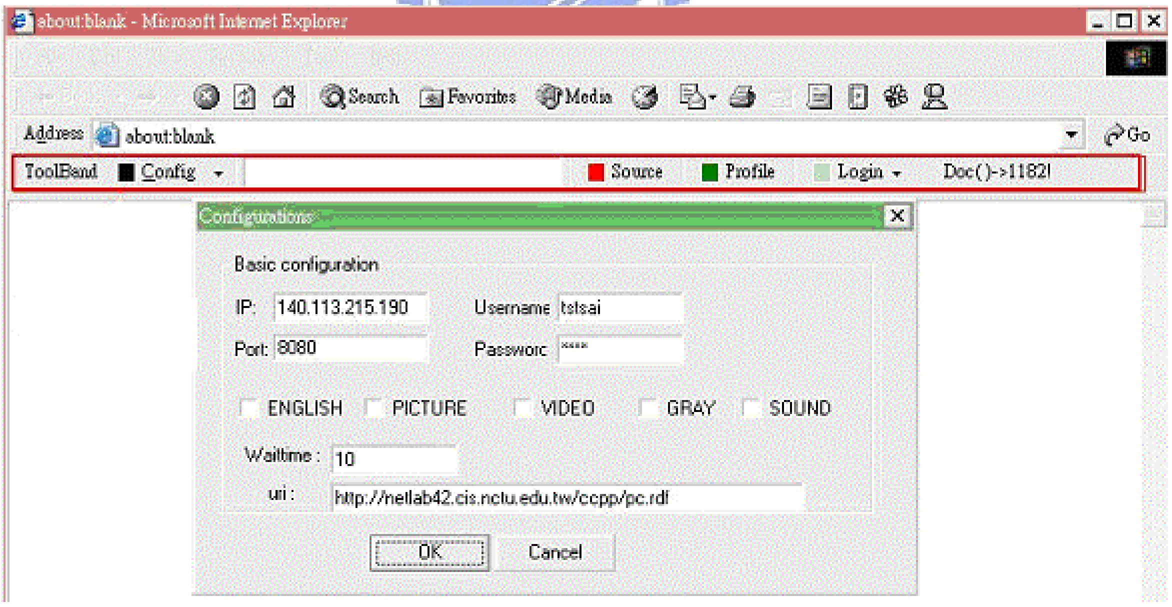
Table 6-2. Differences between BSR and our approaches

Approach / Item	BSR	Our approach	
		Without modification	**With modification**
Modification to user devices	Substantially modified browser	No	Install a small client program
Session Registration	Unnecessary	By accessing UAP's web pages	By accessing UAP's web pages; Or by the client program
Tracking Session Information	Unnecessary	By UAP; Unable to track un-submitted form fields	By UAP; Un-submitted form fields tracked by client program
Handoff Action	Source saves to BSR repository and then target retrieves from UAP	Target retrieves from UAP	UAP retrieves un-submitted form fields from source; Target retrieves from UAP
History Handoff	Directly loaded into browser	Sent as a list of web pages	Sent as a list of web pages
Web pages Handoff	Content is directly loaded into browser	UAP redirects browser to open URLs	UAP sends URLs to client program and then browser opens the URLs

6.4 Implementation and Example

This section shows our implementation. We have implemented a UAP on a Microsoft' s Windows 2000 platform and client programs for performing session registration and handoff for Microsoft' s Windows PC and Pocket PC PDA. Besides, Our system has the flexibility that user devices without our client programs can perform session registration and handoff by accessing web pages on the UAP.

The client program for Microsoft' s Windows PC is implemented as a plug-in in IE, which is the rectangle toolbar under the address bar in Figure 6-7. Using the program, users can perform session registration and handoff directly in IE' s main window. The configuration window of this program is shown in the center of Figure 6-7. This window enables a user to input his username, password, IP address of a UAP, port number on the UAP for accessing the UAP's functionalities, basic user preferences for adaptive browsing (the options: ENGLISH, PICTURE, VIDEO, GRAY, SOUND, and Waiting-time), and URL of user profile.



Basic configuration

IP: 140.113.215.190 Username tstsai

Port: 8080 Password ****

☐ ENGLISH ☐ PICTURE ☐ VIDEO ☐ GRAY ☐ SOUND

Waittime : 10

url : http://nellab42.cis.nctu.edu.tw/ccpp/pc.rdf

Figure 6-7. PC client program

Figure 6-8 shows the client program for Microsoft's Pocket PC PDA. This program provides the same functionalities as a PC's client program except that the PDA client program is a standalone one instead of a plug-in.

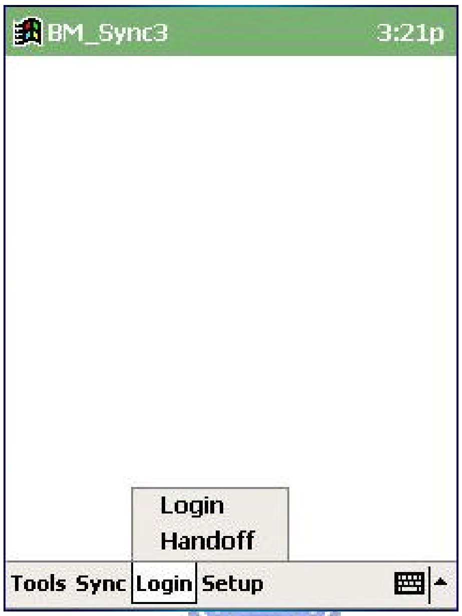
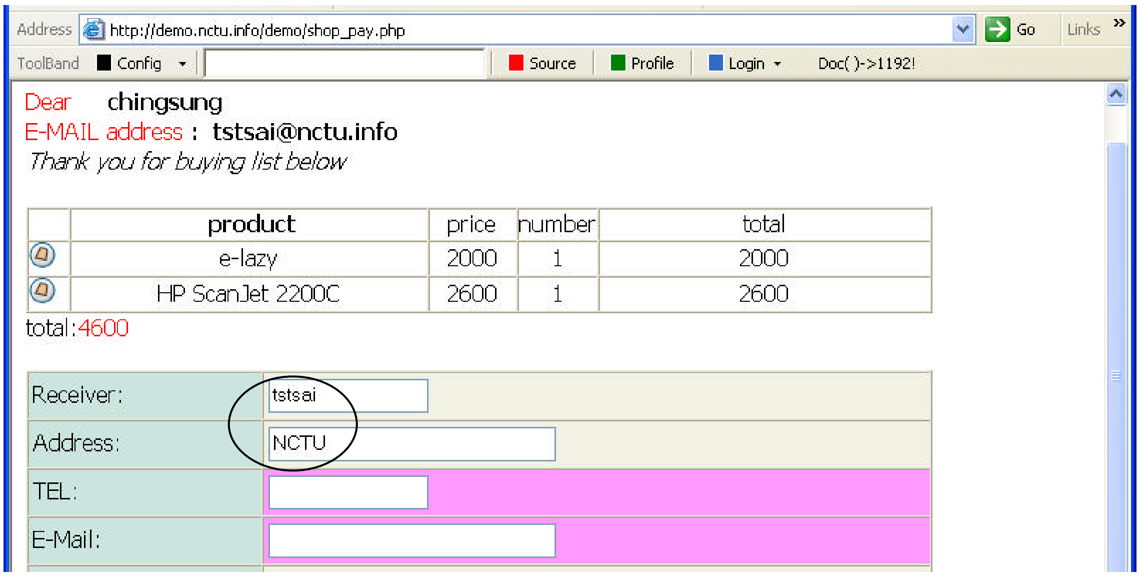


Figure 6-8. Pocket PC client program

The following demonstrates handing over a session for online shopping by using the implemented system. As shown in Figure 6-9, a user first uses a PC to visit an online shop, places his order, and inputs two form fields: *Receiver* and *Address* (the circled two fields).



Dear chingsung
E-MAIL address : tstsai@nctu.info
Thank you for buying list below

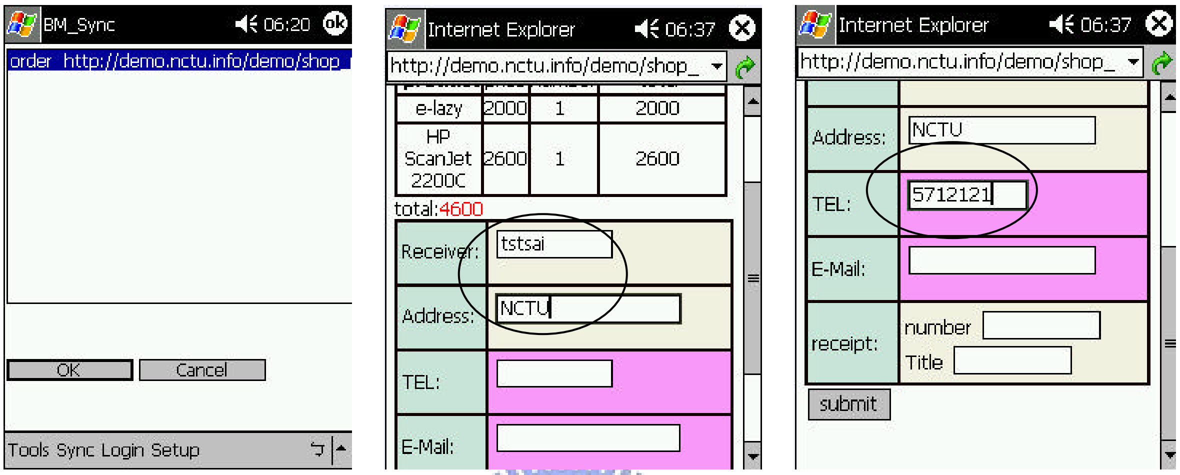
	product	price	number	total
	e-lazy	2000	1	2000
	HP ScanJet 2200C	2600	1	2600

total:4600

Receiver:	tstsai
Address:	NCTU
TEL:	
E-Mail:	

Figure 6-9. A session at a PC

Afterward, the user uses the client program on the PDA to hand over the session (Figure 6-10(a)), and resume the session at the last-viewed page with user input in the *Receiver* and *Address* form fields (the circled two fields) also handed over (Figure 6-10(b)). The user then inputs another form field: *TEL* (Figure 6-10(c)).



order http://demo.nctu.info/demo/shop

http://demo.nctu.info/demo/shop_

e-lazy	2000	1	2000
HP ScanJet 2200C	2600	1	2600

total:4600

Receiver:	tstsai
Address:	NCTU
TEL:	
E-Mail:	

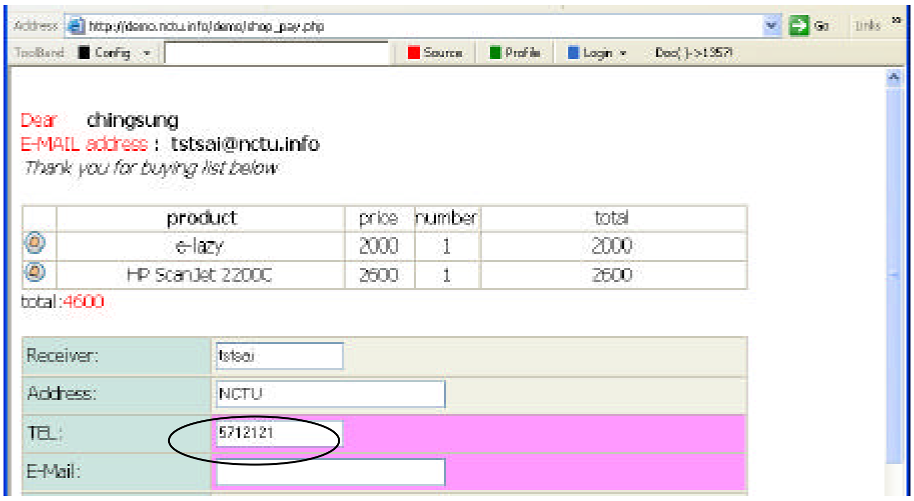
http://demo.nctu.info/demo/shop_

Address:	NCTU
TEL:	5712121
E-Mail:	
receipt:	number Title

(a) (b) (c)

Figure 6-10. Session handoff to PDA

Finally, the session is handed over back to the PC with the user input in the *TEL* form field handed over (Figure 6-11).



http://demo.nctu.info/demo/shop_pay.php

Dear chingsung
E-MAIL address : tstsai@nctu.info
Thank you for buying list below

	product	price	number	total
◎	e-lazy	2000	1	2000
◎	HP ScanJet 2200C	2600	1	2600

total:4600

Receiver:	tstsai
Address:	NCTU
TEL:	5712121
E-Mail:	

Figure 6-11. Session handoff back to the PC

6.5 Summary

In this chapter, we propose a web session handoff system that adopts a proxy-based approach with optional assistant information from user devices and discusses ways to track and hand over a variety of session information in detail. Using proxy-based approaches for session tracking and handoff has several advantages over using client-based approaches or server-based approaches. The advantages include less modification to user devices, practicability, and fault tolerance. During handoff, a source device may be unreachable due to lack of battery or wireless connection. In this case, a proxy-based approach can hand over a session because this session is tracked by a proxy while the user browses. Besides, in order to hand over session information that cannot be tracked by a proxy, we let the less modified user devices collect session information for assisting the proxy in session tracking. We have implemented a special proxy – UAP and client programs for PC and PDA to fulfill the session handoff system. The implementation can successfully hand over not only stateless but also stateful sessions between homogeneous or heterogeneous user devices.

Chapter 7 Topology-and-Direction-aware Fast Handoff for 802.11 Networks

7.1. Overview of the proposed approach

Observe that AP discovery procedure poses a significant delay in handoff and reassociation establishment does not dictate all of the parameters gleaned through AP discovery. Besides, candidate APs with which an MN pre-authenticates or to which an MN proactively distributes keys can usually exclude those APs opposite the movement direction of the MN. Therefore we propose a topology-and-direction-aware approach that resolves a smaller set of candidate AP before handoff. In the proposed approach, there is a location server that provides MNs with topology information, including positions of APs and parameters for AP (re)association, e.g., the timing and capability information [22]. With the topology information, an MN, being able to determine its moving trend and/or position, can resolve a reduced set of candidate APs when the MN is moving. With the reduced candidate AP set, the MN can pre-authenticate with or proactively distributes keys to less APs. Besides, the MN can in advance select a target AP from the candidate AP set. As a result, the MN can directly reassociate with an AP without performing time-consuming AP discovery since the parameters for AP (re)association, which are themselves mostly invariable, has been learned from the location server.

7.1.1 Topology information and location server

In the proposed approach, an MN should be aware of topology information in order to infer imminent handoff and re-solve candidate APs. In general, the topology information includes AP topology and information about other environment objects, e.g., stairs, walls, floors, and turning corners. Among the topology information, AP topology includes positions of APs, re-association relationship between APs, and parameters of APs. The positions of APs and re-association relationship between APs are used for resolving an MN's nearby candidate

APs. The parameters of APs refer to information elements predefined in beacon frames for reassociation setup. As for information about other environment objects, it is used for helping confirm re-association relationship between APs. For example, an MN can include a nearby upstairs AP in candidate APs if the MN knows there are nearby stairs to the upstairs. Another example is a turning corner, which often causes a sudden drop or even a complete loss of radio signals. In this circumstance, an MN with foreknowledge of the topology information is aware of entering a lossy area, and selects (by default) certain APs around the corners.

As for the way to represent positions of APs and other objects, it depends on the environment where an 802.11 network is deployed and on the way to derive the moving trend and/or position of an MN. For example in an indoor environment in which movement of an MN is regular and re-association relationship between APs is simple, though precise position of an MN cannot be obtained, the MN can still derive its moving trend and candidate APs according to the tendency of variation of signal strength between the MN and APs. Therefore, logical representation as simple as that used by Neighbor Graph [28-29,53] may afford. On the other hand, in an outdoor open space environment in which movement of an MN is irregular, physical location as precise as longitudinal and latitudinal coordinates may be necessary if the MN is equipped with a GPS device.

The location server maintains information on neighbor APs manually or in an automated fashion as with [73]. Without loss of generality, we co-situate the location server at the domain authentication server. Therefore, the location server can perform pre-authentication, proactive key distribution [28], or context transfer [53] on behalf of an MN. The location server may respond to an MN's request for topology information on demand. It is at the server's discretion whether to notify an MN of all objects in a wireless network or only nearby objects. In the latter case, an MN should request for topology information when the MN enters the coverage of an AP for the first time. With knowledge of a larger number of objects, an MN demands more storage space but fewer network session start-ups and tear-downs for environment information transfer.

7.1.2. Measuring MN's position and moving trend

To resolve candidate APs for upcoming handoff, an MN measures its moving trend and/or position somehow, e.g. using GPS, sensor network, or other localization techniques. To reduce the load for the measurement, the measurement can be performed at a time interval if the MN's received signal strength (RSS) from the current AP has dropped below some certain threshold.

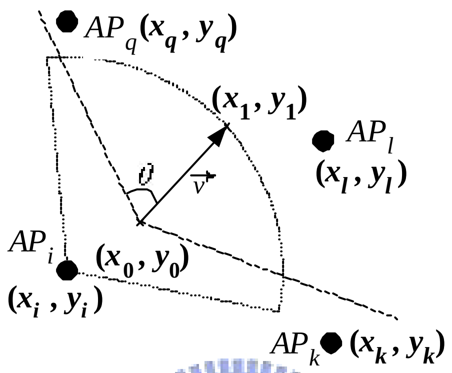


Figure 7-1. Deriving candidate APs based on movement behavior

Similar to that stated in last subsection, the necessity of measuring an MN's position and the representation of the MN's position depend on the environment where an 802.11 network is deployed. If the MN can derive its moving trend and candidate APs without knowing its position, it is not necessary to measuring the MN's precise position. For example, in an indoor environment with simple AP topology and reassociation relationship, an MN can still derive its moving trend according to the tendency of variation of signal strength between the MN and MN's neighbor APs. On the other hand, in an outdoor open space environment in which movement of an MN is irregular, precise physical location measured by GPS equipment may be necessary. For example, the displacement between two successive estimates of an MN's positions, e.g. $(x_1 - x_0, y_1 - y_0)$ in Figure 7-1, gives a vector $\vec{v}$ representing the MN's precise movement direction. Besides using two successive position estimates to find $\vec{v}$, we can use n ($n > 2$) estimates to find $\vec{v}$ instead. This can reduce the adverse effect of sudden or continuous change of moving direction, for example, an MN bouncing quickly beside the border between the coverage of two APs. Provided that an MN is successively positioned at

(x_0, y_0), (x_1, y_1), …, and (x_{n-1}, y_{n-1}). A least-square line f can be employed to fit these data points. In consequence, $\vec{v}$ is represented as a vector from $(x_0, f(x_0))$ to head $(x_{n-1}, f(x_{n-1}))$, reflecting the overall tendency of motion.

7.1.3. Selection of candidate APs

Figure 7-1 illustrates the concept of deriving candidate APs by using an MN's movement trend. In this figure, an MN is moving off it current AP AP_i. With knowledge of topology information, including AP topology, the precise motion vector $\vec{v}$ or a rough movement trend suggests a set S of APs to which the station may likely roam in the near future. To tolerate mobile direction uncertainty, the MN may select S to encompass neighbor APs within the open area stretched by two rays originating from (x_0, y_0), either with angular displacement θ from $\vec{v}$. For instance, AP_l and AP_q in Figure 7-1 form S. The value of θ varies for different MNs since each MN may have its own mobility patterns. If a station tends to make rectilinear migration, small θ suffices. A larger θ up to $180°$ caters for an MN that roams randomly.

Supposing that precise position is available and an MN is moving off its current AP_i. One way to generate a set S of candidate APs is formulated as follows:

```
var S : set of APs init Ø;
var Nbrs: set of APs init neighbor APs of AP_i;
begin for all j ? Nbrs do begin
    if $\frac{\vec{v}\cdot(x_j - x_0, y_j - y_0)}{|\vec{v}|\sqrt{(x_j - x_0)^2 + (y_j - y_0)^2}} > \cos q$
    then $S := S \cup \{j\}$;
    end
end
```

Notice that the inner product of two vectors, say $\vec{v}$ and $\vec{w}$, is mathematically defined as $\vec{v}\cdot\vec{w} = |\vec{v}||\vec{w}|\cos q'$, where q' is the angle between the vectors. Here $\vec{w}$ denotes any vector from point (x_0, y_0) to each of its neighbor APs. A geometric interpretation of a larger $\cos q'$ (than given $\cos q$) is that $\vec{w}$ conforms closer with $\vec{v}$ in terms of direction. Hence, for *S* we single out each *AP_j* such that $\frac{\vec{v}(x_j - x_0, y_j - y_0)}{|\vec{v}|\sqrt{(x_j - x_0)^2 + (y_j - y_0)^2}} > \cos q$.

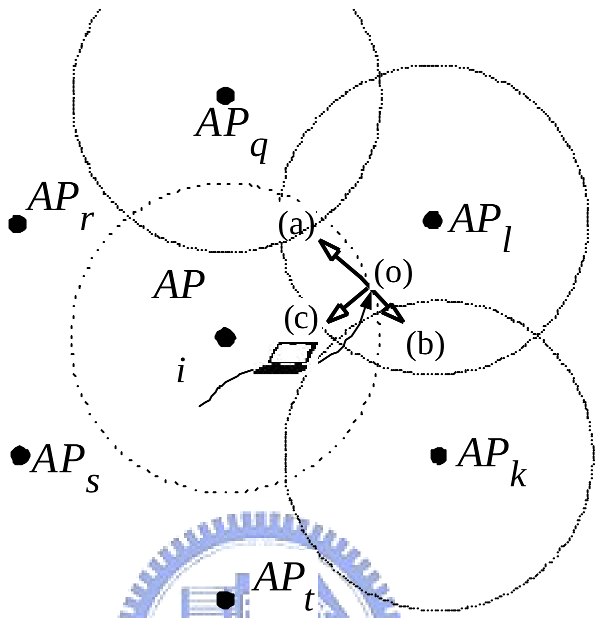


Figure 7-2. Determining candidate APs under different mobility condition

Besides, the generation of *S* can consider the topology information and the RSS between an MN and an AP. For example, an MN can include a nearby upstairs AP in candidate APs if the MN knows from the topology information there are nearby stairs to the upstairs. Figure 7-2 illustrates another example in which generation of *S* considers the distance and/or the RSS between an MN and an AP. In Figure 7-2, the MN moves along the arrowed curve to the position labeled with (o) and therefore uses an ordinary q to generate *S* which consists of AP_l and AP_q. However, since position (o) is close to the service area of AP_k and the RSS between the MN and AP_k is strong, *S* should include AP_k in case the MN suddenly turns to the direction labeled with (b). On the other hand, since position (o) is far from the service area of AP_q and the radio signal of AP_q cannot reach the MN, the MN, even turns to the direction labeled with (a), will not immediately enter the service area of AP_q and therefore *S* need not include AP_q. In the example shown in Figure 7-2, the MN can take leisure time to perform site survey in order to obtain the RSS before or upon generating *S*. The consideration of RSS or distance also applies in case the MN reverses its direction to that labeled with (c). In this case, since AP_s is far from the MN and the RSS between the MN and its current AP may increase, *S* need not include AP_s.

7.1.4 An example algorithm

The main concept of the proposed Topology-and-Direction-aware approach is using topology information, MN's moving trend, and MN's position to resolve a set of candidate APs or a target AP. In reality, its implementation may vary depending on several factors, e.g., the environment where an 802.11 network is deployed and the way to derive the moving trend and/or position of an MN. Besides, operations such as measuring moving trend and generating candidate APs may be performed either by MNs or network hosts. In Figure 7-3(b), we show an example flow to illustrate how the proposed approach can be implemented in an indoor environment as that shown in Figure 7-3(a) which consists of three APs (AP_k, AP_q, AP_k) in passages, AP_i in a rooms, and AP_l in another rooms. In this environment, precise position and moving direction of an MN cannot be measured. However, the MN can measure its moving trend and candidate APs according to the tendency of variation of RSS between the MN and the APs. For example, an MN, moving along a passage, will find that the RSS between the MN and a first neighbor AP has an increasing tendency and the RSS between the MN and a second neighbor AP has a decreasing tendency and then knows that it is approaching the first neighbor AP and leaving the second AP.

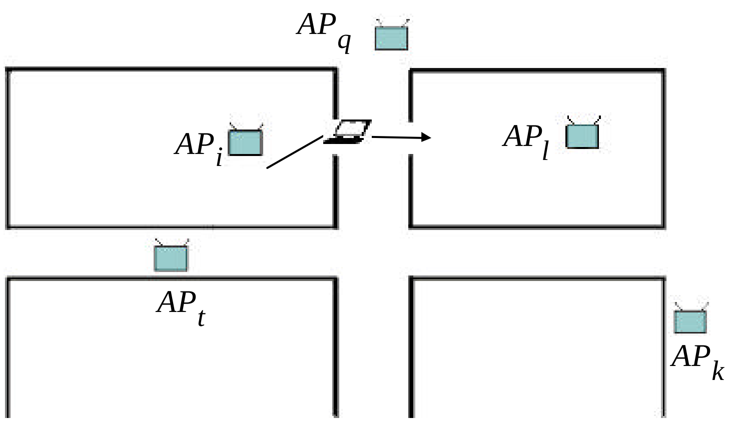
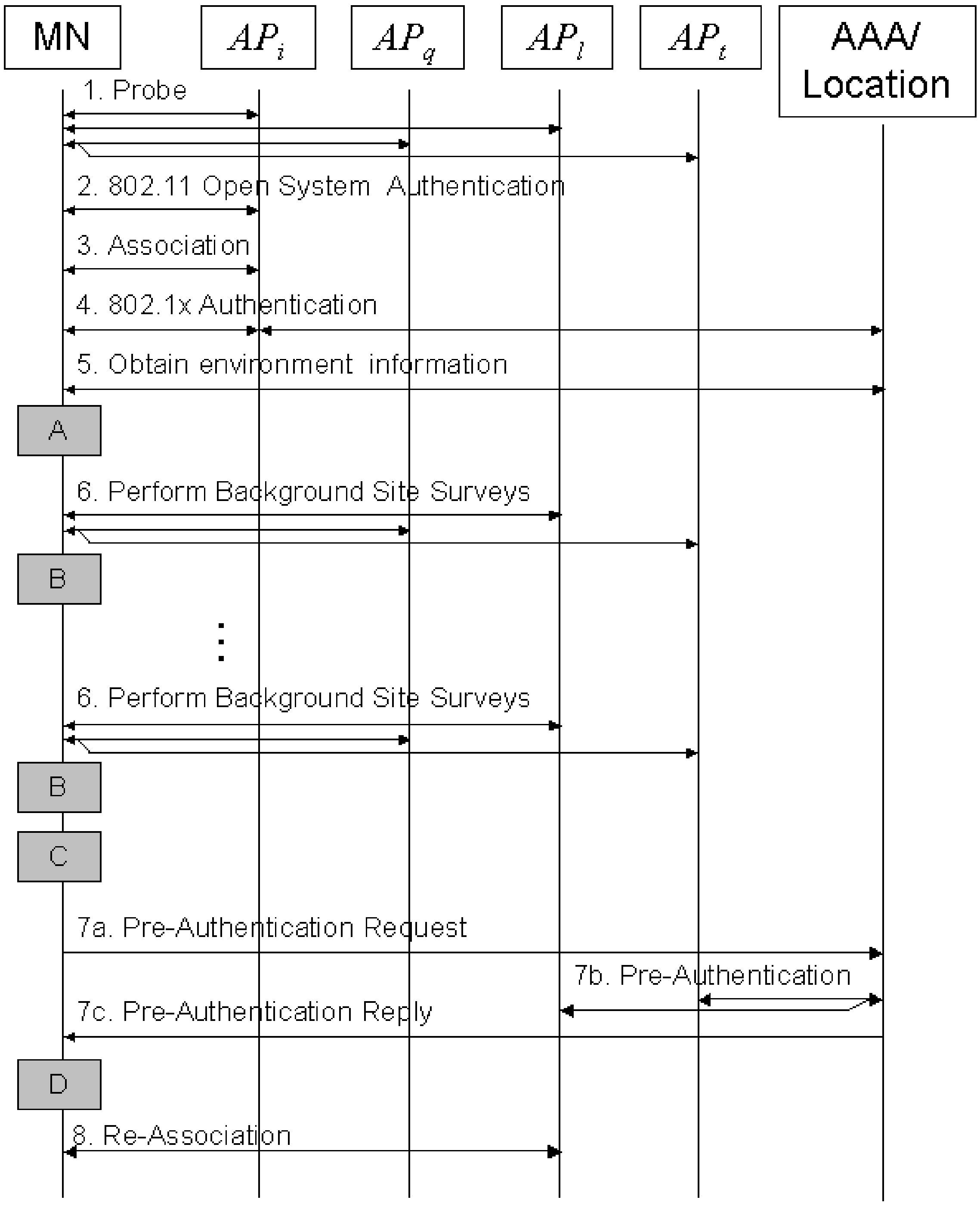


Figure 7-3(a). Example indoor environment


MN
AP_i
AP_q
AP_l
AP_t
AAA/ Location
1. Probe
2. 802.11 Open System Authentication
3. Association
4. 802.1x Authentication
5. Obtain environment information
A
6. Perform Background Site Surveys
B
6. Perform Background Site Surveys
B
C
7a. Pre-Authentication Request
7b. Pre-Authentication
7c. Pre-Authentication Reply
D
8. Re-Association

A: Site Survey Criteria Satisfied B: Generate Candidate APs
C: Pre-Authentication Criteria Satisfied D: Handoff Criteria Satisfied

Figure 7-3(b). Protocol flow for an indoor environment

The protocol flow in Figure 7-3(b) is explained as follow:

1. The MN gets powered in the room serviced by AP_i. The MN then performs AP discovery procedure and finds AP_i at a first channel, AP_l at a second channel, and AP_q and AP_t at a third channel.

2 and 3. The MN selects AP_i as target AP, performs 802.11 open system authentication with AP_i, and then associates with AP_i.

4. The MN performs 802.1X authentication with the AAA server.

5. The MN obtains topology information from the location server, which is co-situated at the AAA server. According to the topology information, the MN knows which APs are neighbors.

6. When **Cite Survey Criteria** are satisfied, e.g., RSS between the MN and current AP falls below a threshold, the MN takes leisure time to perform background site survey to obtain RSS between the MN and neighbor APs (may be all or subset of the MN's neighbor APs). Afterward, the MN generates candidate APs according to the MN's moving trend, RSSs between the MN and neighbor APs, and topology information. The performance of background site survey and the generation of candidate APs are repeated at a time interval as long as the **Cite Survey Criteria** are satisfied.

7. When **Pre-Authentication Criteria** are satisfied, e.g., RSS between the MN and current AP further falls below another threshold, the MN requests the AAA to perform pre-authentication with candidate APs. As shown in Figure 7-3(a), the MN is moving from the room serviced by AP_i to the room serviced by AP_l and therefore the candidate APs are AP_l and AP_q.

8. When **Handoff Criteria** are satisfied, the MN selects AP_l as target AP according to the MN's moving trend, RSS between the MN and neighbor APs, and topology information. The MN then associates directly with AP_l. If direct association succeeds, the AP_l can start serving the MN since the MN has pre-authenticated via the AAA server with AP_l. If direct association fails, the MN performs AP discovery procedure and standard handoff procedure as that defined in 802.11.

After successful direct association, Steps 6 to 7 are repeated for preparing next handoff. Note that in this protocol flow only messages in Steps 5 and 7 are our proposed addenda while other messages are defined in 801.11 or 802.1X standard [25]. That is, the proposed approach can leverage the existing or proposed related protocols.

7.2 Performance discussion

This section compares our approach with the most recent schemes [28,73] in terms of handoff delay and load on APs. For demonstration purposes, we address the scenario applying our approach in Robust Security Networks where handover to a not-yet-authenticated AP can cause the MN to undergo IEEE 802.1X [25] operations. Such early handoff occurs when the station reassociates with a new AP before the due pre-authentication thereto has been completed. The scenario provides our baseline performance.

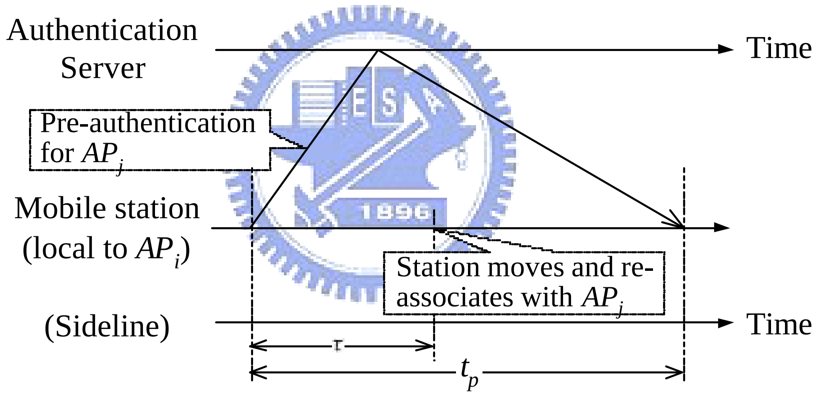

Authentication Server
Time
Pre-authentication for AP_j
Mobile station (local to AP_i)
Station moves and re-associates with AP_j
(Sideline)
Time
τ
t_p

Figure 7-4. A case of early handoff

As shown in Figure 7-4, consider an MN of AP_i switching earlier to AP_j than the intended pre-authentication therein has completed. Let variable τ denote the elapsed time from the point when pre-authentication is initiated to the point when the MN disassociates its current AP_i. Let t_p represent the pre-authentication duration. From network statistics, it is feasible to obtain the probability Pr[$\tau < t_p$] of early handoff. Given delays for reassociation, IEEE 802.1X, and four-way handshake average 2ms, 250ms, and 60ms, respectively [74], handoff in our architecture incurs a latency as follow:

$$L = 2\text{ms} + \Pr[t < t_p] \times 250\text{ms} + 60\text{ms} \qquad (1)$$

Regarding counterpart schemes, suppose that security contexts are available on APs wherever, i.e., these schemes never encounter early handoff (otherwise another term analogous to $\Pr[t < t_p]$ should be included hereinafter.) In that event, IEEE 802.1X operations are bypassed during handoff but AP discovery still demands 0 or 1000 ms, or a delay of 40 to 300 ms, depending on the scanning mode in actual use. If such a delay takes a mean $\bar{t}$, the average handoff latency amounts to

$$L' = \bar{t} + 2\text{ms} + 60\text{ms} \qquad (2)$$

From Equations (1) and (2), our approach outperforms counterpart schemes when $\Pr[t < t_p] \times 250\text{ms} < \bar{t}$. This condition very probably holds, since an IEEE 802.11 network, like a hotspot or office area, is likely to be well-planned and an MN's roaming within the network is in general not speedy. Hence sudden, early handoff in the pre-authentication setting would occur rarely, say, at probability 0.1. In the case of $\Pr[t < t_p]$ and $\bar{t}$ measuring 0.1 and (40+300)/2 ms, respectively, a handoff delay of 87 ms is required by our approach, as opposed to that of 232 ms by counterpart schemes. Handoff delay is curtailed to nearly one third, a marked improvement. Our performance gain follows similarly by examining other combinations of $\Pr[t < t_p]$ and $\bar{t}$. Furthermore, by setting $\Pr[t < t_p] = 1$, we can obtain the delay of the case where the new AP does not belong to S.

Our approach mitigates the load on APs. For each MN, the process involves a limited set S of candidate APs rather than a broader set N of potential next APs. The inequality $|S| \le |N|$ maintains $|S|$ strictly smaller than $|N|$ in most cases, unless θ is set to $180°$ (then $|S| = |N|$.) The difference $|N| - |S|$ per MN reflects strength of our approach. The load is reduced significantly when the system accommodates larger numbers of stations performing frequent handoffs. The expected size of S is susceptible to network topology layout, station' s location, and θ. In particular, θ is quantifiable in line with the mobility paradigm of each MN. If a station tends to make rectilinear migration, small θ suffices. A larger θ up to $180°$ caters for an MN that roams randomly. For example in Figure 7-5, the MN may turns to other directions after a move, as labeled with (a) and (b). In the former, S can be $\{ AP_q, AP_l \}$, while in the latter $\{ AP_l, AP_k \}$. This implies that $|S|$ may reasonably range from 1 to 3 in a cell-like hexagonal model.

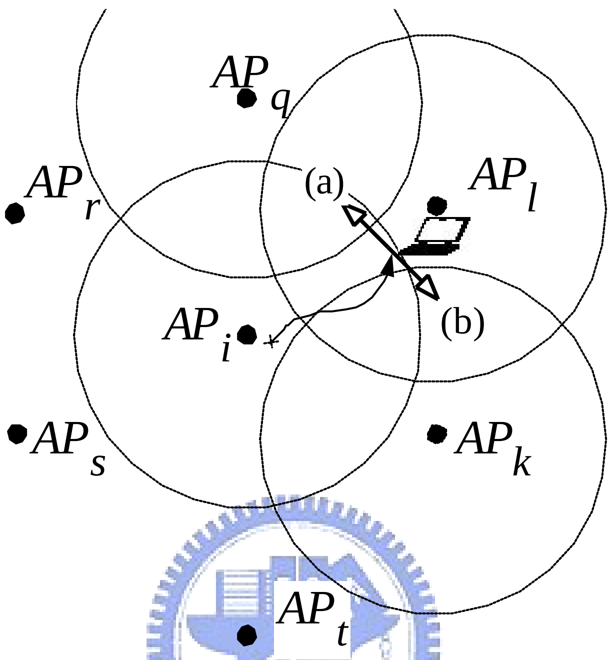


Figure 7-5. Determining S under random mobility

The overhead of deploying the introduced location server resembles that of a network server in Pack's schemes [27,51], which maintains a user's profile and computes frequent handoff regions exploiting a Markov model. Compared with Mishra *et al.*'s schemes [28,29,53], the location server requires additional environment information. The obtaining of topology information is not necessarily steps such as the Step 5 in Figure 7-3(b) during each handoff. Since topology information is mostly invariable, APs and MNs allow for retaining the received information for long-term use. Concerning the computational overhead of generating candidate APs, because the number of APs is normally small, we believe that the incurred cost is acceptable for today's machines if the criteria for activating positioning procedure, background site survey (Step 6 of Figure 7-3(b)), or pre-authentication (Step 7 of Figure 7-3(b)) is properly defined. Alternatively, the location server may derive the MN's locations and/or the neighboring APs on behalf of the MN if necessary.

7.3 Summary

Our proposal has three aspects: awareness of topology information, generating candidate APs along the moving trend, and direct reassociation. As stated in Section 7.1.4, the candidate-AP generation and positioning procedures can be shifted to the network side as well. Evaluation indicates that our approach considerably reduces handoff overhead. Besides, the proposed approach can leverage the existing or proposed mobility related protocols or systems, e.g., Robust Security Networks. Moreover, the topology-and-movement-aware candidate-AP generation scheme can work jointly with the existing fast-handoff schemes that leverage pre-authentication, proactive key distribution, or context transfer. We have implemented the direct association concept on an IEEE 802.11b adapter with a Realtek RTL8180L chipset.

Chapter 8 Conclusions and Future work

In this thesis, we propose a Dynamically Configurable NAT (DCNAT), a hybrid web cache replacement algorithm, a web session handoff mechanism, and a Topology-and-Direction-aware link layer fast handoff scheme to support continuous web access and real-time information pushing for mobile hosts that may visit private networks. These four approaches together provide convenient Internet access in mobile environments and NAT environments.

For the NAT inbound session problem, we propose Dynamically Configurable NAT (DCNAT) so that a host situates in Internet can spontaneously establish inbound session and push instant information to hosts beneath NAT networks. DCNAT enables NAT inbound session by a Binding Entry Request procedure to create NAT binding entries. The dynamic creation of NAT binding entries makes DCNAT very flexible in supporting inbound accesses to the ports/services opened dynamically by the private nodes behind an NAT router. Besides, with DCNAT, a content service provider can push information contents spontaneously to subscribers that are widely spreading and beneath different private networks with NAT. With the dynamic creation of NAT binding entries, DCNAT provides lower blocking probability and better scalability by using just one or several public IP addresses.

In order to improve web cache performance, we propose a hybrid web cache replacement algorithm that divides a cache space into two zones, a hot zone and a cold zone, and adopt a simple LRU replacement algorithm for the hot zone and a complex GD-Family replacement algorithm for the cold zone. This hybrid replacement algorithm combines the advantage of SLRU, LRU, and GD-Family algorithms but applies a complex GD-Family replacement algorithm only to a portion of the cache where strict object ordering is much significant to the overall cache performance. Therefore it can achieve a high DHR just slightly lower than that of GDSP and a high BHR just slightly lower than that of LRU while incurs much less maintenance cost, comparing with GD-Family algorithms.

To resume a web browsing session after a user changes the device that he uses to browse

Internet, we propose a web session handoff system that can hand over not only stateless but also stateful sessions between homogenous or heterogeneous user devices to enable uninterrupted and seamless web accesses. Compared with client-based approaches, our design has several advantages, such as less modification to user devices, practicability, and fault tolerance. We have implemented a UAP on a PC and client programs for both PC and PDA. The implementation can successfully hand over between PC and PDA a stateful session for online shopping applications.

To speed up the process of handoff between 802.11 APs, we propose a Topology-and-Direction-aware fast handoff scheme that resolves a reduced set of candidate APs according to AP topology, MN's its moving trend and/or MN's position. The proposed approach further benefits handoff by allowing 802.11 station reassociates directly with an AP without performing AP discovery during handoff. Besides, the topology-and-movement-aware candidate-AP generation scheme can work in conjunction with the existing fast-handoff schemes that leverage pre-authentication, proactive key distribution, or context transfer.

In the future, we will evaluate more hybrid algorithms that combine algorithms other than LRU and GD-Family. Furthermore, we will study the applicability of hybrid policy in other applications, such as web cache on user devices with limited cache space and memory paging in computer system.

There is still further improvement to the proposed web session handoff system. First, we will improve our client program to collect more assistant information so that the UAP can be more light-weighted and session information can be handed over completely. Besides, since session tracking generates additional processing and storage load to UAPs, we will perform a quantitative analysis on workload and storage space problem of UAPs. Furthermore, we will also apply and refer previous research in scalable hierarchical proxies to design a system that contains multiple cooperated UAPs.

The practicability of the proposed Topology-and-Direction-aware fast handoff approach depends on the accuracy of measurement of MN' moving trend and/or position, especially in indoor environments. Therefore, we will study and implement moving trend estimation mechanism to verify the practicability of the Topology-and-Direction-aware fast handoff approach.

Reference

[1] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)" *RFC 1631,* IETF Network Working Group, May 1994.

[2] H. Schulzrinne and E. Wedlund, "Application-layer mobility using SIP", *Proceedings of IEEE Conference on Service Portability and Virtual Customer Environments*, pp. 29-36, 2000.

[3] Rajive Bagrodia, Thomas Phan, and Richard Guy, "A Scalable Distributed Middleware Service Architecture to Support Mobile Internet Applications", *Wireless Networks, the Journal of Mobile Communication, Computation, and Information (WINET),* vol. 9, no. 4, pp. 311-320, 2003.

[4] P. Srisuresh and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", *RFC 2663,* IETF Working Group, Aug 1999.

[5] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", *RFC 3489,* IETF Working Group, May 2003.

[6] "The netfilter/iptables project". Available from <http://www.iptables.org/>.

[7] P. Mockapetris, "Domain Names – Concepts and Facilities", *RFC 1034,* IETF Working Group, Nov 1987.

[8] P. Mockapetris, "Domain Names – Implementation and Specification", *RFC 1035,* IETF Working Group, Nov 1987.

[9] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira, "Characterizing reference locality in the WWW", *Proceedings of 1996 International Conference on Parallel and Distributed Information Systems (PDIS'96),* Dec 1996.

[10] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker, "Web caching and Zipf-like distributions: Evidence and implications", *Proceedings of Infocom'99*, Apr 1999.

[11] Pei Cao and Sandy Irani, "Cost-Aware WWW Proxy Caching Algorithms", *Proceedings of 1997 USENIX Symposium on Internet Technology and Systems*, December 1997.

[12] Carlos Cunha, Azer Bestavros, and Mark Crovella, "Characteristics of WWW client-based traces", *Technical Report BUCS95-010*, Apr 1995.

[13] A. Bestravros and S. Jin, "Greedydual* Web Caching Algorithm, Exploiting the two Sources of Temporal Locality in Web Request Streams", *Proceedings of 5th International Web Caching and Content Delivery Workshop*, Lisbon, Portugal 22-24 May 2000.

[14] Shudong Jin and A. Bestavros, "Popularity-Aware GreedyDual-Size Web Proxy Caching Algorithms", *Proceedings of the 20th Intl. Conf. on Distributed Computing Systems*, IEEE, Apr 2000.

[15] R. Fielding et al., "Hypertext Transfer Protocol -- HTTP/1.1", *RFC2616*, IETF Working Group, 1999.

[16] K. Moore and N. Freed, "Use of HTTP State Management", *RFC2964*, IETF Working Group, 2000.

[17] Kristol D. and L. Montulli, "HTTP State Management Mechanism", *RFC2965*, IETF Working Group, 2000.

[18] Netscape Support Documentation, "Persistent Client State – HTTP Cookies". Available from <http://wp.netscape.com/newsref/std/cookie_spec.html)>.

[19] Jan Newmarch, "HTTP Session Management", *Electronic Commerce Technical Issues*. Available from <http://jan.netcomp.monash.edu.au/ecommerce/session.html)>.

[20] Henry Song, Hao-hua Chu, and Nayeem Islam, Shoji Kurakake, and Masaji Katagiri, "rowser State Repository Service", *Proceedings of International Conference on Pervasive Computing*, pp. 253-266, 2002.

[21] A. Mishra, M. H. Shin, and W. Arbaugh, "An empirical analysis of the IEEE 802.11 MAC layer handoff process," *ACM Comp. Commun. Rev*., vol. 2, no. 33, pp. 93-102, Apr 2003.

[22] I. S. 802.11, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, Nov 1999.

[23] C. Rigney, S. Willens, A. Rubens, and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", *RFC 2865*, IETF Working Group, Jun 2000.

[24] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko, “Diameter Base Protocol“, *RFC3588*, IETF Working Group, Sep 2003.

[25] I. S. *802.1X, Port Based Network Access Control Revision Draft 11*, Jul 2004.

[26] I. S. *802.11i/D8.0, Draft supplement to standard for telecommunications and information exchange between systems - LAN/MAN specific requirements - part 11: Wireless Medium Access Control (MAC) and Physical layer (PHY) specifications: Specification for enhanced security*, Feb 2004.

[27] S. Pack and Y. Choi, “Pre-authenticated fast handoff in a public wireless LAN based on IEEE 802.1X model”, *Proceedings of (IFIP) Personal Wireless Communications 2002*, pp. 175-182, Singapore, Oct 2002.

[28] A. Mishra, M. H. Shin, J. N. L. Petroni, T. C. Clancy, and W. Arbaugh, “Proactive key distribution using neighbor graphs,” *IEEE Wireless Commun*., vol. 11, no. 1, pp. 26-36, Feb 2004.

[29] M. H. Shin, A. Mishra, and W. Arbaugh, “Improving the latency of 802.11 handoffs using neighbor graphs,” *Proceedings of the 2nd Int’l Conf. Mobile Syst., Applications, and Service*s, pp. 70–83, 2004.

[30] P. Srisuresh, G. Tsirtsis, P. Akkiraju, and A. Heffman, “DNS extensions to Network Address Translator (DNS_ALG)”, *RFC 2694*, IETF Working Group, Sep 1999.

[31] E.S. Lee, H.S, Chae, B.S Park and M.R. Choi, ”An expanded NAT with Server Connection Ability,” *Proceedings of the IEEE Region 10 Conference, Volume: 2*, 1999.

[32] P. Kriens, “Method and System for Communication to A Host within A Private Network,” *U.S. Patent Application Publication 2001/0006523*, Jul 2001.

[33] UPnP Forum, *Internet Gateway Device (IGD) V 1.0,*
Available from <http://www.upnp.org/standardizeddcps/igd.asp>.

[34] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, *RFC 3489*, IETF Working Group, May 2003.

[35] D. Wessels and K. Claffy, "ICP and the Squid Web Cache", *IEEE Journal on. Selected Areas in Communication*, vol. 16, issue. 3, pp. 345-357, Apr 1998.

[36] Rassul Ayani, Yong Meng Teo, and Yean Seen Ng, "Cache Pollution in Web Proxy Servers", *Proceedings of Parallel and Distributed Processing Symposium 2003*, Apr 2003.

[37] N. Young, "Online caching as cache size varies", *Proceedings of 2nd Annual ACM-SLAM Symposium on Discrete Algorithms*, pp. 241-250, 1994.

[38] Martin Arlitt, Ludmila Cherkasova, John Dilley, Rich Friedrich, and Tai Jin, "Evaluating Content Management Techniques for Web Proxy Caches", *Proceedings of the 2nd Workshop on Internet Server Performance*, May 1999.

[39] Rassul Ayani, Yong Meng Teo, and Peng Chen, "Cost-based Proxy Caching", *Proceedings of the International Symposium on Distributed Computing and Applications to Business*, Engineering and Science, Wuxi, China, Dec 2002.

[40] Balachander Krishnamurthy and Craig E. Wills, "Proxy Cache Coherency and replacement – toward a more complete picture", *Proceedings of the 19th IEEE ICDCS'99*, Jun 1999.

[41] C. Perkins, "IP Mobility Support for IPv4", *RFC 3344*, IETF Working Group, Aug 2002.

[42] Jan, R. and et al, "Enhancing Survivability of Mobile Internet Access Using Mobile IP with Location Registers", *Proceedings of INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, pp. 3-11, 21-25 Mar, 1999.

[43] Di Stefano A., and Santoro C., "NetChaser: agent support for personal mobility", *IEEE Internet Computing*, vol.4, no. 2, pp. 74-79, 2000.

[44] Mema Roussopoulos and et al, "Person-level Routing in the Mobile People Architecture", *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pp. 165-176, 1999.

[45] Appenzeller et al., "The Mobile People Architecture", *Technical Report CSL-TR-99-777*, Stanford University, 1999.

[46] Ramiro Liscano et al., “Integrating Multi-Modal Messages across Heterogeneous Networks”, *Proceedings of ENM-97 In conjunction with the ICC-97*, pp. 45-53, 1997.

[10->47] Herman Chung-Hwa et al., “iMobile: A Proxy-based Platform For Mobile Services”, *Proceedings of the first workshop on Wireless mobile Internet*, pp. 3-10, 2001.

[48] Ari-Pekka Kanerva et al., *P920 Deliverable 1: VHE concept description, scenarios and protocols*, 2000.

[49] K. Raatikainen, “Middleware for Future Mobile Networks”, *Proceedings of IEEE International Conference on 3G Wireless and Beyond*, pp. 722-727, 2001.

[50] Tristan Richardson et al., “Virtual Network Computing”, *IEEE Internet Computing*, vol. 2, no. 1, pp. 33-38, 1998.

[51] S. Pack and Y. Choi, “Fast inter-AP handoff using predictive authentication scheme in a public wireless LAN”, *Proceedings of IEEE Networks 2002*, Atlanta, USA, Aug 2002.

[52] C. I. Bauer and J. Rees, “Predictive methods for handover”, *Proceedings of Commun. Syst., Networks and Digital Signal Process. Symp.*, Jul 2002.

[53] A. Mishra, M. H. Shin, and W. Arbaugh, “Context caching using neighbor graphs for fast handoffs in a wireless network”, *Proceedings of the 23rd IEEE Conf. Comp. Commun. (INFOCOM) 2004*, Mar 2004.

[54] Chang Hong-Chun, *A Dynamic-Configurable NAT Approach to Communicate with Hosts inside a Private Network*, Master Thesis, Dep. CSIE of National Chiao-Tung University, 2002.

[55] B. Aboba and M. Beadles, “The Network Access Identifier”, *RFC 2486*, IETF Working Group, Jan 1999.

[56] P. Falstrom, “E.164 number and DNS”, *RFC 2916*, IETF Working Group, Sep 2000.

[57] B. Crain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, “Internet Group Management Protocol, Version 3”, *RFC 3376*, IETF Working Group, Oct 2002.

[58] B. Fenner, “IANA Considerations for IPv4 Internet Group Management Protocol”, *RFC 3228*, IETF Working Group, Feb 2002.

[59] C. Perkins and D. B. Johnson, "Route Optimization in Mobile IP", *expired Internet Draft*, IETF Working Group, Sep 2001.

[60] R Droms, "Dynamic Host Configuration Protocol", *RFC2131*, IETF Working Group, Mar 1997.

[61] Sheldon M. Ross, *Stochastic Processes*, 2nd Edition, John Wiley & Sons Inc, 1996.

[62] National Laboratory for Applied Network Research, access logs at NLANR's proxy caches, Available from <ftp://ircache.nlnar.net/>.

[63] Lin Lih min, *Implementation of an Inter-Devices Mobility for WWW,* Master Thesis, Dep. CSIE of National Chiao-Tung University, 2001.

[64] Tsai Ching Sung, *Inter-Device handoff for WWW Service*, Master Thesis, Dep. CSIE of National Chiao-Tung University, 2002.

[65] Mikael Nilsson, Johan Hjelm, and Hidetaka Ohto, "Composite Capabilities/Preference Profiles: Requirements and Architecture", W3C Working Draft, 2000. Available from < http://www.w3.org/TR/2000/WD-CCPP-ra-20000721/)>.

[66] Franklin Reynolds, Chris Woodrow, Hidetaka Ohto, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies", W3C Working Draft, 2003. Available from < http://www.w3.org/TR/2003/WD-CCPP-struct-vocab-20030325/)>.

[67] Hidetaka Ohto, and Johan Hjelm, "CC/PP exchange protocol based on HTTP Extension Framework", W3C Note, 1999.
Available from < http://www.w3.org/TR/NOTE-CCPPexchange)>.

[68] S. Kent, and R. Atkinson, "Security Architecture for the Internet Protocol", *RFC2401*, IETF Working Group, 1998.

[69] A. Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 1996.

[70] T. Dierks and C. Allen, "The TLS Protocol Version 1.0", *RFC2246*, IETF Working Group, 1999.

[71] E. Rescoria, "HTTP Over TLS", *RFC2818*, IETF Working Group, May 2000.

[72] E. Rescorla and A. Schiffman, “The Secure HyperText Transfer Protocol”, *RFC2660*, IETF Working Group, 1999.

[73] *I. P802.11F, Recommended practice for multi-vendor Access Point interoperability via an Inter-Access Point Protocol across distribution systems supporting IEEE 802.11 operation*, Jul 2003.

[74] B. Aboba, “Fast handoff issues,” IEEE-03-155r0-I, IEEE 802.11 Working Group, Mar 2003.