

# PPP Migration: A Technique for Low-Latency Handoff in CDMA2000 Networks

Anand Kagalkar, Sarit Mukherjee, Sampath Rangarajan, Katherine Guo  
Data Networking Research Center  
Bell Laboratories, Holmdel, New Jersey

## Abstract

In current CDMA2000 standard, a Packet Data Serving Node (PDSN) acts as an IP gateway to the Internet. Mobile Nodes (MN) connect to a PDSN using a Point-to-Point (PPP) session and IP packets are tunneled over the PPP session from the client to the PDSN which then routes the packets onto a packet network. A CDMA2000 network is a hierarchical network where packets from an MN to the PDSN are transported over a Radio-Access Network (RAN). An MN could move from one RAN to another and still be anchored under the same PDSN; it is also possible that when an MN moves from one RAN to another, the anchor PDSN itself becomes different. In the latter case, there are two ways to handle mobility: (i) tear down the PPP session from the MN to the old PDSN and establish a new PPP session from the MN to the new PDSN, and (ii) use the fast-handoff mechanism as specified in the CDMA2000 standard where a P-P (PDSN to PDSN) tunnel is established to tunnel PPP frames from the old PDSN to the new PDSN and then to the MN. In this paper, we present a better approach to handling mobility than either of the above two techniques. The method is to migrate the PPP state from the old PDSN to the new PDSN transparent to the MN; once the PPP state migration is completed, the new PDSN will serve as the IP gateway to the MN. We have implemented the PPP migration technique and through experimental measurements show its benefits.

**Keywords:** CDMA2000 Network, Packet Data Serving Node (PDSN), Point-to-Point Protocol (PPP), Handoff, Voice over IP (VoIP).

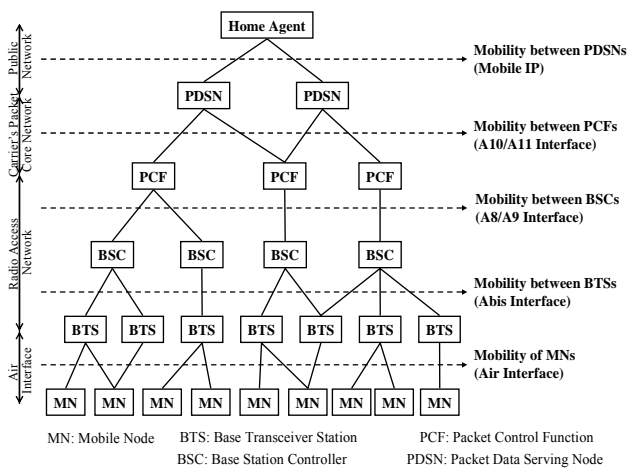
## 1. Introduction

The Code Division Multiple Access 2000 (CDMA2000) third generation wireless networks standards support packet data services along with the traditional voice services. Figure 1 shows the architecture of a CDMA2000 network that supports packet data services is hierarchically organized [1]. The Base Station Controller (BSC) communicates to Mobile Nodes (MN) via the Base Transceiver Stations (BTS). The BSC is connected to a Packet Control Function (PCF) that manages the relay of packets between the Radio Access Network (RAN) and the packet network through the Packet Data Serving Node (PDSN). The PDSN is the gateway to the IP network for a MN. It establishes, maintains and terminates

Point-to-Point (PPP) [18] link layer sessions to MNs, and is responsible for routing MN originated or MN terminated packets.

The creation of a packet data session between a MN and a PDSN proceeds with a MN initiating a data call. The BSC forwards the call to the PCF using the standard A8/A9 interface [2]. The PCF selects a PDSN based on the MN's unique device identity, known as the Mobile Station ID (MSID), and establishes a GRE tunnel with the PDSN using the standard A10/A11 interface [3]. After this all data frames sent by the MN get tunneled to the PDSN. The MN initiates a PPP session which is terminated by the PDSN which also performs IP routing for the MN. The PPP session enables data services for the MN. Therefore, as far as IP connectivity is concerned, the MN gets *directly* connected (i.e., single hop) to the PDSN.

In this architecture, MN's mobility needs to be addressed at different levels within the hierarchy. An MN could move between different BTSs controlled by the same BSC, between different BSCs controlled by the same PCF (in which case a A8 data session between the new BSC and the PCF needs to be established) or between different PCFs that connect to the same PDSN (in which case a A10 data session between the new PCF and the PDSN needs to be established). In all these cases, the PPP session between the MN and the PDSN remains intact and the movement is transparent to the link-layer (PPP) and the layers above (IP, TCP/UDP and applications). However, at the highest level of the hierarchy, it is possible that the MN moves between BTSs that are hierarchically controlled by different PCFs that may select different PDSNs to terminate the MN's PPP session. Mechanisms that handle mobility when a MN moves from the control of one PDSN to another, referred to as the inter-PDSN mobility, are the focus of this paper.



**Figure 1: Hierarchy and handoff requirement in CDMA2000 Network.**

To communicate with other hosts in the Internet, the MN needs to be assigned an IP address. One possibility is for the PDSN to assign an IP address to the MN when the PPP session is established; a sub-protocol within PPP called the IP Configuration Protocol (ICP) [15] facilitates this. The IP address obtained by the MN using this mechanism is normally referred to as a Simple-IP address. The drawback of this assignment technique is that if an MN moves from the control of one PDSN to another and reestablishes a PPP session with the new PDSN, it is assigned a new IP address specific to the network where the new PDSN resides. This closes all transport level (TCP, UDP) sessions that are in progress at the MN. Another option for the MN is to obtain a Mobile-IP address [12] and use it to communicate with Correspondent Hosts (CH) on the Internet irrespective of the current point of attachment. This requires Mobile-IP support from the CDMA2000 Network where a Foreign Agent (normally co-located with the PDSN) tunnels packets to/from a Home Agent (HA) residing on the MNs home network. The HA in turn routes these packets to the CH. This enables the MN to keep a fixed IP address (an IP address that belongs to the Home Network) even when it moves between PDSNs thereby keeping all transport level sessions alive during such movement. In this paper, we assume that IP addresses are assigned using Mobile-IP; this is the preferred approach and has been deployed in current commercial CDMA2000 networks to handle inter-PDSN mobility (as shown in Figure 1).

The inter-PDSN mobility can be handled in two different ways. One way is to tear down the PPP session between the MN and the source (i.e., old) PDSN and establishing a new PPP session between the MN and the target (i.e., new) PDSN. This changes the

anchor of the PPP end point for the MN. The MN then has to go through Mobile-IP re-registration to get IP packets routed to it through the target PDSN. Another way is to use the fast-handoff mechanism as specified in the CDMA2000 standard where a P-P (PDSN to PDSN) tunnel [4] is established to tunnel PPP frames from the source PDSN to the target PDSN and then to the MN. This keeps the anchor of the PPP at the source PDSN intact and, therefore, no PPP establishment or Mobile-IP re-registration is required. Establishing a new PPP session with a target PDSN is an acceptable solution if the MN is dormant (i.e., the MN is not actively sending and receiving data, in which case a traffic channel to the BTS will not exist). But if the MN has active sessions that carry real-time traffic such as VoIP, the delay involved in establishing a new PPP session (plus Mobile-IP re-registration) would adversely affect these sessions as we show using experimental results in a later section. The fast-handoff mechanism using a P-P tunnel is better suited in a situation where the MN is active. But this mechanism suffers from the additional requirement for backhaul bandwidth within the carrier's network to tunnel PPP frames from one PDSN to another. This requirement could be quite high as shown later in the paper.

We propose PPP migration as a technique to handle the inter-PDSN mobility. When an MN moves from the control of one PDSN to another, our solution is to migrate the PPP state from the source PDSN to the target PDSN transparent to the MN. That is, the MN need not explicitly re-establish the PPP session with the target PDSN. This solution is suitable for handing over both dormant and active MNs. We show using experiments that the delay incurred in migrating PPP state is very small and has very little impact on real-time sessions such as VoIP. In addition, the overhead due to backhaul bandwidth requirement seen with the P-P tunneling mechanism is avoided. Our solution is designed to co-exist with P-P tunneling in that a P-P tunnel could be setup initially to forward PPP frames from the source PDSN to the target PDSN and when PPP state has been migrated, the P-P tunnel could be closed.

Although we discuss PPP migration within the context of current hierarchical CDMA2000 networks, this solution makes even more sense in a flat architecture where Base-Station Routers (BSRs) provide all-IP connectivity to the MNs. In this architecture, the functionality of the BTS, the BSC/PCF and the PDSN are all co-located within a single network element (Base-Station Router) that provides a radio interface on one side and interface to the data network on the other

side. Currently, there is expanding interest in such a simplified architecture. One of the main problems within the context of BSR architecture is support for mobility. MN's movement between two BSRs has the same effect as inter-PDSN movement in a hierarchical architecture. If current solutions are used, inter-BSR movement will either require PPP sessions to be reestablished or P-P tunnels to be established. The difficulty with this is that a BSR is expected to have geographical coverage similar to a BTS and thus multiple such movements can be expected within the context of a single real-time session (for example, a single VoIP call). This exacerbates the problems that were discussed earlier with both the current solutions. As will be shown later in the paper, PPP migration takes on the order of a few milli-seconds and is an ideal solution for a BSR architecture.

Earlier, it was mentioned that we assume the use of Mobile-IP. This is not a strict requirement but will lead to a complete PPP state migration. With Simple-IP, the PDSN assigns IP addresses to the MNs. If the PPP state is migrated, the IP address assigned by the source PDSN will be unusable at the target PDSN. In this case, the sub-protocol within PPP that assigns IP addresses (i.e., IPCP) can be rerun at the target PDSN with the rest of the PPP state migrated. This requirement will be discussed in detail later in the paper.

## 2. Point-to-Point Protocol (PPP) State Migration

Within the CMDA2000 Network context, PPP provides a mechanism to establish a point-to-point link between the MN and the PDSN and then encapsulate and transport IP packets over this link. The PPP link establishment goes through two phases. In the first phase, called the Link Control Phase, the establishment of the point-to-point link is negotiated through a sub-protocol called the Link Control Protocol (LCP) [18] and then the user is authenticated using an authentication protocol such as CHAP (Challenge Handshake Authentication Protocol) [17] or PAP (Password Authentication Protocol) [10]. In the second phase, called the Network Control Phase, a sub-protocol called the Internet Protocol Control Protocol (IPCP) [15] is used to manage the specific needs of the IP packets that are transported over the PPP link. IPCP enables the PDSN to assign an IP address to the MN (in the case of Simple-IP) and also enables the negotiation of the header compression algorithm that are used to compress TCP/IP or RTP/UDP/IP headers. In addition, the Network Control Phase consists of

another sub-protocol called the CCP (Compression Control Protocol) [14] that is used to negotiate payload compression algorithm that may be used on packets transported over the PPP link. Once these two phases are complete, IP packets are encapsulated and transported over the PPP link. Thus, the four sub-protocols LCP, CHAP/PAP, IPCP and CCP, in that order, make up the different steps in the configuration of a PPP session. When a PPP session is migrated from the source PDSN to the target PDSN, parameter information negotiated as part of the sub-protocols as well as state information that may have been accumulated as part of packet processing (such as header compression state and payload compression state) have to be moved. We now consider the parameters for migration for each of the sub-protocols in more detail.

- **LCP:** LCP packets are exchanged between the MN and the PDSN to configure, test and later terminate the data link. Link specific parameters are exchanged using LCP configuration options. The parameters that need migration are Asynchronous Control Character Map (ACCM), Maximum Receive Unit (MRU), and the information about the sub-protocols to run next. The ACCM specifies a mapping of the control characters to be escaped out when PPP frames are transported over the PPP link and the resulting characters that need to replace the escaped characters. This maintains the PPP frame boundaries. The MRU defines the size of the PPP payload. The last parameter is useful when PPP is migrated without running all sub-protocols at the source PDSN, e.g., through LCP the end-points can negotiate that authentication has to be performed using CHAP. This will be discussed in more detail later.
- **CHAP/PAP:** These are authentication protocols, one or none of which is negotiated as part of the LCP configuration option. CHAP is three way handshake protocol where the authenticator (PDSN) sends a "challenge" to the MN which then computes a "response" based on a one-way hash function (which is the secret key) and then returns the response to the PDSN. PAP is a clear-text authentication protocol based on username and password. For PPP migration, username and the secret key (for CHAP) or the password (for PAP) should be moved. In practice, the secret key or the password is stored in an AAA server that is accessible to all the PDSNs. In such a case, the secret key or the password need not be moved.

- **IPCP:** This sub-protocol allows the PDSN to assign an IP address and DNS server IP address to the MN (in case of Simple-IP) and negotiate IP header compression algorithm to use on IP packets transported over the PPP link. The header compression algorithms are VJ (Van Jacobson) compression [7] for TCP/IP headers and ROHC (Robust Header Compression) [5] for RTP/UDP/IP headers. In case of Mobile-IP, the IP address and DNS server IP address will not be assigned by the PDSN and therefore, need not be moved. In case of Simple-IP, IPCP must be rerun at the target PDSN to assign a new IP address to the MN as the IP address assigned by the source PDSN will not be topologically consistent at the target PDSN. During this the new DNS server IP address is also assigned by the target PDSN. Therefore, when PPP migration is performed, only the IP header compression algorithm information is moved from the source to the target PDSN.
- **CCP:** The PPP Compression Control Protocol (CCP) is responsible for configuring, enabling, and disabling data compression algorithms on both ends of a PPP link. The compression algorithm is negotiated for each direction. The algorithms used in CDMA2000 standard are MPPC [11], LZS [6] and Deflate [19]. When PPP migration takes place, information about the negotiated compression algorithm for both directions and known at the source PDSN should be moved to the target PDSN.
- **Compression States:** As indicated earlier, header and payload compression states should be moved when PPP state is migrated. Compression states for both header and payload changes on a per packet basis and the most recently computed state should

be moved when PPP is migrated. If the compression states are not moved, the states need to be recreated from new packets processed at the target PDSN and this will lead to short-term inefficiencies. While the correctness of PPP state processing will not be affected if compression states are not moved, it is advantageous to move them. As we argue later in the paper, for real-time sensitive applications such as VoIP that benefit from PPP migration, it is more important to consider moving header compression state than payload compression. In fact, for VoIP, payload compression will not even be enabled given the small payload size of VoIP packets. Because of this, in our implementation, we have chosen not to account for payload compression and instead move only header compression state. Header compression algorithms work by requiring the sender to send only those header fields that have changed compared to the immediate previous packet that was sent. If the receiver has incrementally computed and stored the header for the previous packet, it can construct the new header using the previous header and the changed header fields. This means, header compression state will amount to a single packet header that is stored at the receiver.

Table 1 shows the information and the approximate number of bytes to be moved as part of PPP state migration from the source PDSN to the target PDSN when PPP is migrated. The number of bytes is applicable to a prototype we have developed on the Linux operating system.

Parameter	Information	Size for Linux (approx)	Frequency of Change
LCP	ACCM, MRU, Authentication protocol	70 Bytes	Once during setup
PAP	Username, password	100 Bytes	Once during setup
CHAP	Username, secret key	150 Bytes	Once during setup
IPCP	Header compression protocol	30 Bytes	Once during setup
CCP	Compression algorithm	10 Bytes	Once during setup
Header Compression	Header information	40 Bytes	With every data packet
	Total	300 Bytes	

**Table 1: PPP state information size (note only one of PAP or CHAP is used).**

### 3. PPP State Migration Mechanism

We assume that Mobile-IP is in use at the MN and PPP migration co-exists with P-P tunneling; that is, a P-P tunnel is setup between the source and the target on which PPP frames may be tunneled until the PPP state is migrated. Once PPP migration is completed, the P-P tunnel is released. This provides the flexibility to delay PPP migration until some time after the MN has moved from source PDSN to target PDSN. This way, PPP migration could be performed after the MN has stayed within the target PDSN control for some time to avoid ping-pong effects. In the meantime, the P-P tunnel will be used as specific support to tunnel PPP frames.

The packet flow to and from the MN is controlled by the Reference source not found.. Since the HA tunnels the IP packets for the target PDSN and the source then tunnels PPP frames to the MN. When the MN is under the control of the target PDSN, the target P-P tunnel with the source PDSN through which the target knows the source to setup the tunnel is beyond the scope of this paper). PPP frames are tunneled to the target PDSN. The PPP frames received at the target PDSN are tunneled to the MN using standard Mobile-IP tunneling. In the reverse direction PPP frames received at the target are tunneled over the P-P tunnel to the source.



1. PPP Migration Request →



Figure 2: PPP migration procedure.

The PPP migration between the source and the target PDSNs is a four-step process:

1. When the target decides to initiate PPP migration, it starts holding (i.e., buffering) PPP frames from the MN (i.e., it does not forward any more PPP frames received from the MN to the source) and sends a PPP Migration Request to the source. In our implementation, the request goes over a separate UDP connection.
2. When the source receives this request, and decides that migration is possible (we will discuss why the source might decide otherwise later) it starts buffering IP packets from the HA. The source then processes the IP packets to generate PPP frames and sends a PPP Migration Response to the target. The response contains the parameters and the state of the PPP session to migrate the PPP session. The source then migrates the PPP session up locally and sends a PPP Migration Confirm message back to the target. The target starts PPP processing by first releasing the PPP frames on hold and then processing new PPP frames arriving at the target. The output of this PPP processing is IP packets that will be tunneled to the target PDSN. When the migration is received at the source, it starts buffering IP packets on hold to the target PDSN. The source then tunnels new IP packets received at the target PDSN. The source also moves the PPP state information to the target PDSN and sends a PPP Migration Ack to the target. At this point, all PPP frames received at the target and the target PDSN are tunneled IP packets received at the HA respectively to one

Once the PPP session is migrated, the FA at the target PDSN takes over the Mobile-IP packet forwarding functionality. For this, it follows the standard Mobile-IP registration sequence using standard Mobile-IP messages, as described in steps 5 through 7. It sends Mobile-IP Agent Advertisement message to the MN, receives the Mobile-IP Registration Request from the MN and forwards it to the HA in the home network. The HA sends a Mobile-IP Registration Reply message which is forwarded to the MN. Once Mobile-IP registration is complete packets between the MN and the CH are tunneled between the new FA at the target PDSN and the HA. At this point, the P-P tunnel between the source and the target PDSN is released. Note that this could have been done right after Step 4, but keeping the P-P tunnel provides the flexibility to tunnel IP packets (not PPP frames) between the source

PDSN and the target PDSN after the PPP session is brought up at the target PDSN but before Mobile-IP Registration binding the MN to the new FA at the target PDSN is complete.

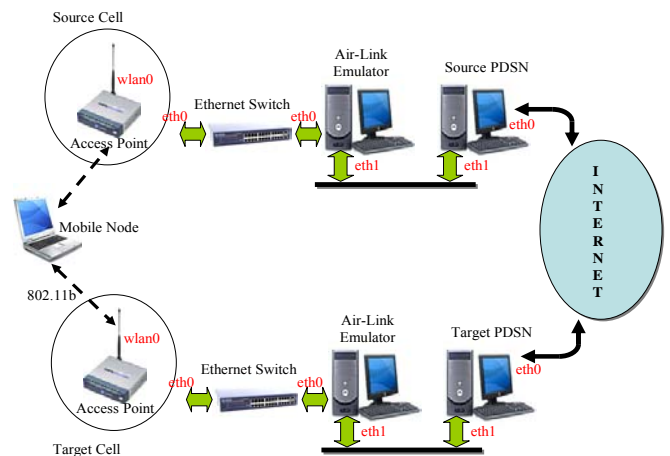
The new messages (shown in the call flow in steps 1 through 4) required to transfer the PPP state is described in more detail below:

1. **PPP Migration Request (MS-ID):** The Target PDSN sends a request to Source PDSN to migrate the PPP state. The state to be migrated is identified by MN's MS-ID (Mobile Station ID).
2. **PPP Migration Response (MS-ID, PPP-State-Info):** The Source PDSN sends the PPP state information for all the negotiated sub-protocols corresponding to the requested MS-ID. If the PPP state in Source PDSN is in the middle of negotiation of a sub-protocol, it will delay until the negotiation is completed. The PPP-State-Info contains the state parameters described earlier. In addition, information about the sub-protocols for which negotiations have been completed will be sent so that the target PDSN can complete the negotiations for the remaining sub-protocols. Although it is possible to migrate PPP state at sub-protocol boundaries, a recommended implementation would be to complete all sub-protocol negotiation at the source before migrating the state to the target. Note that if Simple-IP is used, this will be indicated as part of the state transfer so that IPCP can be rerun at the target.
3. **PPP Migration Confirm (MS-ID, Yes/No):** If the Target PDSN can support the PPP parameters negotiated for all the sub-protocols between the Source PDSN and the MN, the Target PDSN brings up the PPP session using the migrated state and confirms positively. Otherwise, the Target PDSN confirms negatively and the PPP state is not migrated. In this case, the PPP session will be re-negotiated between the MN and the Target PDSN.
4. **PPP Migration Ack (MS-ID):** If the Source PDSN receives a positive confirmation, it will acknowledge this and starts forwarding IP packets to the Target PDSN over the P-P tunnel that was previously established to tunnel PPP frames. The Target PDSN will start the process of taking over as the FA. Once this is completed, IP packets will be directly forwarded from the HA to the new FA, and the connection established to tunnel IP packets between the source and the target will be brought down.

## 4. Implementation

We built a prototype of the PPP migration mechanism on the Linux operating system to perform experimental measurements. PPP migration is implemented using a network that accurately emulates all the functionalities relevant to PPP link creation, maintenance and termination, as mandated by the CDMA2000 standard. In order to obtain this, the features needed from the network are (i) standard PPP setup mechanism, both control and bearer, between the MN and the PDSN, (ii) CDMA2000 air link and RAN characteristics for introducing appropriate packet loss and delay between the MN and the PDSN, (iii) mechanism for the MN to move between cells that are hierarchically controlled by different PDSNs, (iv) signaling below PPP layer to indicate the MN's movement from source to target area.

The network configuration used to obtain this is shown in Figure 3. Two cells are created using IEEE 802.11 access points (AP). Each AP is connected to a Linux-based home grown PDSN through an Air-Link Emulator that emulates CDMA2000 air link and RAN characteristics very accurately [9]. In order to create handoff of a MN from the source to the target area, we move a Linux-based IEEE 802.11 node from the Source to the Target Cell, which are controlled by the Source and Target PDSNs, respectively.



**Figure 3: Experimental setup for development and testing of PPP migration.**

In a CDMA2000 network any time a MN is turned on or it moves from one cell site to another, signaling messages are exchanged (below PPP layer) between the MN and the network (i.e., BTS, BSC, PCF and PDSN). In our network such signaling is emulated

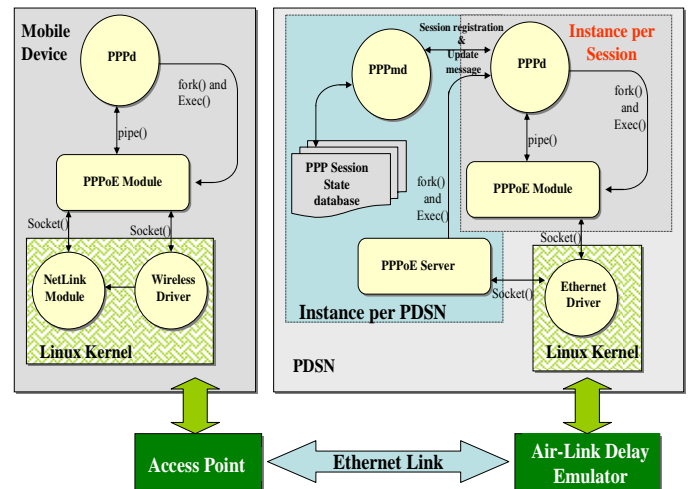
using PPPoE (PPP over Ethernet Encapsulation) **CITE PPPoE** protocol layer that runs below PPP. The PPPoE serves both as an encapsulation protocol to send PPP frames over Ethernet/802.11b link, and as a layer to keep all the signaling transparent to the PPP. We, therefore, run a PPPoE client at the MN and a PPPoE server at the PDSN, with the AP and Air-Link Emulator bridging the PPPoE Ethernet frames between the MN and the PDSN. Bridging is required as PPPoE client and server must be on the same link layer network. This configuration ensures that a direct standard compliant PPP link can be established between the MN and a PDSN. Note that the Air-Link Emulator only bridges the PPPoE control frames. The PPPoE frames containing PPP frames (i.e., PPPoE data frames) are passed through the Emulator so that appropriate radio link and RAN characteristics are emulated.

When a MN moves from a serving PDSN to a target PDSN, the CDMA2000 network provides the target PDSN information that can be used to determine and contact the serving PDSN (and for PPP migration, to obtain original PPP state information of the MN). Since the link layer (PPPoE in our case) keeps the MN mobility transparent to the PPP layer, this information is maintained and conveyed by the PPPoE layer. In our implementation, when a PPPoE link is established between the MN and the PDSN, the serving PDSN provides the information, which will be needed during migration, using PPPoE signaling messages to the PPPoE client on the MN. When the target PDSN accepts the migrating MN, the PPPoE client at the MN passes the information about the serving PDSN using PPPoE messages.

Figure 4 illustrates the prototype implementation of the PPP migration scheme. The figure shows the relation between various software modules on the MN and the PDSN. The following major modules are developed:

- PPPd:** It is the open source daemon application that implements the PPP state machine (i.e., PPP control) and handles IP data traffic. The same module is used at the MN and the PDSN with different configurations so that MN acts a client and the PDSN as the server. To support PPP migration, the application has been augmented with the capability to bring up the PPP stack without any explicit message exchange between the end points. When PPP migration is required, PPPd on the PDSN is invoked with several parameters received from the lower layer (i.e., PPPoE) as part of the PPP migration request from the MN. They include the original PPPd session id

and IP address and port number of the source PDSN that contains the already established PPP state. The PPPd at the target PDSN uses these parameters to contact the source PDSN to retrieve the PPPd state at each sub-protocol level (i.e., LCP, CCP, CHAP/PAP and IPCP). When PPP migration is not required, PPPd is brought up in the traditional fashion whereby the PDSN negotiates different sub-protocol parameters with the MN to establish a new PPP session.



**Figure 4: Modules developed for implementing PPP migration.**

- PPPmd:** The state of the already established PPP session is maintained at the terminating PDSN. This state information must be maintained for every incoming PPP session (typically each session represents one active MN). For PPP migration this state information needs to be moved from the serving to the target PDSN. Therefore, at each PDSN we keep a repository of the states of the PPP session that the PDSN currently handles. This repository is made accessible from any participating PDSN into whose hierarchy a MN may roam. Although there is no restriction on where the PPP session state repository should be kept, we recommend maintaining one repository instance per PDSN. This distributed scheme reduces the risk of single point of failure, and also potentially reduces the enormous load that could have been generated by movement of MNs using a centralized scheme. In our prototype this repository is served by a process running on the PDSN called PPPmd (PPP migration daemon). When PPPd completes a PPP negotiation with a

MN and brings up the PPP protocol stack, it sends a consolidated message, comprising of negotiated parameters for each sub-protocol, to local PPPmd in a UDP packet. The format of the message conforms to standard PPP messages [18]. In addition to these parameters complete information about the PPPd session is sent in each message. This information is used as a key for indexing into the session state repository maintained by PPPmd. The port number and IP address at which PPPmd listens are passed to PPPd from PPPoE server using command line arguments. We refer to the address-port pair as the migration-address and migration-port, respectively. PPPmd responds to PPPd, for every message received, with an acknowledgement confirming the proper processing of the update message. In reality if a PPP session consists of LCP, CCP, CHAP and IPCP then four separate messages are sent by PPPd to PPPmd and four acknowledgements are received by PPPd from PPPmd.

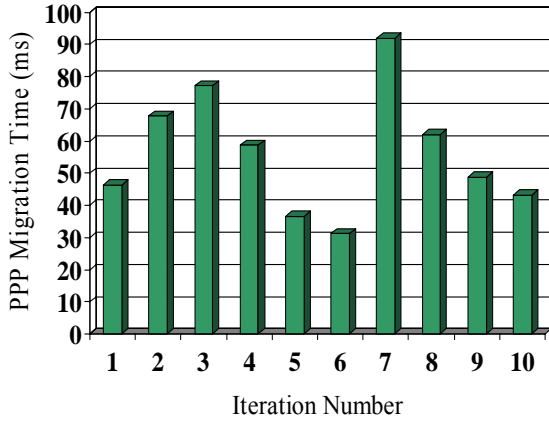
- PPPoE Module:** It is the link manager that encapsulates PPP frames over Ethernet and monitors the status of the physical link state for initiating handoffs. We use the open source PPPoE application and modify it to emulate necessary handoff related signaling so that the link management functionality can be kept hidden from PPPd. The PPPoE module at the client is responsible for detecting a PPPoE server and initiating link level session establishment with the server (PDSN), thus emulating a data call. The physical link state is known within the Linux kernel. If a physical link goes down, the PPPoE module waits until the link comes up and at that time searches for any available PPPoE server (which will be the target PDSN) and proceeds to establish a new PPPoE session. As part of this procedure, the PPPoE module sends previous session parameters to the new server (i.e., target PDSN). These parameters are then passed on to the PPPd at the target PDSN, which will migrate the old PPP session from the source PDSN without performing PPP negotiation with the PPP client at the MN. PPPoE does all the signaling required for PPP migration using vendor extension fields in the PADR message [16]. In order to monitor the link status, the PPPoE module uses the netlink-based socket interface of Linux. The link state change updates sent by the network device driver are sent to the netlink module running in the Linux kernel. The netlink module passes this information to the PPPoE module.

## 5. Benefits of PPP Migration

In order to illustrate the benefits of PPP migration, we compare our implementation to (i) tearing down the PPP session with the source PDSN and initiating a new PPP session with the target PDSN, and (ii) using a P-P tunnel for forwarding PPP frames between the source and the target PDSNs, keeping the PPP anchored at the source PDSN. While we evaluate the first benefit using our experimental setup, we use a simple mathematical model to quantify the second benefit. Results from these comparisons are discussed in the next two sections.

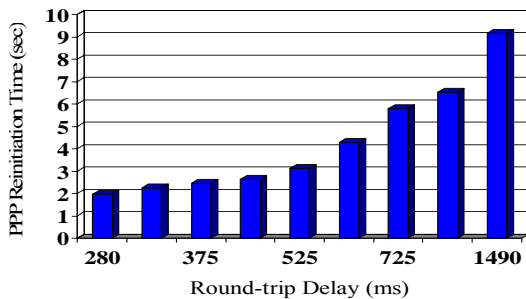
### 5.1 Initiation of new PPP session

We use the experimental setup of Figure 3 to compare the performance of PPP migration with PPP teardown and re-initiation. We compare the delay involved in setting up a new PPP session at the target with the delay in moving the PPP state from the source to the target. Note that once the PPP state is moved, PPP packet processing can begin immediately at the target. Therefore, this comparison illustrates the delay involved in starting the PPP packet processing at the target. The Air-Link Emulator (refer to Figure 3) is used to model different wireless link characteristics by controlling the error rates in fundamental and supplemental channels thereby introducing different amounts of delays across the wireless link. For each of these delays, we measure the time to re-initiate a PPP session at the target PDSN. For PPP migration, we measure the time to (a) move the PPP state where the complete state is moved including IPCP (applicable in case of Mobile-IP) and (b) move the complete PPP state except for IPCP which is re-negotiated at the target (applicable in the case of Simple IP). Note that for the former, the time to start PPP packet processing at the target is independent of the wireless link delay whereas for the latter, it is dependent on the wireless link delay as IPCP is re-negotiated at the target. In our experiments, we use the 1xRTT version of the Air-Link emulator. Figure 5 shows the time taken to migrate the complete PPP state, including the IPCP state, from the source to the target for ten different iterations. The average time for migration is around 56.34ms. The variance in the delay is due to the variance in the scheduling delay of the migration process at the process scheduler at the source and the target PDSNs.



**Figure 5: PPP migration time, the time taken to migrate the complete PPP state, including the IPCP state, from the source to the target PDSN).**

Figure 6 shows the time to reinitiate the PPP session at the target for different round-trip delays across the wireless link. For wireless link round-trip delays varying from 280ms to 1490ms, the PPP session reinitiation can take from around 2sec to around 9sec. Measurements performed on a commercial implementation of a 1xRTT network have shown that the mean round-trip delay varies from 400ms for an average loaded cell to 600ms for a heavily loaded cell. There is a large amount of variance in the round-trip time across the wireless link as well and hence the choice of round-trip delay from 280ms to 1490ms in our experiments.

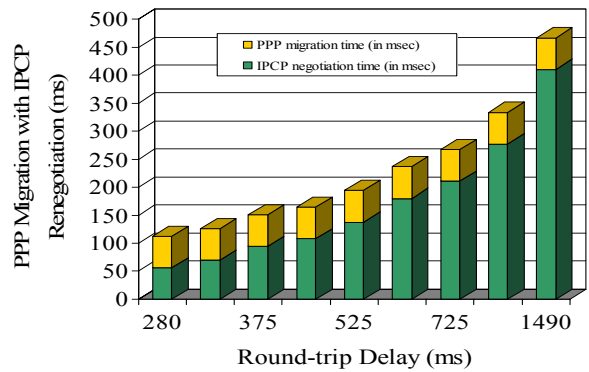


**Figure 6: PPP re-initiation time, the time to reinitiate the PPP session at the target PDSN.**

These results illustrate the advantage of using PPP migration as opposed to initiating new PPP sessions at the target. Compared to an average PPP migration time of around 56ms during which time data flow will be

interrupted, PPP re-initiation even with a small round-trip delay across the wireless link of 280ms takes around 2sec. Overall, there is a factor of 35 to 170 increase in the time during which data flow is interrupted when the MN moves from the source to the target. For VoIP, any handoff time larger than around 250ms leads to noticeable disruption in voice quality and hence PPP re-initiation is not even a viable option for moving active VoIP calls from one PDSN to another during MN mobility. As discussed earlier, with a BSR architecture where active VoIP calls can expect to move between multiple BSRs given the limited coverage area of a BSR, this becomes even more important.

To study the effect of PPP migration with IPCP negotiation at the target (applicable for Simple IP), we measured the time taken for IPCP negotiation only, at the target. The total time for PPP processing to start at the target then would be the sum of the PPP migration time and the IPCP negotiation at the target. These measurement results are shown in Figure 7.



**Figure 7: The total time for PPP processing to start divided into two parts (1) PPP migration time and (2) IPCP negotiation time at the target PDSN.**

For each round-trip delay value across the wireless link, the IPCP negotiation time and the average PPP migration time shown earlier (56.34ms) is shown. For the range of delays from 280ms to 1490ms, the total time is in the range of 110ms to 460ms, still much smaller than the PPP re-initiation time that varies from 2sec to 9sec.

## 5.2 Using a P-P tunnel between the source and the target PDSN

In order to illustrate the benefit of PPP migration over the mechanism of using a P-P tunnel between the source and target PDSN, we analyze the impact of P-P

tunnels established using the P-P tunnel mechanism. When MNs move from PDSN to PDSN, the PPP migration mechanism does not require any P-P tunnels for data traffic between PDSNs, however, the P-P tunnel mechanism would establish P-P tunnels between PDSNs over the backhaul network. The number of P-P tunnels established is a direct indicator of the extra amount of backhaul traffic introduced by the P-P tunnel mechanism.

Depending on the configuration of the PDSN hardware, a typical PDSN could have the capacity to support maximum of 500,000 active sessions in a CDMA2000 1xEV-DO system [13].

Consider a network where the total number of PDSNs is  $G$ , and the total number of active MNs in the network is  $H$ . We also assume all the active MNs stay active and no dormant MNs become active. Therefore the total number of PPP sessions remains constant. In addition, the average number of PPP sessions at originated at each PDSN is  $M = H/G$ .

The P-P tunnels and PPP sessions at each PDSN can be divided into two categories:

- I. this PDSN is the source PDSN for the P-P tunnel and the PPP session
- II. this PDSN is the target PDSN for the P-P tunnel and the PPP session

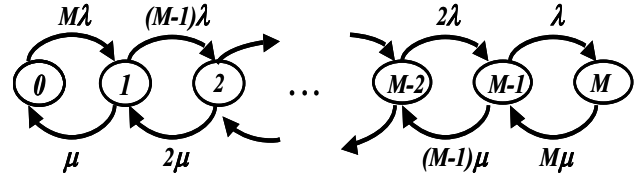
We are interested in the number of P-P tunnels in the system, which can be calculated as the number of P-P tunnels originated at each PDSN multiplied by the number of PDSNs in the system. Therefore, at each PDSN, we only need to study the behavior of  $K$ , the number of P-P tunnels using this PDSN as the source.

For a given MN with an active PPP session, there are only two possible cases of MN movements regarding the PPP session between the MN and its source PDSN.

- (a) The MN moves away to a neighboring PDSN where a P-P tunnel is established between the source PDSN and the target PDSN.
- (b) The MN moves back from a neighboring target PDSN to its source PDSN, where the P-P tunnel is eliminated.

For a given PDSN, let the rate that a MN moves away to a neighboring PDSN be  $\lambda$ , and let the rate that a MN moves back from a neighboring target PDSN to this source PDSN be  $\mu$ . The movement of a MN is independent of each other. In addition, the length of time each MN stays attached to a PDSN is independent

of the number of P-P tunnels established at the PDSN. The behavior of  $K$ , the number of P-P tunnels using this PDSN as the source can be modeled using a special class of Markov chains, the birth-death process with state-transition-rate diagram shown in Figure 8.



**Figure 8: State-transition-rate diagram for  $K$ , the number of P-P tunnels using this PDSN as the source.**

Consider the  $M$  PPP sessions originated at a given PDSN. When  $K = 0$ , all the  $M$  MNs are within the range of the source PDSN, since the rate that one specific MN moves away to a neighboring PDSN is  $\lambda$ , the rate that any of the  $M$  MN moves away to any of the neighboring PDSNs is  $M\lambda$  because each MN movement is independent. The state transition rate from state  $K = 0$  to state  $K = 1$  is therefore  $M\lambda$ , and from state  $K = 1$  to state  $K = 2$  is  $(M-1)\lambda$ , and so on.

Similarly, when  $K = M$ , all the  $M$  MNs are not within the range of the source PDSN. The rate that one specific MN moves back to the source PDSN is  $\mu$ . The rate that any of the  $M$  MNs moves back to the source PDSN is  $M\mu$  because of the independence of MN movements. The state transition rate from state  $K = M$  to state  $K = M-1$  is therefore  $M\mu$ , and from state  $K = M-1$  to state  $K = M-2$  is  $(M-1)\mu$ , and so on.

This is a classic infinite-server infinite-storage finite population M/M/ $\infty$ / $\infty$ /M birth-death queueing system. This queueing system can be modeled as [8]:

$$\begin{aligned} \lambda(k) &= \lambda(M-k); & \text{when } 0 \leq k \leq M \\ \lambda(k) &= 0; & \text{when } K > M \\ \mu(k) &= k\mu, & k = 1, 2, \dots \end{aligned}$$

where  $\lambda(k)$  and  $\mu(k)$  are the birth and death rate respectively when the population of the system is  $k$ .

The equilibrium distribution for the population of the system is therefore

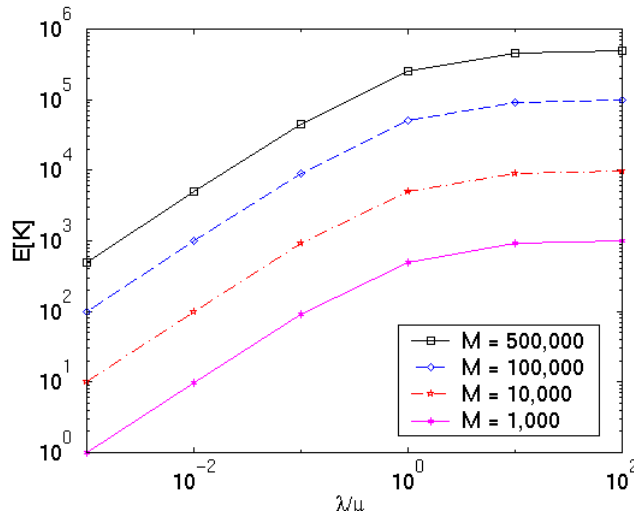
$$p_k = p_0 \rho^k M! / (k!(M-k)!), \quad 0 \leq k \leq M, \quad \text{where } \rho = \lambda/\mu$$

with  $p_0 = (1+\rho)^{-M}$

The expected value for the population in the system is therefore  $\sum E[K] = \sum_{k=0}^{k=M} (kp_k) = M\rho/(1+\rho)$

Note that this is the expected number of P-P tunnels where this PDSN is the source PDSN.

We plot the expected number of P-P tunnels sourced at each PDSN  $E[K]$  against  $\rho = \lambda/\mu$  in Figure 9.



**Figure 9: Expected number of P-P Connections sourced at each PDSN.**

Intuitively, when MN movement is random, on average, half of the PPP sessions initiated by this PDSN moves out of the region of the source PDSN, therefore require the use of P-P tunnels. The birth-death Markov chain analysis shows that when  $\lambda = \mu$ , and  $\rho = 1$ , that is, when the rate a MN moves in a region served by a PDSN is the same as the rate a MN moves out of the same region,  $E[K] = M/2$ . This result is consistent with the intuition.

We assume there are  $G$  PDSNs in the system, the total number of P-P tunnels in the system is therefore  $GE[K]$ . If the average bandwidth for each PPP session is  $B$ , the total amount of backhaul traffic generated by the P-P tunnel mechanism is therefore  $BGE[K]$ .

Our own measurements in a live CDMA2000-1xRTT network shows that an average cell has an average per user FTP bandwidth of 78Kbps. A congested cell has an average per user FTP bandwidth of around 56Kbps. Based on these measurements, we assume the average bandwidth for each PPP session is 70 Kbps. Consider a system with 3 PDSNs, with the average number of PPP sessions originated at each PDSN being 100,000, the total amount of backhaul traffic generated by the P-P

tunnel mechanism is therefore 2.1 Gbps. On the other hand, using the proposed PPP migration mechanism results in no backhaul traffic.

## 6. Conclusions

In this paper, we proposed PPP migration as a mechanism for handling mobility of a MN between PDSNs in a 3G CDMA2000 network. We have shown at the traditional approach of tearing down a PPP session at the source PDSN and re-initiating a new PPP session at the target PDSN incurs latency that is beyond what real-time traffic such as VoIP can tolerate. Using experimental results, we have shown that the PPP migration approach leads to latencies which are very minimal. In addition, using a mathematical model we have shown that the 3GPP2 proposed standard for fast-handoff using a P-P tunnel is not a desirable solution either because of the large overhead in additional backhaul traffic.

## 7. References

- [1] 3<sup>rd</sup> Generation Partnership Project 2, "Interoperability Specification (IOS) for CDMA 2000 Access Network Interfaces - Part 1 Overview," June 2004, A.S0011-B v1.0.
- [2] 3<sup>rd</sup> Generation Partnership Project 2, "Interoperability Specification (IOS) for cdma2000 Access Network Interfaces - Part 6 (A8 and A9 Interfaces)," June 2004, A.S0016-B v1.0.
- [3] 3<sup>rd</sup> Generation Partnership Project 2, "Interoperability Specification (IOS) for cdma2000 Access Network Interfaces - Part 7 (A10 and A11 Interfaces)," June 2004, A.S0017-B v1.0.
- [4] 3<sup>rd</sup> Generation Partnership Project 2, "cdma2000 Wireless IP Network Standard," October 2004, P.S0001-B v2.0.
- [5] C. Bormann, Editor, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," Internet Engineering Task Force (IETF) Request for Comments (RFC) 3095, July 2001.
- [6] R. Friend and W. Simpson, "PPP Stac LZS Compression Protocol," Internet Engineering Task Force (IETF) Request for Comments (RFC) 1974, August 1996.
- [7] V. Jacobson, "Compressing TCP/IP headers for low-speed serial links," Internet Engineering Task

- Force (IETF) Request for Comments (RFC) 1144, February 1990.
- [8] L. Kleinrock, "Queueing Systems Volume 1: Theory", Wiley-Interscience, 1975.
  - [9] G. Li, M. Lu, M. Meyers and P. Feder, "Wireless Data Service Emulator (WiDSE): A Real-time Look and Feel Emulator for Wireless Packet Data Systems," Technical Memorandum, Lucent Technologies, March 2002.
  - [10] B. Lloyd and W. Simpson, "PPP Authentication Protocols," Internet Engineering Task Force (IETF) Request for Comments (RFC) 1334, October 1992.
  - [11] G. Pall, "Microsoft Point-To-Point Compression (MPPC) Protocol," Internet Engineering Task Force (IETF) Request for Comments (RFC) 2118, March 1997.
  - [12] C. Perkins et. al., "IP Mobility Support for IPv4," Internet Engineering Task Force (IETF) Request for Comments (RFC) 3220, August 2002.
  - [13] Private communication with a PDSN vendor.
  - [14] D. Rand, "The PPP Compression Control Protocol (CCP)," Internet Engineering Task Force (IETF) Request for Comments (RFC) 1962, June 1996.
  - [15] G. McGregor, "The PPP Internet Protocol Control Protocol (IPCP)," Internet Engineering Task Force (IETF) Request for Comments (RFC) 1332, May 1992.
  - [16] L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)," Internet Engineering Task Force (IETF) Request for Comments (RFC) 2516, February 1999.
  - [17] W. Simpson et. al., "PPP Challenge Handshake Authentication Protocol (CHAP)," Internet Engineering Task Force (IETF) Request for Comments (RFC) 1994, August 1996.
  - [18] W. Simpson et. al., "The Point to Point Protocol (PPP)," Internet Engineering Task Force (IETF) Request for Comments (RFC) 1661, July 1994.
  - [19] J. Woods, "PPP Deflate Protocol," Internet Engineering Task Force (IETF) Request for Comments (RFC) 1979, August 1996.