Appeared in IEEE Personal
Communications Magazine
Vol. 1(3), 1994, pp. 36-46.

An Architecture and Communication Protocol for Picocellular Networks

Rohit Ghai and Suresh Singh

Dept. of Computer Science
University of South Carolina
Columbia, SC 29208
email: singh@cs.scarolina.edu
Tel: (803)-777-2596
Fax: (803)-777-3767

Abstract

In this paper we present an architecture for a *picocellular network* and a novel communication protocol for mobile users in such networks. The cell size in these networks is very small – typically of the order of a room in a building, resulting in an environment where the movement between cells is very rapid. This makes the problems of tracking and maintaining communication between mobile users non-trivial.

To better track these mobile users we propose an architecture consisting of a two level hierarchy of *fixed* nodes. This architecture combined with the idea of *groups* ensures that seamless communication is maintained between mobile users. Groups are special collections of picocells and represent regions where a mobile user is likely to be located in the near future. Thus, messages addressed to this mobile user are multicast to all the picocells

in the group guaranteeing delivery and obviating the need for messages to be forwarded between picocells. The mobile user will therefore see few breaks in communication as it moves between cells.

1 Introduction

Developments in cellular communication and wireless LAN technology will make it possible, in the near future, for mobile users to transparently access communication networks from anywhere at any time. The mobile user will typically use powerful palmtop or laptop computers that are equipped with wireless communication capability to connect to wired networks. This type of computing environment is referred to as *mobile computing*. The mobility of the nodes and the small size (in terms of storage space) of the hand-held devices make traditional solutions to network problems such as addressing, routing and flow-control inapplicable.

Recently several authors (see for instance [1, 7, 9]) have developed solutions to some of these problems for *microcellular* or *intra-campus* networks. A microcellular network is one where a geographical area (such as a university campus) is divided into regions called 'microcells'. Each microcell has a diameter of the order of several hundered meters and contains a *Mobile Support Station* (MSS) that provides a connection point for mobile users in that microcell. All the MSSs are connected via a wired network and are responsible for *tracking* the mobile users as they move between cells and *forwarding* packets for all currently open conections to the new MSS.

In this paper we consider an *in-building* or *picocellular* network and propose an architecture and a communication protocol to set up and maintain connections between pairs of mobile users. A picocellular network is similar to a microcellular network with the difference that the cell size is now of the order of 10 meters and is typically a room in a building as in Figure 1(a) (see [4, 9]). The advantage of a smaller cell size is higher throughput (because there will be fewer users per cell), greater frequency reuse (i.e., while adjacent cells use different radio frequencies, non-adjacent cells may use the same frequency) and low-power transmitters (resulting in longer battery life). The main drawback is that small cells imply *faster* inter-cell node mobility resulting in a host of routing, tracking and hand-off problems.

In a picocellular environment, the *cell latency* of mobile users (i.e., time a MH spends in a cell) is of the order of 4–5 seconds (assuming a speed of 2 to 3 meters/sec and a cell diameter of 10 meters), and the number of mobile users is very large (e.g., in an office building). If we use the traditional approach used in microcellular networks the amount

of tracking information that will need to be maintained by each MSS and the number of packets that need to be forwarded between the MSSs will be huge. Furthermore, the MH will see a *break in service* every time it moves into a new cell (i.e., every 5 seconds) while waiting for packets to be forwarded.

We propose a three-level architecture as shown in Figure 1(b). The SH (Server Host) has the responsibility of tracking the mobile hosts and maintaining their connections. In our approach, packets for an MH are *multicast* to MSSs of neighboring cells so that when the MH moves to a new cell, there are packets already waiting for it and there is no break in service. Another advantage of our approach is that the MSS nodes can be simple inexpensive devices. This is an important consideration because the number of MSS nodes in a building environment will be huge. An obvious drawback is the large amount of buffer space required at each MSS (because of duplication of data). As we will see in section 6, however, this buffer overhead can be reduced. In section 7 we discuss how our group-based protocol can be adapted to provide different grades of service for different types of applications running at the MH.

The rest of the paper is organized as follows. In the next section we present a brief literature review. In section 2 we present our architecture. An overview of the communication protocol is provided in section 3. We also present a simple model for computing the retransmission overhead for the traditional microcellular approach. In section 4 we present the protocol in detail. Section 5 discusses a strategy for flow-control and buffer management. Section 6 discusses techniques to reduce the buffer overhead of our group-based protocol. A summary of results is presented in section 7 and our conclusions are presented in Section 8.

1.1 Literature review

Problems associated with microcellular networks have been quite extensively studied in the literature. [7] present a set of IP-based protocols to provide continuous network access to mobile hosts in a microcellular environment. [6] proposes a very flexible solution to the problem of addressing mobile users – the Virtual Internet Protocol or **VIP**. [1] discusses the problem of multicasting in similar environments. [4] introduced the idea of a cellular

packet switch to efficiently handle the problem of hand-off in cellular communications.

In all of these papers the network architecture used is a two-level hierarchy consisting of the mobile users at the lower level and the mobile support stations at the second level who handle routing and other protocol details. This architecture requires each MSS to maintain a complicated set of forward pointers along which packets have to be forwarded to the current location of the MH. As discussed earlier this approach sufers from the twin drawbacks of overhead in terms of packets forwarded and frequent breaks in service to the MH as it crosss cell boundaries.

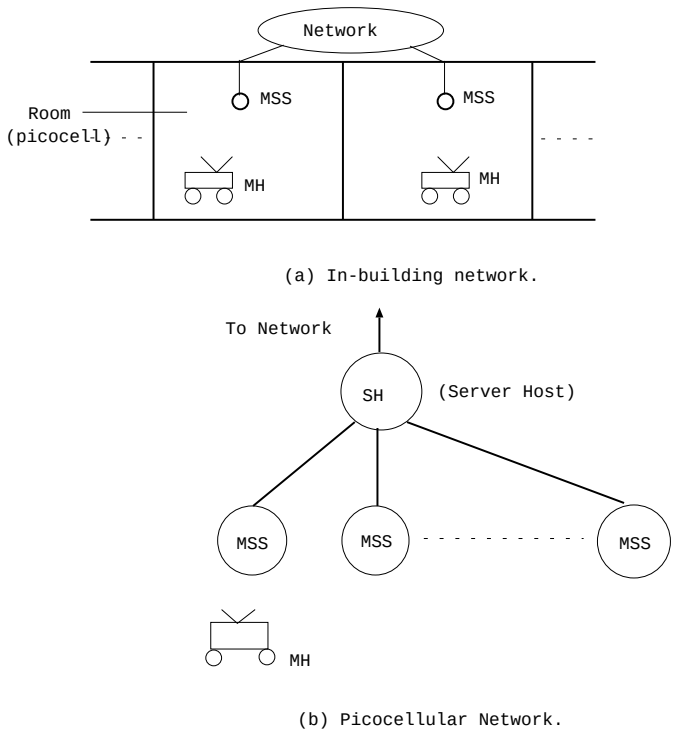


Figure 1: Typical architecture of a picocellular network.

2 Architecture for a Picocellular Network

The target environment of the proposed protocol is a typical office building. The building is assumed to have an already existing wired backbone network overlaid on top of which is the "picocellular wireless network". A cell in this network is typically restricted to a room

and corridors may be made up of several cells. The cell shape may not be uniform and is constrained by the architecture of the building as is movement between cells of mobile users, see Figure 1.

The system can be viewed as a three level hierarchy. At the lowest level is a dense collection of Mobile Hosts (MHs) which communicate with each other and with stationary hosts in the network as they move around between cells. Communication in each cell is handled by Mobile Support Stations (MSSs). The MSSs are all connected to an assigned supervisor machine called the Supervisor Host (SH). If the subnet spans a large physical area we may choose to have more than one SH per subnet e.g., if a subnet straddles many floors then we may have one SH per floor.

The MHs are small battery powered handheld devices equipped with a wireless transciever. The MSSs are also small in size and are typically placed on the ceiling of each room. Each MSS contains a large amount of buffer space since the MHs contain very little. The SH is a server machine connected to the wired network and it handles most of the routing and other protocol details.

We assume that every MH has a globally unique address (such as a Virtual IP address as in [6]) regardless of where it is in the network. This is necessary because, as pointed out in [7], if the address of an MH depends upon its location then higher-level software will have to deal with address changes which is very inefficient. The SH maintains the current location of every MH within its domain (i.e., its subnet). In addition, the SHs and MSSs together track the MHs as they move between cells. The MSSs are mainly responsible for flow-control and maintaining communication with all the MHs within its cell. In addition, a MSS buffers messages to and from all MHs.

3 Overview of the protocol

The basic concept used in our protocol to maintain a connection with a rapidly moving MH is the idea of a *group*. Let us assume that the MH is in cell c_i at some given time. A collection of cells sorrounding c_i that the MH can move into is defined as the current group of the MH and is denoted by g_i. Any message destined for the MH is multicast to all the MSSs of the cells in g_i. This ensures that even if an MH moves rapidly between cells,

messages are not lost and no pointer structures need to be maintained at the MSSs.

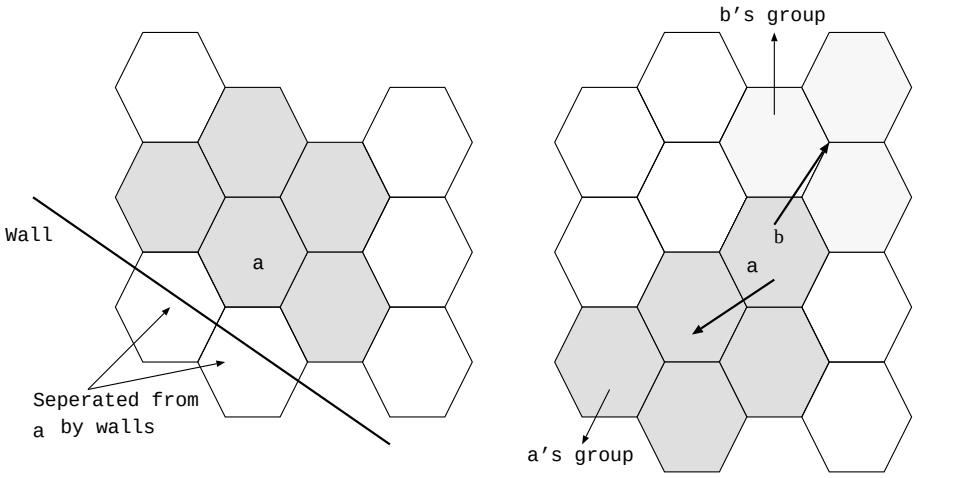
The shape and composition of a group is determined by the architecture of the building, the speed at which the MH moves between cells and the direction of motion. Cells that are seperated by a wall will almost never belong to the same group, likewise cells on different floors of the same building will not belong to the same group, see Figure 2(a). The speed at which an MH moves and the direction of motion also has a bearing upon the cells that are included in a group.

In Figure 2(b) the arrow represents the direction of motion of MH a and MH b. The shaded cells indicate the composition of the current group of a and b. It is important to observe that two MHs in the same cell, but moving in different directions, will have different groups. It is noteworthy that the direction of motion of an MH can be determined by keeping track of its old MSS, new MSS and the architectural constraints. The speed can be determined (with a high probability) by keeping track of cell sizes and cell latency times of MH nodes. In an in-building environment the motion of MH nodes is "purposeful" and therefore these parameters can be estimated with a high degree of reliability. For example, a person walking along a corridor has a well defined speed and direction of motion. Similarly, a person in an elevator has a well defined direction and speed of motion.

The SH keeps track of the current group of every active MH within its domain. When a message needs to be deliverd to an MH, the SH multicasts it to the current group of the MH. The group is updated each time the MH moves between cells and all the MSSs that belong to the new group are informed by the SH of the group identity number and the identity of the corresponding MH.

An MSS buffers all messages for an MH that it receives from the SH. If the MH is local, the MSS transmits the messages to the MH. If the MH is not local, the MSS holds on to the messages until either the MH enters its cell or until the SH informs it that it is no longer in the group. In the latter case, all messages are discarded. In the former case, during the initial 'handshake' between the MH and MSS , the MH informs the MSS of the *sequence number* of the next message it expects. The MSS transmits all messages starting with that sequence number. This method ensures that duplicate messages are not received at the MH. If the buffers of the MSS are full, the SH holds on to the packets until such time buffer

space becomes available at the current MSS.



(a) Building architecture affects group. (b) Affect of direction of motion.

Figure 2: Shape of groups.

3.1 Why Groups?

Traditional microcellular networks concentrate all network intelligence in the MSSs. Thus, as mobile hosts move between cells, the previous mobile support station is responsible for forwarding data for all its open connections to the current MSS. In this section we examine the *cost* of performing this operation and thus provide a justification for using groups.

3.1.1 Cost of Traditional Approaches

Let us focus on a single MH as it moves between two cells. When the MH moves into a new cell, it needs to inform the new MSS of its arrival and re-establish its open connections. Every MSS periodically (every T sec) transmits a *beacon* inviting new mobile hosts to announce themselves. After the MH announces itself, the new MSS informs the old MSS of the location of the MH. Thus all buffered packets can be *forwarded* to the new MSS.

We can divide these forwarded packets into two types – those present in the buffer of the old MSS before the MH moved away (N_1) and those that arrived after the MH moved out but could not be forwarded because the whereabouts of the MH were not known (N_2).

In addition there is a third cost that is noteworthy (for simplicity of analysis, however, we ignore it here). Let us assume that an MH originally set up connections when it was in the cell of MSS_1. It now moves out of this cell and into MSS_2,MSS_3,MSS_4 etc. In most discussions in the literature, [1], MSS_1 continues to serve as the connection end-point for all open connections and continues to forward packets to the current MSS of the MH. Only after the MH moves "far away" does the connection end-point get changed to the current MSS. The reason for doing this is that it would be too complicated and expensive for the connections to be constantly changed as the MH moves about (i.e., new route calculations to the new MSS, setting up new virtual circuits, etc.). The drawback of this approach is that every packet to/from an MH gets transmitted to MSS_1 and thence to its destination.

Ignoring the last cost discussed above, the total number of packets that need to be forwarded when an MH moves between two cells is,

$$N = N_1 + N_2$$

Let us assume that inter-packet arrival times (for a MH) are exponentially distributed with a mean of λ packets/sec and that the packet transmission times (from the MSS to the MH) are exponentially distributed with mean μ packets/sec. We can therefore model the packet behavior for a MH as a M/M/1 queue and obtain,

$$N_1 = \frac{\rho}{1-\rho} \text{ and } N_2 = \lambda T$$

where $\rho = \lambda/\mu$.

In a mobile environment, many mobile users typically cross cell boundaries per second. Packets need to be forwarded for each of these mobile users. Let M represent the number of mobile users that cross a cell boundary per second. Finally, let R denote the number of cells in the building. Then the total number of packets that need to be forwarded *per second* in the picocellular network may be written as,

$$N \times M \times R$$

Example: Let the number of rooms in the building be $R = 200$, $M = 1 - 2$ mobile users cross a cell boundary per second and the load per MH is, $\rho = 0.9$. Let the bandwidth

available to each user be 2Mbps (see [4]), assume the average packet length is 1KByte and $T = 100$ msec (in the literature a value of 1 sec is used which, we believe, is too long). Thus we obtain $N_2 = 22$ packets and $N_1 = 10$ packets. The total number of packets being forwarded per second over the network is therefore between 6000 and 12,000. This means the bandwidth used to forward packets is between 48Mbps and 96Mbps!

In the above discussion we have not considered the length of time for which the MH receives no service after it has announced itself to the MSS (since the MSS is waiting for packets to be forwarded). This time will depend upon the underlying network that connects the MSS nodes together.

What is the cost of our group based solution? The answer depends largely upon the underlying network architecture used to support the picocellular environment. If the MSS nodes are connected via a local area network supporting cheap multicast then the group based solution has no additional communication overhead since no packets are forwarded and, of course, no break in service to the MH. If, on the other hand, the cost of multicasting is equal to multiple single transmissions, then our group based solution will incur a large overhead because the same packet gets transmitted to all group members.

It is important to point out that our group-based solution as presented is expensive in terms of buffer overhead since packets for a MH are stored at several MSS nodes (i.e., all the group elements). In section 6 we present a *delayed buffering* policy that reduces this cost significantly.

4 Protocol description

The protocol we describe is a connection-oriented protocol. Whenever an MH wants to communicate with another MH, it requests a connection. The SH establishes a *virtual circuit* between the two MHs and returns a virtual circuit number to the MH. Every subsequent packet will contain this virtual circuit number.

The protocol views the system as a three level hierarchy with the three tiers of the hierarchy being (from bottom upwards) Mobile hosts (MHs) , Mobile Support Stations (MSSs) and Supervisor Hosts (SHs). The protocol requires that the following data structures be

maintained at these three levels:

- **MH :** The MH maintains the following information. It keeps the identity of the previous MSS, the current MSS and the identity of the current SH. In addition, it maintains a list of its current connections and the attributes of each connection. Each connection has a *virtual circuit* number, sequence number of last packet received and a status flag. The status can be one of:

  - ACTIVE_REMOTE : this means that the connection is active and the other end of the connection is in a different subnet (i.e., in the domain of another SH)
  - ACTIVE_LOCAL : this means that the connection is active and the other end of the connection is in the same subnet
  - NASCENT : this means that the connection is currently in the establishment phase

  If the status of a connection is ACTIVE_LOCAL, the group id of the MH at the other end of the circuit will also be recorded. This is because if the two communicating MHs are in the same subnet then the communication will not be through the SH. The MSS will directly multicast the message to the group of the destination MH.

- **MSS :** Each MSS maintains two tables. The first contains "static" information and lists the ID of all the groups it can belong to and the multicast address of each group, see Table 1. Consider a simple example shown in Figure 3. Some of the groups MSS_i belongs to are: $g_1 = \{i, 1, 2, 3, 4\}$, $g_2 = \{i, 3, 4, 10, 11\}$, $g_3 = \{i, 2, 3, 4, 8, 9, 10\}$, $g_4 = \{i, 1, 2, 3, 6, 7, 8\}$, $g_5 = \{i, 1, 2, 5, 6\}$, $g_6 = \{2, 1, 3, i\}$, $g_7 = \{5, 1, 6, i, 2\}$ etc. The group g_2 is the current group of an MH in MSS i when the direction of motion of the MH is from i to 4. When the direction of motion is from i to 3, we get group g_3. Group g_7 is the current group of an MH in MSS_5 when it has a direction of motion from 5 to 1. All these groups are listed in the table maintained at MSS_i along with their multicast addresses.

  The second table, Table 2, contains the ID of all MHs in cells that belong to one of the groups that the MSS lies in. Thus, if a MH lies in cell i with direction of

Group ID	IP multicast address
g_i	MA_i

Table 1: Group information at each MSS.

MH ID	Group ID
MH_i	g_i

Table 2: MH information at each MSS.

Loc of MH	Direction	Group	IP Multicast address
c_i	$\vec{v}$	g_i	MA_i

Table 3: MH Information avavilable at each SH

Group ID	Members	IP multicast address
g_i	$m_{i1}, m_{i2}, m_{i3}, \ldots$	MA_i

Table 4: Static information at SH.

motion towards cell 4 then the multicast group for this MH is g_2 (see Figure 3). Thus, MSS_i, MSS_3, MSS_4 and MSS_{10} will all contain the MH ID in their respective tables. Observe, however, only MSS_i can actually transmit to the MH since it is currently in cell i. All other MSSs in g_2 merely buffer messages for the MH.

When an MSS is initially powered up, it is provided with all the information in Table 1. It announces its birth to the SH who incorporates the MSS into its own tables.

- **SH :** At each SH there is a table that contains the information shown in Table 3. The first column is the position of an MH (in a cell c_i), the second column is its direction of motion. These map uniquely into a group ID and a corresponding multicast address.

  The SH also maintains a table of all the virtual circuits that are in operation in its subnet. The table contains the following information for each entry:

  - vc_{local} :the local virtual circuit number
  - vc_{remote} : the remote virtual circuit number of the MH at the other end of the connection (i.e., the other MH is in another subnet). Note that while the virtual circuit number is only locally (i.e., within each subnet) unique, the pair $< SH, vc\# >$ is globally unique.
  - status :the status of the connection which can be one of ACTIVE_REMOTE, ACTIVE_LOCAL, NACENT or POINT.
    - POINT : this means that the circuit has been adopted by another SH because the MH moved out of the local subnet into the subnet of another SH (SH_{new}) and this SH (SH_{local}) is currently simply forwarding the packets for virtual circuit vc_{local}.
  - vc_{point} : the virtual circuit number when the local MH (whose connection status was ACTIVE_REMOTE) moves away into another subnet.
  - SH_{point} :if status is POINT then this entry is the ID of the SH to which packets with this connection number are to be forwarded. The corresponding virtual circuit number is vc_{point}.
  - SH_{remote} : If the status is ACTIVE_REMOTE then it is the ID of the SH at the other end of the connection

– g_{local} :the group of the MH which is the local end the connection

Finally, each SH also keeps a table containing the mapping of the group IDs to the IDs of the member MSS's of the group. The information is of the form in Table 4.

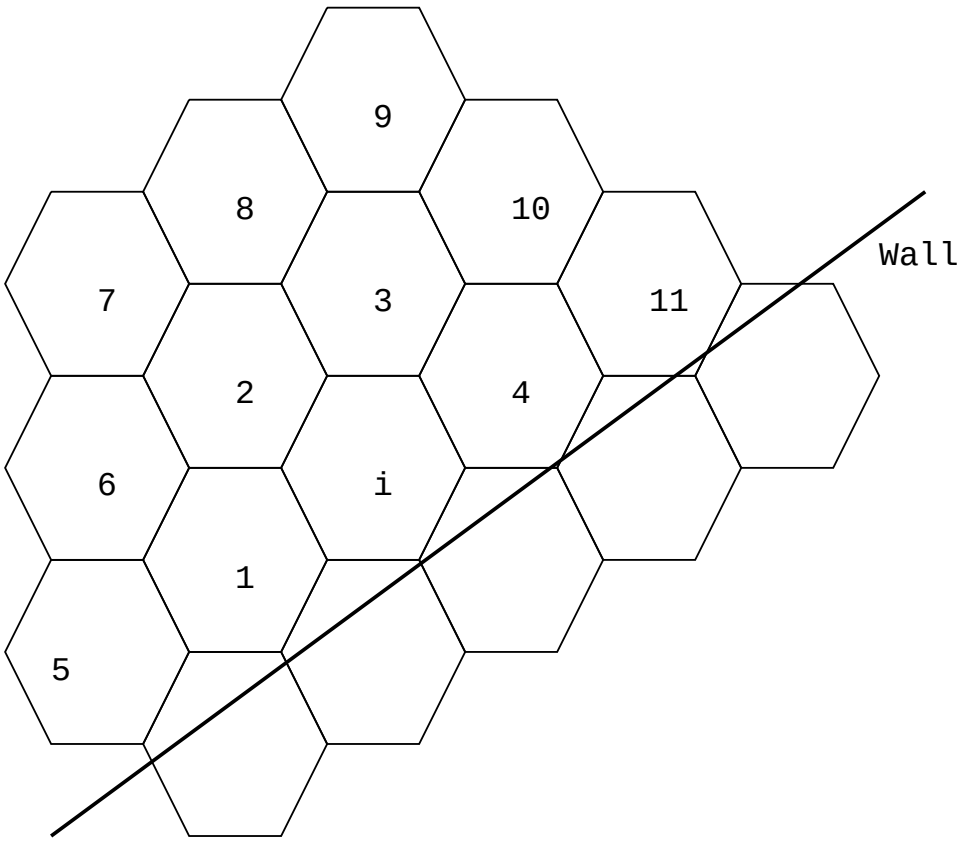


Figure 3: Groups that MSS_i can belong to.

4.1 Connection Establishment Phase

We present the protocol followed to establish a connection between two mobile hosts. Steps involved in a normal connection set up are shown in Figure 4. The steps are:

1. The MH (MH_s) wishing to initiate communication with another MH (MH_d), informs its MSS(MSS_s) of its intentions which in turn conveys this request to the SH (SH_s).

The request includes the ID of the MH with which a connection is desired and the group id of MH_s.

2. SH_s assigns a virtual circuit number (the smallest number not currently in use) and this number is conveyed to MH_s. MH_s includes this circuit number in its list of connections. The status assigned to the connection at MH_s is NASCENT. MH_s sends an ACK to SH_s.

   It may not be obvious why the SH assigns a virtual circuit number before it has actually located the destination MH. The reason is that if the SH first goes through the process of locating the destination MH, then the source MH may move out of its domain. In case we had chosen to not assign a circuit number right away, then the source MH would need to request circuit set up again. However, if a circuit number has been assigned with the status being NASCENT, the new SH will *adopt* the circuit that SH_s established. Thus the MH will not need to request a connection setup again.

3. On receiving the ACK, SH_s places the virtual circuit in its table of connections. The fields SH_{remote} and vc_{remote} are left blank for now. The status is NASCENT and the group of MH_s is set from the latest available information. If no ACK is received before a TIMEOUT interval ends, then it is likely that MH_s moved away into the domain of another SH. In this case SH_s gives up trying to establish the connection and terminates the protocol.

4. After receiving the ACK, SH_s tries to locate MH_d. It first tries to see if MH_d is present in its own subnet. It refers to the list of all the MHs registered with it. It also broadcasts a LOCATE packet in its own subnet. The LOCATE packet will contain the virtual circuit number and the ID of MH_s (and, ofcourse, the ID of MH_d).

5. If MH_d is local then the MSS whose cell it is in conveys the virtual circuit number and ID of MH_s to MH_d. MH_d includes this virtual circuit into its list of connections and sets the status of the circuit to ACTIVE_LOCAL. The MSS then responds with an ACK.

6. Upon receiving the ACK from a local MSS,SH_s also sets up an entry for the connection in its table and and informs MH_s of success. MH_s changes the status of the connection to ACTIVE_LOCAL. SH_s then terminates the protocol.

7. In the instance that MH_d is not registered with SH_s or if no ACK arrives in response to the LOCATE packet, SH_s will infer that MH_d is in another subnet. SH_s then broadcasts the LOCATE request to all other SHs (including itself in case MH_d, which is local, moved between cells in step 5) The LOCATE packet contains the virtual circuit number assigned at the initiating end and the ID of SH_s.

8. Upon receiving a LOCATE from SH_s, every SH attempts to locate the MH_d in its subnet. The SH that does find it (SH_d)then tries to establish the other end of the connection. It assigns the lowest available virtual circuit number to this connection and sends a packet to MH_d telling it about the connection request from MH_s and the virtual circuit number assigned.

9. MH_d responds with an ACK and adds the virtual circuit number to its list of connections with the status set to ACTIVE_REMOTE.

10. SH_d on receiving the ACK then includes this circuit number in it's table of connections. In addition it adds the remote virtual circuit number and the id of the remote SH (SH_s) to the entry. The status assigned to the connection is ACTIVE_REMOTE. Having done this, SH_d responds to SH_s with a notification of successful establishment of the other end of the connection. This notification is accompanied by the virtual circuit number assigned at it's end and SH_d's id.

11. If SH_d does not receive an ACK, it infers that MH_d has moved into the domain of another SH. It abandons any further attempts to establish the connection. Note that SH_s will continue sending LOCATE packets for some maximum time or until some SH responds. So eventually the connection will either be established or attempts to establish the connection will cease.

12. On receiving the packet from SH_d confirming successful connection establishment, SH_s changes the status of the connection to ACTIVE_REMOTE and informs MH_s

which also changes the status of the connection to ACTIVE_REMOTE. SH_s then terminates the protocol.

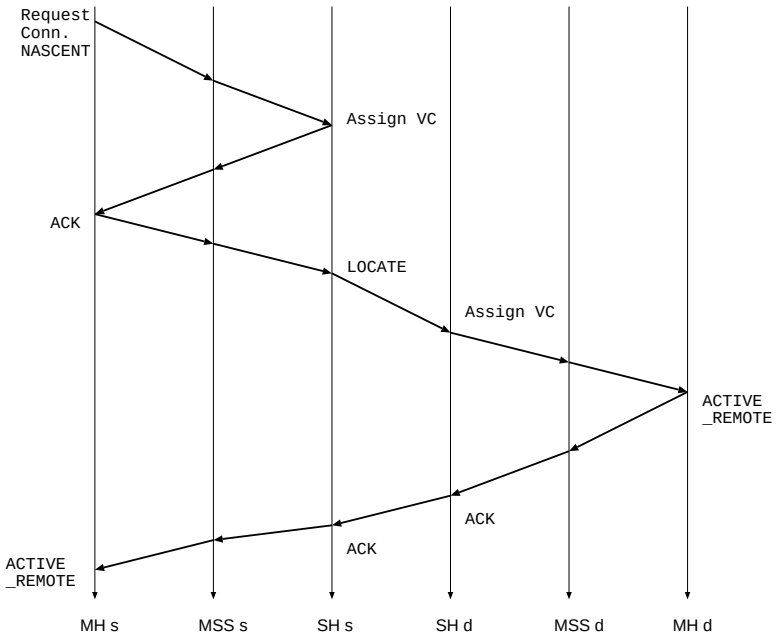


Figure 4: Normal connection establishment (remote).

This procedure completes the establishment of the connection. It is worth mentioning that the reason for requiring an ACK for each connection notification to an MH is to account for a possible movement of the MH into another subnet.

4.2 Connection Maintainence

Let us now see what happens when an MH moves between cells in the same subnet and between subnets.

Note that each MSS records the IDs of all the MHs within its purview and their current groups. This is done during the initial 'handshake' between the MSS and the new MH that enters its cell. The MSS periodically transmits a beacon signal. Upon receiving this signal, the MH sends a GREET packet. The MSS responds with a GREET_ACK. Along with the GREET the MH also sends the ID of its old MSS and its old SH. This information will be made use of by the SH in assigning a group to the MH.

If the MH has stayed in the same subnet but moved to a new MSS, this information is sent to the SH who records it and assigns a new group ID. All future packets to the MH will be multicast to this new group. In addition, the SH sends this new group ID to all *local* MHs who have connections with this MH (i.e., ACTIVE_LOCAL connections).

If the MH has moved into a new subnet, the new SH will have to *inherit* all the ongoing connections that the MH had opened previously. On discovering that it has moved into a new SH's domain, the MH first registers itself with the new SH (SH_{new}) and conveys to the SH the virtual circuit numbers of its ongoing connections and its old SH (SH_{old}) ID. SH_{new} assigns these circuits new circuit numbers (this process is called *adopting*) and then sends a message to SH_{old} that the MH has moved into its domain. It also sends the old and new connection numbers of the circuits that it has adopted.

If a connection was ACTIVE_REMOTE (at SH_{old}), SH_{old} compares the values of SH_{remote} (at SH_{old}) and SH_{new}. If these values are the same it means that the two endpoints of the connection are now in the same subnet. In this situation SH_{new} changes the status of the connection to ACTIVE_LOCAL and informs the two MHs of this change. If SH_{remote} (at SH_{old}) is not the same as SH_{new}, SH_{old} sets the status of the connection to POINT and sets SH_{point} to SH_{new} and vc_{point} to the virtual circuit number assigned by SH_{new} to this connection. SH_{old} is thus simply forwarding packets at this point. If the connection was ACTIVE_LOCAL (at SH_{old}), SH_{old} changes the status of this connection to ACTIVE_REMOTE. It sets the field SH_{remote} to SH_{new} and the field vc_{remote} to the new circuit number that SH_{new} has assigned. SH_{old} also informs the local MH of this change in the status of the connection.

4.3 Communication between two MHs

Once the connection has been established the MHs at the two ends of the connection can communicate. We describe the journey of a packet as it travels from MH_i to MH_j. There are two distinct cases to be considered:

- MH_i and MH_j are in the same subnet. In this case MH_i will give the packet to be sent to the MSS (MSS_i) of the current cell. The packet is labelled with the virtual circuit number and the current group of MH_j as known by MH_i (g_j). MSS_i includes

the current group of MH_i (g_i) and multicasts it to the group g_j. Recall that since the status of the connection would be ACTIVE_LOCAL, MH_i will have the current group information of MH_j. The multicast packet will reach MH_j and it will use the value of g_i to update its information about the current group of MH_i and vice versa. It is important for MH_i and MH_j to exchange packets quite frequently to keep each other informed (perhaps by exchanging NULL packets) about their latest group IDs.

- MH_i and MH_j are in different subnets. MH_i gives the packet to MSS_i . The packet contains the virtual circuit number (vc_i). MSS_i sends it to SH_i. SH_i looks up the entry corresponding to the virtual circuit number vc_i and finds the values for the remote virtual circuit number (vc_j) and the ID of the SH at the other end of the connection (SH_j). SH_i now sends the packet labelled with vc_j to SH_j. SH_j on receiving the packet labelled with vc_j looks up the entry corresponding to this connection and finds out the status of the connection. The status will be either ACTIVE_REMOTE or POINT. If the status is POINT then SH_j is simply serving to forward packets for this connection. SH_j looks up the entry SH_{point} in the table. It forwards the packet to SH_{point} labeled with vc_{point}. If however the status of the circuit is ACTIVE_REMOTE then SH_j will multicast the packet to the group g_{local} in its subnet.

5 Flow Control and Buffer Management

The problem of flow-control and buffer management is quite interesting in our system because of the three-tiered architecture used. We use one flow-control mechanism for communication between an MH and an MSS, and a second, similar, flow-control mechanism for communication between an MSS and its SH. In order to explain these flow-control schemes better, we first need to discuss the buffer management protocol. Let us focus on a specific MH, m, that has k open connections and an MSS, M.

Buffer allocation at M: Let us assume that initially M does not lie in m's current groups but after some time, m moves close enough that M is included in a current group. Let us further assume that m *does not* lie in the cell belonging to MSS M.

When M is first included in m's group, it allocates b buffers for m (enough to handle all open connections). b is smaller than some maximum. If there are no free buffers available at M (i.e., M is already included in several groups), then M simply cannot support m's connections.

The b buffers are treated as a FIFO queue with new packets (for m) put in at the tail of the queue. If M receives more than b packets for m, it retains only the latest b packets (i.e., once all b buffers are full, packets are dropped from the head of the buffer queue to make room for more recent packets), discarding duplicates.

Flow-Control at M: While m is not in M's cell, M does not communicate with m and therefore does not participate in any type of flow-control for m. Let us assume that m eventually moves into M's cell.

During the initial handshake, m sends a packet to M containing the sequence number of packets m expects to receive for *each* of its k open connections. In addition, for every open connection m also includes a *permit*. A permit for a connection is a number representing the number of packets that m can buffer locally for that connection.

Upon receiving the above information, M begins transmitting packets to m starting with the appropriate sequence numbers for each connection and upto the maximum allowed by the permit for that connection. The order in which the connections are serviced may depend upon some priority mechanism (e.g., real-time video is transmitted before a data file). In this paper, however, we are not concerned with this aspect of the problem. If m stays in Ms cell long enough, it transmits packets with sequence number and permit information as above whenever its local buffers become free.

Periodically (while m is in M's cell) M transmits a message to the SH containing a sequence number for each of m's connections along with a permit for each connection (the value of this permit depends upon b and the number of packets of each connection that are in M's buffer). Thus for some connection i if M has packets numbered upto $j + l$ in its buffers but if the latest packet transmitted to m is numbered j then the sequence number for this connection sent to the SH is $j + 1$ (not $j + l$). On receiving this message the SH *discards* all packets for this connection (for m) numbered upto

j because m has successfully received them. Note that if M had instaed transmitted $j+l$ to SH then it is possible that m may not receive all packets. For example, if m moved out of Ms cell into a non-group cell which does not initially have packets for m then m never receive packets $j+1$ to $j+l$ since the SH has already discarded them.

Upon receiving the above packet from M, the SH begin multicasting packets for each connection starting from the given sequence number and upto a maximum sequence number as determined by the value of the permits.

6 Reducing Buffer Overhead

Group-based communication, while providing seamless communication to MH nodes, is expensive when it comes to buffer overhead at the MSS nodes. Thus, if g is the current group for some MH then each packet for this MH is buffered $|g|$ times (one copy is present at all MSSs that belong to g). In this section we present an approach to reduce this buffer overhead while still maintaining seamless communication with the MH.

Before proceeding with the description of our new buffering polocy, it is important to point out that in most building environments the size of a group, $|g|$, is typically of the order of 2 or 3. This is because of the severe limitations placed by the building architecture. For example, if we consider the case when an MH is in a room then the current group will consist of the neighboring cells that can be got to through a doorway. Similarly, if we consider an MH walking along a corridor in a particular direction, the group will consist of the current cell, the next cell in the corridor that lies in the direction of motion and possibily a neighboring room that opens out into the corridor.

6.1 Delayed Buffering

It is easy to see that once an MH moves into a cell, it spends some time there before moving on into another cell. Let us denote this *cell latency* by δ. Since the MH remains in the current cell for δ time, there is no need to buffer packets at other MSS nodes of its group for at least this period of time. In this section we formalize this idea and derive a simple expression for buffer overhead.

Let δ (cell latency of an MH) have a pdf f_δ. Define a *threshold*, τ, that represents the time the MH is *expected* to spend in the cell. Let $p_1, p_2, \ldots p_k$ denote the sequence of packets that arrive for transmission to the MH. Further, let $a(p_i)$ denote the time the packet p_i arrives at the current MSS and $d(p_i)$ the time when the packet is expected to be fully transmitted to the MH[1] assuming it remains in the same cell (i.e., $\delta > d(p_i)$). Our buffering policy can then be stated as follows:

> **Buffering Policy:** Let $1 \leq m \leq k$ be the smallest index such that $d(p_{m-1}) < \tau < d(p_m)$. Then, only buffer packets $p_m, \ldots, p_k$ at other MSS nodes of g.

In Figure 5(a) we show the sequence of packet arrivals and departures for the case when $\delta > \tau$. Note that the transmission of p_k is incomplete because the MH moves out of the cell in the middle of it. According to our buffering policy only packets $p_m, \ldots, p_k$ will be buffered at other MSS nodes of g. Figure 5(b) illustrates the case when $\tau > \delta$ (i.e., the MH moves out of the cell before expected). In this case, no packets were buffered at other MSS nodes of g. However, packets $p_m, \ldots, p_l$ will have to be *retransmitted* by the SH to the new MSS. Note that up until time δ packets $p_{l+1}, \ldots, p_k$ have not yet been transmitted by the SH and do not contribute to either the buffering cost or the retransmission cost.

What is the cost of our new buffering policy? From the discussion above it is clear that there is still a buffering overhead (as in Figure 5(a)). In addition, however, there is a retransmission cost caused because of our delayed buffering policy. Further, when $\tau > \delta$ there is a break in service at the MH while packets get retransmitted. Let us formally define these three costs.

- Define the *buffer overhead* per MH per cell as the *fraction* of packets that are buffered at MSS nodes other than the current one. In our original discussion of the group-based communication protocol, (section 3) this fraction is 1 since *all* packets are buffered

[1] Recall that the channel allocation policy followed in most cellular systems is TDMA. For example, [5] discusses a TDMA scheme where the slots are 8 bytes long. Thus, it will take a long time for a large packet to be completly delivered to the MH and the MH can potentially move out of the cell in the middle of transmission. In such a case, the new MSS will have to retransmit the complete packet.

at all MSS nodes belonging to g. It is noteworthy that the buffer overhead is *zero* if $\tau > \delta$. On the other hand, if $\tau < \delta$, this cost may be written as,

$$\mathcal{B} = \frac{k - m + 1}{k}$$

where packets $p_1, p_2, \ldots, p_k$ arrive for transmission to the MH in the current MSS and, $d(p_{m-1}) < \tau < d(p_m)$ (see Figure 5(a)). It is important to note that the *total* overhead is actually $\mathcal{B}(|g| - 1)$. However, $\mathcal{B}$ is a more generic measure because it is independent of the group size.

- The *retransmission cost*, when $\tau > \delta$, is the *number* of packets that will have to be retransmitted. Note that if were not using delayed buffering this cost would be *zero*. We can write this cost as,

  $$\mathcal{R} = |S|$$

  where, $S = \{p_m, \ldots, p_l\}$ and $a(p_i) < \delta < d(p_i), m \leq i \leq l$ in Figure 5(b).

- The third cost we must take into account is, $\mathcal{P}$, the *probability of break in service* at the MH. This happens when the MH moves into a new cell and has to wait for packets to be forwarded. $\mathcal{P}$ is *zero* in our original group-based protocol. If we use delayed buffering, however, when $\tau > \delta$ there will be a break in service because packets have not been prebuffered at group elements. We can therefore write this cost as,

  $$\mathcal{P} = Pr[\tau > \delta]$$

  Note that for the traditional approach $\mathcal{P} = 1$ because whenever the MH moves between cells, there is a break in service while packets get forwarded.

6.2 Experimental Study

In order to study the cost-performance behavior of our delayed buffering policy we ran simulations and documented the buffering cost, retransmission cost and probability of interruption of service as a function of τ. We assume an MH moves between several cells while maintaining an open connection. Inter-packet arrival time for this MH (at MSS nodes) are

exponentially distributed with mean 45 packets/sec and packet transmission time (from the MSS to the MH) is also exponentially distributed with mean 50 packets/sec. We only look at communication *to* the MH because there is no overhead cost for communication from the MH to the MSS

We assume the *mean* cell latency of the MH is 5 seconds. In the first experiment we assume that f_δ is a *normal distribution* with $\sigma = 0.288$. In the second experiment f_δ is exponentially distributed. Finally, the time taken for the MH to announce itself to a new MSS is taken to be 100 msec (a constant).

During our experiments we noted that, on occassion, packets do not necessarily get delivered to an MH in either the current cell or even the next one as in Figure 6(a). Cases such as in Figure 6(b) also occur. Here packets p and q arrive in the first cell where the MH spends δ_1 time. These packets do not get transmitted in the next cell (perhaps because δ_2 is small or because there is a large backlog of untransmitted packets). They are finally transmitted in the third cell. These packets therefore contribute *twice* to the buffering cost and retransmission cost. We incorporated these extra costs in our simulations.

For each experiment we also determined the number of untransmitted packets forwarded from the old MSS to the new MSS in the *traditional* approach (denote this number by $\mathcal{T}$). For simplicity the *time* to forward these packets is taken to be zero. For purposes of comparison, we calculated the ratio $\mathcal{R}/\mathcal{T}$ for different values of τ (note that $\mathcal{R}/\mathcal{T}$ is zero if delayed buffering is not used).

Figure 7 shows the relationship of buffer overhead $\mathcal{B}$ as a function of τ (all data points lay well within the 90% confidence intervals). As τ increases the buffer overhead drops to zero because almost no packets are buffered at other MSS nodes of the group. Figure 8 shows the *fraction* of packets retransmitted $\mathcal{R}/\mathcal{T}$ as a function of τ. Notice that for large τ the fraction of retransmitted packets increases because the probability that $\tau > \delta$ (i.e., $\mathcal{P}$) increases, see Figure 9. Viewing Figures 7 and 8 together it appears that for τ close to 4.0 the number of retransmitted packets is small (below 10% of $\mathcal{T}$) while the buffer overhead is less than 20%. Thus if $|g| = 6$ then the buffer overhead is twice that required if we use the traditional approach (i.e., no groups) but the fraction of retransmitted packets is 1/10 th and the probability of a break in service at the MH (because of delay in retransmitting

packets) is less than 0.2.

Figures 10 and 11 illustrate the behavior for the case when f_δ is the exponential distribution. Here for a value of τ equal to 4.0 the buffer overhead is 40% and the retransmission overhead is about 45%. The savings in this case are not as pronounced as in the case of the normal distribution but are significant nevertheless.

7 Summary of Results and Extensions

We have shown that the group-based protocol ensures *seamless* communication with the mobile users as they move between cells. This is an important quality of service parameter in the picocellular environment because cell latency here is of the order of 4–5 seconds and a service interruption every 4 or 5 seconds could be very annoying (particularly in an audio connection). The main drawback of our approach is the buffer overhead. In section 6 we introduce the delayed buffering policy that reduces this overhead significantly while maintaining a seamless connection with a high probability.

It is interesting to observe that the distribution of cell latency times affects the performance of our delayed buffering policy. Thus if f_δ is a normal distribution delayed buffering performs well on our three measures of performance. If, on the other hand, f_δ is the exponential distribution the performance is not as good. Therefore, it will be interesting to

experimentally determine the cell latency distribution for real in-building environments and tailor a buffering policy that performs well in those situations.

An interesting consequence of our approach is the flexibility in terms of cost-performance tradeoffs from the point of view of the MH. Thus, for some applications (e.g., data transfer) it may not be important to guarantee seamless communication (only reliable communication is necessary). In these situations we could use groups that consist of only the current MSS. If, on the other hand, it is important to maintain seamless communication (audio channel of a videophone service) then we can use larger groups. Thus we could use different groups for different types of services provided to the mobile user (and charge the user for these services accordingly).

8 Conclusions

In this paper we have presented a novel architecture for picocellular networks. Furthermore, we presented a group-based communication protocol for mobile users that ensures uninterrupted service to the MH. We compared our approach with existing microcellular approaches and showed that, in comparison, the traditional approaches exhibit poor performance in terms of retransmission overhead and frequent breaks in service to the MH.

References

[1] A. Acharya and R. Badrinath, "Delivering multicast messages in networks with mobile hosts" *13th Intl. Conf. on Distributed Computing Systems*, May 1993.

[2] Dale Buchholz,Paul Odlyzko,Mark Taylor and Richard White, "Wireless In-Building Network Architecture and Protocols" *IEEE Network Magazine*, pp 31-38, November 1991.

[3] D. Duchamp, Steven K. Feiner and G. Q. Maguire, "Software technology for wireless Mobile computing" *IEEE Network Mag.*, pp 12-18, November 1991.

[4] David J. Goodman, "Cellular Packet Communications" *IEEE Trans. on Comm.*, vol. 38, no. 8, pp 1272-1280, August 1990.

[5] David J. Goodman, "Trends in Cellular and Cordless Communications" *IEEE Communications Magazine*, pp 31-40, June 1991.

[6] F. Teraoka and M. Tokoro, "Host Migration Transparency in IP Networks: The VIP Approach", *SIGCOMM*, Vol. 23, No. 1, Jan 1993, pp. 45-65.

[7] J. Ioanidis, D. Duchamp and G. Q. Maguire, "IP-based protocols for mobile internetworking" *Proc. of ACM SIGCOMM'91*, pp 235-245, September 1991.

[8] Fumino Teraoka, Yasuhiko Yokote and Mario Tokoro, "A network architecture providing host migration transparency" *Proc. of ACM SIGCOMM'91*, pp 209-220, September 1991.

[9] R. Steele, "The cellular environment of lightweight handheld portables" *IEEE Commun. Mag.*, vol 27, no. 7, pp 20-29, July 1989.

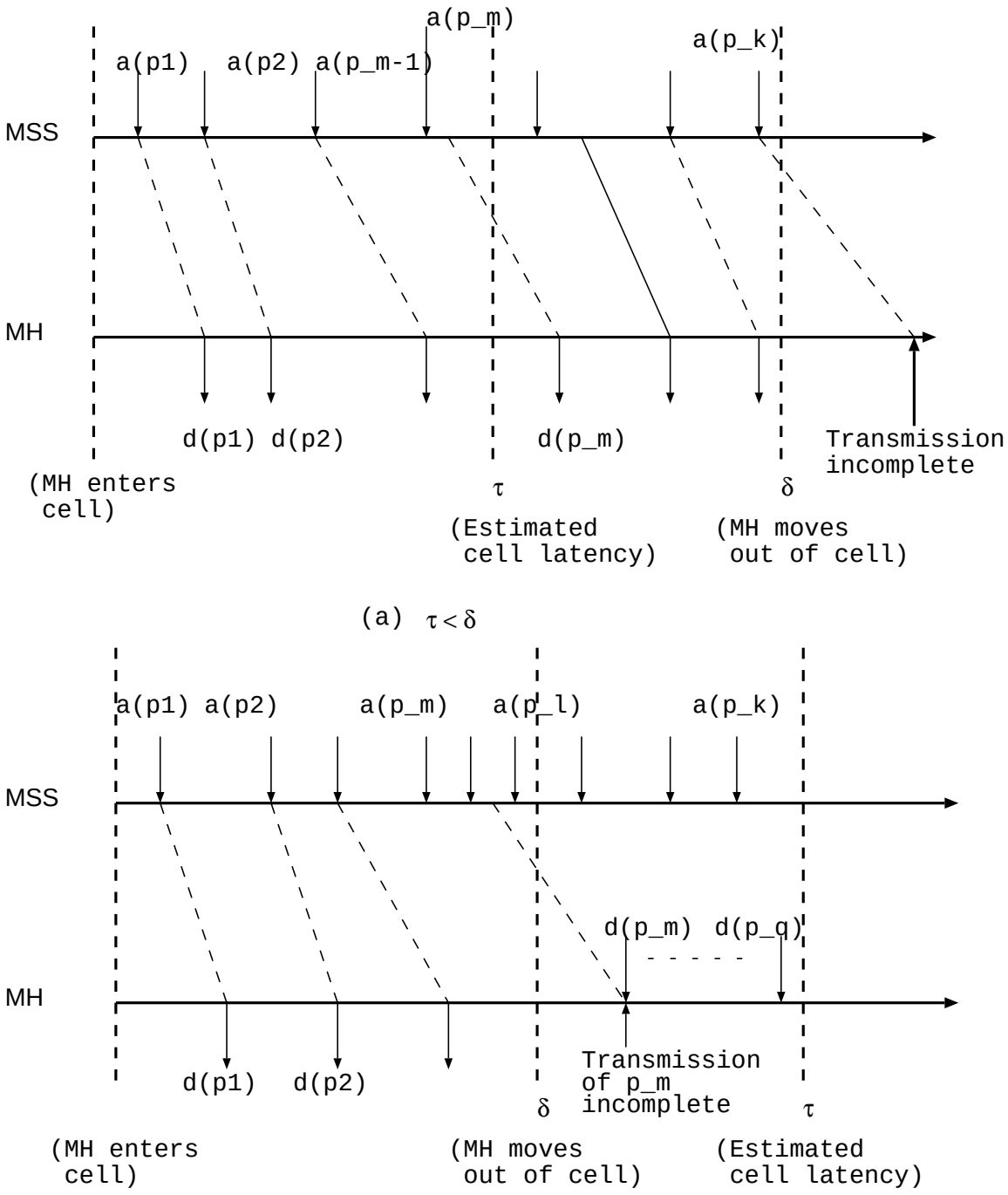
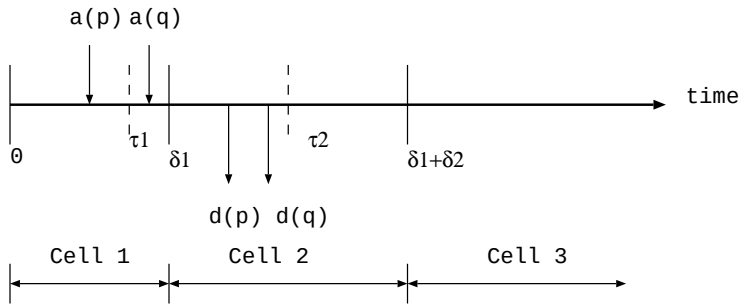
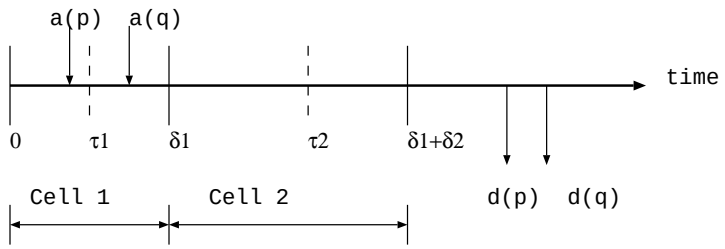


Figure 5: Effect of estimated cell latency on delayed buffering.



(a) Packets are forwarded once.



(b) Packets are forwarded many times.

Figure 6: Packets may follow a MH for many cells.

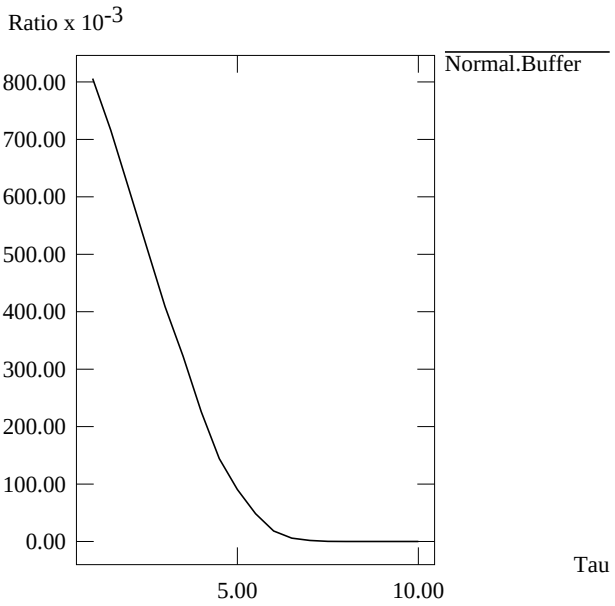


Figure 7: Buffer Occupancy at Group elements – *Normal distribution.*

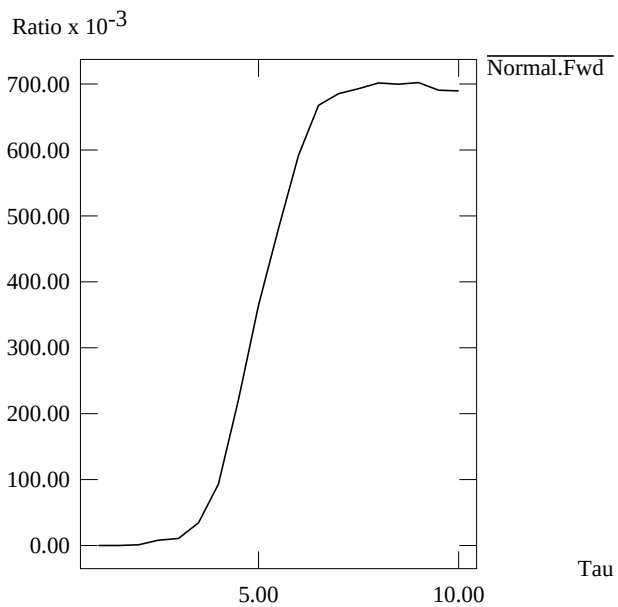


Figure 8: Packets Forwarded – *Normal distribution.*

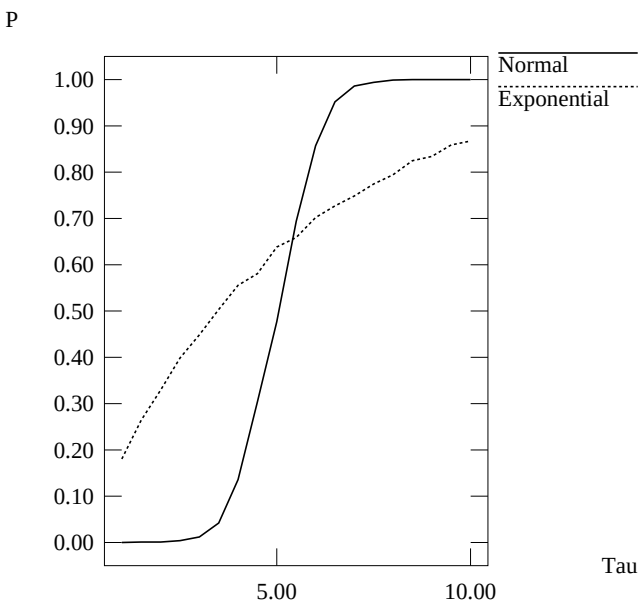


Figure 9: Probability of interruption of service, $\mathcal{P}$.

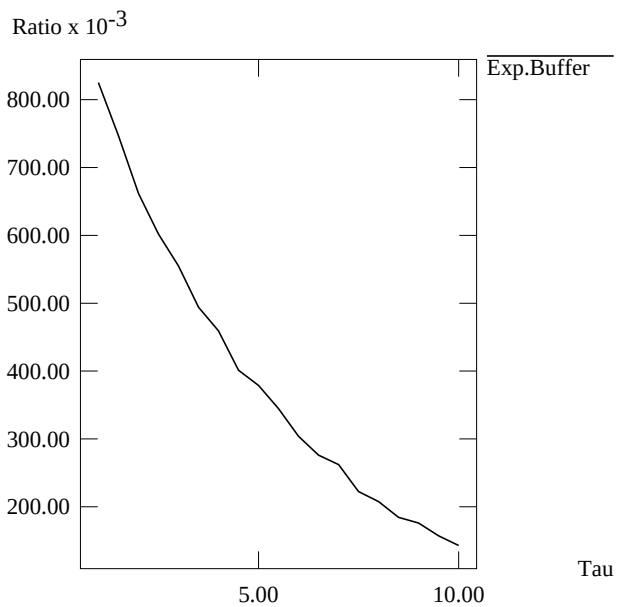


Figure 10: Buffer Occupancy at Group elements – *Exponential distribution.*

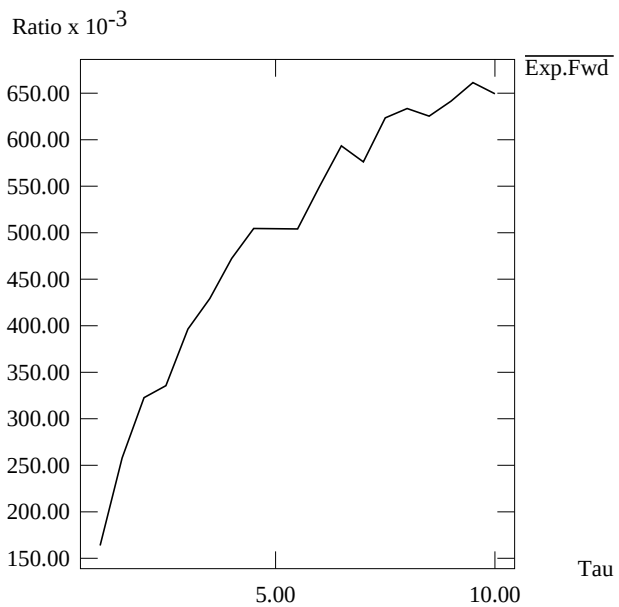


Figure 11: Packets Forwarded – *Exponential distribution.*