Wireless Networks 7, 21–31, 2001

Distributed Protocols for Networks with Mobile Users – The Mobilizer Approach

BOAZ MIZRACHI, MOSHE SIDI and ISRAEL CIDON
Department of Electrical Engineering, Technion, Haifa 32000, Israel

Abstract. This paper introduces a novel approach, called Mobilizer, of operating synchronous communication protocols in cellular mobile environment. First, we present a distributed protocol, called Mobile Propagation of Information (MPI), for broadcasting information in mobile environment. Then, we present the Mobile Propagation of Information with Feedback (MPIF) protocol, which can be used to implement the Mobilizer approach, i.e., enable broadcast-based synchronous protocols run over distributed networks with mobile users. We prove the correctness of the protocols and show that the additional message complexity overhead, induced due to the Mobilizer, is linear with the number of users' movements.

Keywords: distributed protocols, mobile users, Mobilizer, PIF

1. Introduction

One of the important tasks applied in a communication network is enabling the broadcast of information within the network components, generally referred to as nodes or stations [6]. The term broadcast is used to describe the operation of propagating a piece of information, termed message, from some source component to all other connected components in the network [6]. This message may be transmitted by the nodes to each other through communication links, which induce some delay upon its propagation [5]. The communication network can be abstractly described by an undirected graph, where nodes represent stations, and vertices represent communication links connecting the stations. Any task, that is needed to be performed by the network stations, should be translated into an algorithm to be carried out by one or more of the stations. As the graph gets bigger, and links' delay gets larger and unknown, some synchronization mechanism must be deployed, since the communication processors are located in remote sites. Therefore, a distributed communication protocol must be operated at the nodes, to enable performing the required algorithm.

Such a distributed protocol, developed for broadcasting messages along distributed communication networks, is the PI – Propagation of Information protocol, described in [5]. Its purpose is to enable some station to initiate a propagation of a message, to be received eventually by all connected stations. It is based on the simple fact, that if every station will send this message to its immediate neighbors, as soon as it receives it for the first time, then eventually all connected stations will receive it. This protocol is considered to be one of the basics of distributed protocols. It is also used as an underlying protocol, which with various additional parts constructs series of other distributed protocols, such as leader election and topology updates protocols.

An important feature for a broadcast protocol is for the source station to receive a feedback indication, guaranteeing that all connected stations have already received the message propagated by the source. Owning such an ability, a source may use it to determine whether to move to the next phase of some more general protocol it is running, and hence, synchronize between distributed processes. The basic protocol having such a feature is the PIF – propagation of information with feedback [5]. This distributed protocol is based on the PI, with the following extension: after a station receives a copy of the propagated message from each of its immediate neighbors it sends the message to the station it received it from for the first time. When the source station gets the message back from its own immediate neighbors, it is guaranteed that all stations have received the message, and the protocol then terminates.

Developing this protocol made it possible to move another important step forward in the field of distributed communication protocols. When having some useful protocol operating over a synchronized communication network, one may want to implement it over a distributed a-synchronized network, as described earlier. It can be most efficient if there would be a unified way to convert such a synchronized protocol to a distributed unsynchronized one. We might also want the message complexity overhead to remain low.

A protocol combining these features is the Synchronizer [2]. It is based on the PIF, running over some of the network graph edges. In this paper [2], some variations of this Synchronizer protocol are described, each having its advantages and disadvantages, regarding message complexity and run time. Basically, the Synchronizer is a distributed protocol, which can be combined with any other synchronous protocol, enabling its operation efficiently over a distributed communication network.

As communication stations became smaller in their physical dimension, the need for supporting their mobility has emerged. Still, it is desired to keep them connected to the network even while they are moving. A solution for this problem was the cellular communication system, which consists of base stations communicating through a static network (referred to as the backbone), and mobile users com-

municating with the stations through some wireless channels [3].

With the increase in the number of users, still having the limitations of available radio frequency bandwidth, the cells became smaller. Thus, the movements between cells, referred to as handoff procedures, had become more rapid. Now, a protocol designed for such an environment must take these facts into account, and handle rapid movements of users. Indeed, protocols were developed, especially for the cellular communication environments, each supporting a specific function to be executed at the network [1–4].

Our purpose is to design a distributed protocol, which can be combined with any given protocol, where the last was designed originally for synchronized static communication network. The combined protocol then can be executed over a cellular network, containing highly mobile users. Therefore, we call it the Mobilizer – mobilizing protocols over cellular networks.

We introduce here the first step towards this desired Mobilizer. Our goal is to develop a protocol that can mobile any given PIF based protocol, designed originally to operate over static synchronous network. Also, we would like to have the overhead message complexity as low as possible when combining this mobilizer with the static protocol, so the combined protocol will remain efficient as the original static one.

First, we introduce a protocol, called Mobile PI (MPI), which accomplishes the same task as PI did, but over a mobile environment, i.e., enables a broadcast of series of messages to mobile users. The MPI, like the PI, guarantees that all active users will eventually receive the series of messages at a FIFO order. Naturally, some additional message complexity is expected due to supporting users' movements. We show that MPI message complexity is bigger only by relatively small addition compared to the static PI. This complexity addition is a linear function of the number of the users' movements, each movement adding only a constant number of messages. We also show that MPI converges to PI when no handoff procedures are being performed during the propagation of the message.

Then, we introduce a second protocol, called Mobile PIF (MPIF), which like the PIF, gives the source station the feedback indication, after all users have received the broadcasted message. The MPIF message complexity is also bigger than the static PIF one only by a linear addition. This protocol also converges to the static one when users do not move during the broadcast. Since one can combine the MPIF with any other static synchronous protocol at will, we developed a mobilizer for PIF based protocols, with low message complexity overhead.

The rest of this paper is organized as follows. The next section contains precise definitions of the model and the problem. In section 3 we give an overview of the MPI protocol, its pseudo code and properties. In section 4 we describe the MPIF protocol and its properties.

2. The model

Consider a cellular wireless system with N cells and P mobile users. Each of the cells is controlled by a unique stationary node called Base Station (termed shortly station). Stations communicate with each other through a static communication network, termed the base network. Mobile users (termed shortly users) communicate via a wireless channel with one of the stations at a time. The users cannot communicate with each other directly. This model is very much the same as described in [3].

2.1. Base stations

The static network model is a standard fixed point-to-point communication network. The base network is described by a connected undirected graph $G = (V, E)$, where the vertices of the graph represent the stations ($|V| = N$) and the edges represent bidirectional communication channels between them. A station may communicate directly only with its neighbors, and messages to non-neighboring vertices are sent along some path connecting them in the graph. The base network is assumed to be reliable, i.e., a message sent on a communication link will eventually be received correctly after some finite time. We denote the base stations as $B_i,\ i = 1, \ldots, N$.

2.2. Mobile users

The mobile users' model is based on a cellular wireless mobile stations model. A user, denoted by $M_k,\ k = 1, \ldots, P$, can be in one of the following states:

1. Communicating directly with some station B_i through a unique wireless channel between them. In this state, M_k is assumed to be connected with B_i through logical bidirectional edge, i.e., any message sent between them will eventually reach its destination correctly after some finite time, unless M_k moves to another cell.
2. Performing handoff procedure from B_i to B_j, i.e., changing the station with which M_k communicates, from B_i to B_j.
3. Disconnected, having no connection with any of the stations.

Each of the users possesses non-volatile memory which holds its unique identification M_k, its current station identification B_i that it is communicating with, and the last message identification number it received from some station.

2.3. Handoff model

Our handoff procedure model is a practical one. The cells' ranges overlap partially. Thus, consider some M_k, which is communicating with station B_i. When M_k reaches the border of the B_i's cell it enters some B_j's cell, M_k detects B_j's transmission and decides at some time to change its

communication channel to B_j instead of B_i. At that moment it sends a message *Detect*(M_k, B_j, B_i) to B_j, indicating intention of M_k to communicate with B_j. This message includes, besides of M_k and B_j's identifications, the identification of the last station B_i that M_k communicated with. B_j detects the existence of M_k at some time, by receiving the *Detect* message, then it sends a *Join*(M_k, B_j) message to M_k. This message instructs the user to register B_j as its new base station. M_k receives this message and replies to B_j with *Connect*(M_k, B_j, l, S), indicating the last message number l, originated by S that it received correctly. Then, B_j sends B_i a *Free*(B_j, B_i, M_k) message, meaning B_j takes responsibility for M_k from now on. This last message is implemented only in cases where a feedback for reception is needed by the source S, i.e., in Mobile PIF.

Two stations are termed handoff adjacent cells if a user can perform a handoff procedure between them. At first we will not assume any relation between the graph topology and the physical layout of the stations, i.e., neighboring stations are not necessarily handoff adjacent, and vice versa.

2.4. Mobility assumption

Users can move very fast in mobile environments, especially when cells cover small ranges. We would like to enable users in our model to move as fast as they like, so we do not upper bound the rate of handoff procedures a user can perform. We do not even assume that a user completes its last handoff procedure *before* is starts a new one, i.e., when a station sends a *Join* message to a user when detecting it entering its cell, the user can move quickly *before* receiving this message and be detected by another station. This, naturally, makes our model more realistic, but it induces more difficulties in developing an algorithm under such assumptions.

3. Mobile PI (MPI)

PI is a distributed algorithm enabling a single vertex to broadcast messages over the network to all other vertices. After the source vertex had transmitted the message to its immediate neighbors it is ensured that after a finite time interval the message will be propagated and received by all other vertices in the connected network. We recall that the source node does not have any indication *when* this event occurs.

3.1. Problem description

The primary goal is to perform the PI protocol over the stations and users. When PI is performed over a network containing mobile users, the mobility must be taken in account. A user can move from a cell that its station has not received the broadcasted message yet to a cell that its station already broadcasted the message to its users, and thus, will not receive the message. The goal is to develop a protocol that will prevent such cases and guarantee that broadcast messages arrive at every mobile user. At first we ignore users which are disconnected or become disconnected while the broadcast is being propagated. It is naturally assumed that these users do not intend to be part of the network at that time, and thus, they are not guaranteed to receive the broadcasted messages.

It is assumed that only a station might initiate a broadcast. If a user wishes to initiate such a broadcast it can instruct its current station to do so.

3.2. Definitions

Let us define now the notations, messages and data structures used by the protocol.

3.2.1. Notations

- M_k, mobile user, uniquely identified in the entire network.
- B_i, base station unique identifier.
- S, base station that is the source of the broadcasted message.
- d_l^S, message number l, $l = 1, 2, \ldots,$ originated by source station S.
- G_{B_i}, the set of all stations which are neighbors of B_i.

3.2.2. Messages

- *Msg*(d), data message d sent between two stations.
- *Msg*(d, B_i), data message d received from neighbor station B_i.
- *Trans*(B_i, M_k, d_l^S), data message d_l^S which B_i sends M_k. If $M_k = All$ then d is addressed to all locally connected users.
- *Detect*(M_k, B_i, B_j), information, transmitted by a user, and received by a new station detecting the user entering its cell. This message represents the act of detecting the user. M_k and B_i are the user and the new station identification correspondingly, and B_j is the last station the user communicated with.
- *Join*(M_k, B_j), message B_j sends M_k when it detects it at first at its cell.
- *Connect*(M_k, B_j, l, S), a reply for *Join*(M_k, B_j) that M_k sends B_j, indicating that the last message M_k received correctly from a station is d_l^S.

3.2.3. Data structures

- $Laststation_{M_k}$, M_k internal memory register holds station identification from which the last *Join* message received.
- $h_{B_i}^S$, B_i internal register holds identification number of the last message received, originated by S.
- $h_{M_k}^S$, M_k internal register holds identification number of the last message received, originated by S.

3.3. Problem definition

Let us first introduce some definitions to help us define the problem more formally.

Definition 1. A user M_k has *completed a handoff procedure* with a station B_i at $t = t_1$ if:

- B_i received a *Connect* message from M_k at $t = t_0$, and
- if at $t = t_0$ station B_i has messages for user M_k, then M_k received these messages before $t = t_1 > t_0$.

Definition 2. A user M_k is *eventually connected after time* t_0 if there exists some $t > t_0$ such that M_k has completed a handoff procedure with some station at t.

We denote by S the source station that initiates the broadcast. We denote by T_{last} the time upper bound it takes a message to be propagated from S to the last station to receive the message. We denote by t_l the time S started the propagation of $Msg(d_l^S)$. We shall now define the subset of ACTIVE users that are guaranteed to receive the propagated message. Formally, ACTIVE – the subset of users, each has its own t_2, $t_2 > t_l + T_{\text{last}}$, such that this user has completed its handoff procedure with some station at $t = t_2$. Intuitively, if a user moves too fast within the network cells without completing a single handshake with one of the stations, then it cannot be guaranteed to receive the message.

Let us now define the problem more formally. Suppose S initiates a broadcast of a series of messages $d_l^S, l = 1, 2, \ldots$, each at time t_l ($t_1 < t_2 < \cdots$) to all users. It is needed to guarantee that all users in the ACTIVE subset will eventually receive the messages $Msg(d_l^S)$.

3.4. MPI – An informal description

We will now describe a new protocol, which we call Mobile PI (MPI), that solves the problem described above. MPI is based on PI so that in the static network (the backbone) MPI is evolving the same. At time t_l S sends d_l^S to its immediate station neighbors and transmits it to users connected to it. When B_i receives d_l^S for the first time it forwards it to its other neighbor stations, and broadcasts it to all users currently connected to it. Other copies of d_l^S received later by B_i are ignored.

When a user M_k receives d_l^S for the first time it accepts it and copies its identification l into its internal variable $h_{M_k}^S$. Other copies of d_l^S received by M_k are ignored.

When M_k enters a new cell managed by B_j, and B_j detects M_k as described in the model before, B_j sends M_k a *Join* message instructing it to communicate the network through B_j from now on. M_k responds by a *Connect* message, informing B_j with the identification of the last message d_l^S M_k received from the network. If B_j already broadcasted S messages with identifications greater than l then it transmits the newer messages to M_k. In this way, it is guaranteed that users in movement will not miss messages already broadcasted in the cell they perform a handoff procedure to.

It is assumed that S wants to send series of messages with serial numbers as identification, $l = 1, 2, \ldots$. It is also needed to guarantee that users receive these messages in the *same* order they are sent (FIFO order).

Let us recall that our major goal is to construct a protocol that can enable a distributed algorithm, designed for static network, to be executed over a mobile users environment. Thus, after developing the MPI protocol it is needed to combine the original algorithm with the MPI. This is done, naturally, by inserting the algorithm's messages into the data field d of the MPI messages $Msg(d)$.

3.5. MPI – The protocol

Let us now describe the protocol more formally.

3.5.1. Assumption

Before entering the protocol we assume the following:

- $h_{B_i}^S = 0$, $h_{M_k}^S = 0$ at all users and stations.
- At t_0 S receives $Start_0$ from the outer world.

3.5.2. Algorithm running at station B_i

For $Start_l$ or $Msg(d_l^S)$:
  if $(h_{B_i}^S < l)$ then
    $h_{B_i}^S = l$,
    send $Msg(d_l^S)$ to all $B_k \in G_{Bi}$,
    send $Trans(B_i, All, d_l^S)$.
For $Detect(M_k, B_i, B_j)$:
  send $Join(M_k, B_i)$.
For $Connect(M_k, B_i, l, S)$:
  while $(l < h_{B_i}^S)$ do
    $Trans(B_i, M_k, d_l^S)$,
    $l = l + 1$.

3.5.3. Algorithm running at user M_k

For $Trans(B_i, M_k, d_l^S)$ or $Trans(B_i, All, d_l^S)$:
  if $(h_{M_k}^S = l - 1)$ then
    accept d_l^S,
    $h_{M_k} = l$.
For $Join(M_k, B_i)$:
  send $Connect(M_k, B_i, h_{M_k}^S)$,
  $Laststation_{M_k} = B_i$.
If needed:
  send $Detect(M_k, B_i, Laststation_{M_k})$ to local station B_i.

3.6. Properties of MPI protocol

The properties of the protocol appear in the following theorems, whose proofs appear in the appendix.

3.6.1. Correctness

Theorem 1. Suppose S sends $Msg(d_l^S)$ at time t_l, then:

(1) there exists some finite time interval T_l so that *all* base stations B_i have already received $Msg(d_l^S)$ before $t = t_l + T_l$,

(2) if $M_k \in$ ACTIVE then M_k will eventually receive $Msg(d_l^S)$.

Theorem 2. Let M_k be a user receiving messages sent by a source station S, then:

(1) suppose M_k receives message d_l^S before receiving message d_m^S, then $l < m$;

(2) user M_k receives message d_l^S in the fastest way possible, under the assumptions of the model described above.

3.6.2. Message complexity

Let us denote by Z the number of handoff procedures being held at $t \in [t_l, t_l + T_l]$ by users in the network.

Theorem 3. For a broadcast of the single message d_l^S, the upper bound for message complexity of MPI protocol is $2|E| + P + 4Z$.

Reduction 1. When no handoff procedures are being held at $t \in [t_l, t_l + T_l]$ (i.e., $Z = 0$), the message complexity converges to the static PI protocol complexity, i.e., $2|E| + P$.

3.7. Disconnected users – Proposals

The given MPI protocol does not support disconnected users, i.e., if some M_k was disconnected at some time interval $[t_1, t_2] \cap [t_l, t_l + T_l] \neq \emptyset$, then it is not guaranteed that M_k will receive message d_l^S. This problem may be realistic in wireless communication. It can be observed easily that if we allow disconnected users to participate in this protocol then it is required to have the broadcasted messages to be stored at some point(s) of the network. We can show that in the worst case, when messages are broadcasted rapidly and users are being disconnected for infinite time period then unbounded memory space is required at the network to store these messages in case some disconnected user will reconnect and acquire them. Even if users are not disconnected from the network for too long, stations which rapidly generate messages may cause congestion in the network.

It is also reasonable to assume that in many applications, when some disconnected user is reconnected, it will not be needed to send it *all* the messages it missed. One can be satisfied with sending it *some* of the last messages broadcasted at the network, thus, decreasing the need for memory space at the network. We introduce a method to deal with this problem.

3.7.1. Disconnected users – Centralized approach

It is assumed that each of the broadcasted messages has its own lifetime, i.e., the finite time interval in which it is still relevant. Let us denote this interval by *TTL*. It can be also assumed that *TTL* is determined by the source S, and it is measured from the time the message broadcast was initiated. If the broadcast of d_l^S starts at $t = t_l$, then it is assumed that at $t = t_l + TTL$ d_l^S is no longer relevant and if some M_k has not received it yet, then it is not needed to send it this message. Therefore, each of the stations will eliminate these messages at its *TTL* from their memory. Let us denote by T_S the time it takes d_l^S to leave S and reach the last station in the network. Observe that if $TTL < T_S$ then the d_l^S might not reach some connected users, so, it requires that $TTL > T_S$.

We mention that such a solution, based upon Time To Live, is similarly implemented within the well known Internet Protocol (IP).

3.7.2. Disconnected users – Decentralized approach

Let us assume that different stations have different memory size, in this case we may want to let the station store messages up to its memory capacity. If we allow multiple sources broadcast then another problem may arise, now each station must decide which of the messages it discards in case of congestion. Considering the fact that sources can be in different distances from the station, FIFO based policy for discarding messages may not be the one that comply with the life time policy mentioned above.

We assume that a combination of these two approaches is a viable solution for the disconnected users problem.

4. Mobile PIF (MPIF)

An important property for a broadcast protocol is the source ability to know whether and when all broadcast targets have received the message correctly. This is not one of the PI protocol's properties. Thus, a natural expansion is the PIF (Propagation of Information with Feedback) protocol, where the source gets an indication after the broadcast is completed. We would like to have the similar expansion for the MPI protocol, i.e., implement a distributed broadcast protocol for mobile users environment, where the source gets indication after all users have received its message.

4.1. Problem description

With the users model described earlier, we would like to develop a protocol, applied to the users and stations, where the broadcast source is informed after all users receive its message. As mentioned before, broadcasting to users is a more difficult task than broadcasting to stations, since a user might perform a handoff to station which already transmitted the message. Applying PIF on users is even more difficult task as will be explained.

The main principle of PIF consists of the fact that every node marks its first neighbor it gets the message from, as its parent node. When receiving the message or an acknowledgement from all its other neighbors, the node acknowledges back to its parent (acknowledging is done by simply sending the same message back to the parent node).

When the source node gets acknowledgement from each of its neighbors then the source can be sure its message arrived all the other nodes on its connected network, and thus, PIF terminates.

This main principle can not be applied directly for mobile users. A problem may arise when M_k performs a handoff from B_i which has not transmitted the message yet to B_j which already did. This is even more difficult when B_j had also returned an acknowledgement to its parent station. We would like to ensure that the source S gets the termination indication only after the last user has received the message, even those who perform handoff procedure while the broadcast is being done. It is also desirable to keep the messages complexity low, and if there are no users performing handoff procedures while the broadcast is held, then the complexity should match that of the ordinary static PIF.

The previous problem of disconnected users still exists, as in the MPI case, but now it becomes more critical. If a user leaves the ACTIVE subset within the time MPIF is being propagated (i.e., disconnects or moves rapidly forever, so it never completes its last handoff procedure), then the protocol can never be terminated since that user will never return an acknowledgement for the message. To solve this problem, an additional assumption about our model must be made. We assume that all active users, at the time the MPIF was initiated, are in the ACTIVE subset, and we allow a user to disconnect only *after* it sends a proper message to the last station detected it. In cases where this assumption is not so practical, some kind of time-out mechanism must be applied at the stations. The reason for this is that the base network can not distinguish between a disconnected user and a user that delays its acknowledgement because of high error rate in the transmission media. Naturally, a disconnected user can not be detected by any of the stations, and must not be accounted as a target for the broadcast, otherwise the source S might wait forever for its feedback.

4.2. Definitions

Let us redefine some of the messages and data structures, and define some additional ones, to be used in the MPIF protocol.

4.2.1. Messages

- *Ack*(B_j, M_k, l, S), acknowledgement for a message d_l^S that M_k sends B_j, after receiving the message from it.
- *Free*(B_j, B_i, M_k), a message B_j sends B_i, instructing B_i to remove M_k from its connected list.
- *Disconnect*(M_k, B_j), a message M_k sends the current station it is found in its region, indicating that M_k disconnects from the network, and that the last station it received a *Join* message from was B_j.

4.2.2. Data structures

- $Connlist_{B_i}$, list of users identifications connected currently to B_i. This list is stored at B_i, and changes dynamically.
- $Acklist^S_{B_i}$, list of users' identifications which already returned an acknowledgement to B_i, for current broadcast of source station S. This list is also stored at B_i, and changes dynamically.
- $Parent^S_{B_i}$, B_i's internal variable, holds the parent station identification, from which B_i received d_l^S for the first time.
- $Laststation^{M_k}_{B_i}$, the identification of the last station B_j, which M_k was connected to, before it connected to B_i. This information is stored at B_i in order to remove M_k from B_j's *Connlist*. It is kept for each user M_k detected by B_i, until it receives a *Free* message for that user.

4.2.3. Definitions

A generalization and redefinition of a former definition, to be used for the MPIF protocol, is:

Definition 3. We now say that a user M_k has *completed a handoff procedure* with a station B_i at $t = t_2$ if:

- B_i received a *Connect* message from M_k at some $t = t_0$, and
- if at $t = t_0$ station B_i has messages for user M_k, then M_k received these messages at $t = t_1 > t_0$, and sent back an *Ack* message for each of these messages, which were received by B_i at $t = t_2 > t_1$.

4.3. Problem definition

Let us now define the problem more formally. Denote by S a station that initiates a broadcast of a message d_l^S, at time t_l to all users in ACTIVE subset. It is needed to ensure that there exists some finite T_l^{ack}, so that at time $t_l + T_l^{\text{ack}}$, S gets back an indication meaning all users in the subset $\{M_k \mid M_k \in \text{ACTIVE}, \forall t \in [t_l, t_l + T_l^{\text{ack}}]\}$ already received the message d_l^S.

4.4. MPIF – An informal description

We shall now describe a protocol, which we call Mobile PIF (MPIF), to solve this problem. This protocol is based on MPI with the extension of handling acknowledgement messages. As far as the backbone network is concerned, the protocol is similar to the ordinary PIF for fixed networks, i.e., each of the stations marks its first neighbor it gets the message from as its parent in a virtual broadcast tree. When receiving an acknowledgement or another copy of the message itself from all the rest of its neighbors, it sends back an acknowledgement to its parent node. When the source node S receives an acknowledgement from all its immediate neighbors the protocol terminates. But, what about the users?

When a station receives the messages at first, it broadcasts it to all users connected to it, each of which must respond with an acknowledgement. The problem of users performing handoff within $t \in [t_l, t_l + T_l]$ is solved in the following manner. Consider M_k performing handoff from B_i

to B_j. There are four possible cases regarding the state of the broadcast of d_l^S at the time M_k performs this handoff procedure:

1. B_i transmitted d_l^S, but B_j did not transmitted it yet.
2. Neither B_i nor B_j transmitted d_l^S.
3. Both B_i and B_j already transmitted d_l^S.
4. B_i did not transmit d_l^S but B_j already did.

From the above, the fourth case can be the most problematic one, since in the worst case B_j may already returned an acknowledgement to its parent node, and after M_k disconnected B_i, the last could transmit d_l^S, get back all the acknowledgements needed from the rest of the users, and returned an acknowledgement itself to its parent. Thus, the source S might have the indication *before* M_k received d_l^S, violating the desired property.

To avoid this kind of scenario the following is done: when M_k gets the message $Join(M_k, B_j)$ from B_j – message sent when M_k was detected as entering B_j's cell – it replies with $Connect(M_k, B_j, l-1, S)$. Now, if B_j already did broadcast d_l^S, it sends this message to M_k and waits for an acknowledgement. When it received, B_j sends the $Free(B_j, B_i, M_k)$ message to B_i, the last station M_k was connected to, meaning B_j is responsible for M_k from now on. When B_i receives this message it should remove M_k from its connected list, and continue with MPIF protocol regardless M_k. This is how we always keep some station acknowledgement delayed, until M_k acknowledges it back itself.

Finally, it can be shown that S gets back acknowledgement from each of its immediate neighbors only after all users have received the broadcast, even those which move during the broadcast propagation.

If a user M_k is about to disconnect from the network then it sends the message $Disconnect(M_k, B_i)$, where B_i is the last station M_k received a *Join* message from. If B_i is the station receiving this message then it removes M_k from its *Connlist*. Otherwise, the receiving station B_j sends a *Free* message to B_i, instructing it to remove the user from its *Connlist*.

This protocol manages a broadcast of a series of messages identified by $l = 1, 2, \ldots$. However, as the source waits for feedback for d_l^S *before* initiating the broadcast of d_{l+1}^S, one can omit the message sequential number l without loosing any off the protocol properties. Here we decided to keep this identifier for convenience.

4.5. MPIF – The protocol

4.5.1. Assumptions

In addition to the assumptions at the beginning of MPI protocol, just before entering MPIF protocol:

- $Acklist_{B_i}^S = \emptyset$,
- $Connlist_{B_i}$ holds users' identifications connected to B_i,
- $Laststation_{B_i}^{M_k} = Null,\ \forall M_k \in Connlist_{B_i}$,
- $Laststation_{M_k} = B_i,\ \forall M_k \in Connlist_{B_i}$.

It is also assumed that the source station S may initiate the propagation of the message d_l^S just *after* receiving the complete feedback for d_{l-1}^S.

4.5.2. Algorithm running at station B_i

When the main message of the MPIF is received, if it is the first time, update local variables and send it to the immediate neighbors, and transmit it the all users in the cell. If all neighbors returned it, forward it back to the first station you received it from.

For $Start(d_l^{B_i}, B_i)$ or $Msg(d_l^S, B_j)$:
  $Acklist_{B_i}^S = Acklist_{B_i}^S \cup \{B_j\}$,
  if $(h_{B_i}^S < l)$ then
    $h_{B_i}^S = l$,
    $Parent_{B_i}^S = B_j$,
    send $Msg(d_l^S, B_i)$ to all $B_k \in G_{B_i}$,
    send $Trans(B_i, All, d_l^S)$ to all users in B_i's cell,
  if $(Connlist_{B_i} \cup G_{B_i} \subseteq Acklist_{B_i}^S)$ then
    send $Msg(d_l^S, B_i)$ to $Parent_{B_i}^S$,
    $Acklist_{B_i}^S = \emptyset$.

When a user is detected, and it is for the first time, send it a *Join* message:

For $Detect(M_k, B_i, B_j)$:
  $Laststation_{B_i}^{M_k} = B_j$,
  if $(M_k \notin Connlist_{B_i})$ then
    send $Join(M_k, B_i)$ to M_k.

If the user replied *Join* with *Connect*, add its identification to *Connlist*. If the user had already received the currently propagated message, inform its last station to remove it from its *Connlist*. Otherwise, send the message to the user:

For $Connect(M_k, B_i, l, S)$:
  $Connlist_{B_i} = Connlist_{B_i} \cup \{M_k\}$,
  if $(l >= h_{B_i}^S)$ then
    $Acklist_{B_i}^S = Acklist_{B_i}^S \cup \{M_k\}$,
    if $(Laststation_{B_i}^{M_K} \neq Null)$ then
      send $Free(B_i, B_j, M_k)$ to $Laststation_{B_i}^{M_k}$,
      $Laststation_{B_i}^{M_k} = Null$,
  if $(l < h_{B_i}^S)$ then
    send $Trans(B_i, M_k, d_{l+1}^S)$ to M_k.

When a user wants to disconnect, remove it from *Connlist* and check if it was the last user that did not receive the currently propagated message. If so, return back acknowledgement. If this user is still registered in another station's *Connlist*, then inform that station, so it will be removed:

For $Disconnect(M_k, B_j)$:
  if $(M_k \in Connlist_{B_i})$ then

$Connlist_{B_i} = Connlist_{B_i} \setminus \{M_k\}$,
if $(Connlist_{B_i} \cup G_{B_i} \subseteq Acklist^S_{B_i})$ then
send $Msg(d_l^S)$ to $Parent^S_{B_i}$,
$Acklist^S_{B_i} = \emptyset$,
if $(B_i \neq B_j)$ then
send $Free(B_i, B_j, M_k)$ to B_j,
if $(Laststation^{M_k}_{B_i} \neq Null)$ then
send $Free(B_i, Laststation^{M_k}_{B_i}, M_k)$ to $Laststation^{M_k}_{B_i}$,
$Laststation^{M_k}_{B_i} = Null$.

When another station sends a *Free* message, remove this user from *Connlist*, and check whether or not this removal justifies returning an acknowledgement. If this user is known to be registered in another station, forward this *Free* message to that station:

For $Free(B_j, B_i, M_k)$:
if $(M_k \in Connlist_{B_i})$ then
$Connlist_{B_i} = Connlist_{B_i} \setminus \{M_k\}$,
if $(Connlist_{B_i} \cup G_{B_i} \subseteq Acklist^S_{B_i})$ then
send $Msg(d_l^S)$ to $Parent^S_{B_i}$,
$Acklist^S_{B_i} = \emptyset$,
if $(Laststation^{M_k}_{B_i} \neq Null)$ then
send $Free(B_i, Laststation^{M_k}_{B_i}, M_k)$ to $Laststation^{M_k}_{B_i}$,
$Laststation^{M_k}_{B_i} = Null$.

When an *Ack* is returned by a user, that was connected to another station, send that station *Free*:

For $Ack(B_j, M_k, l, S)$:
if $(B_j = B_i)$ then
$Acklist^S_{B_i} = Acklist^S_{B_i} \cup \{M_k\}$,
if $(Laststation^{M_k}_{B_i} \neq Null)$ then
send $Free(B_i, Laststation^{M_k}_{B_i}, M_k)$,
$Laststation^{M_k}_{B_i} = Null$,
if $(Connlist_{B_i} \cup G_{B_i} \subseteq Acklist^S_{B_i})$ then
send $Msg(d_l^S, B_i)$ to $Parent^S_{B_i}$,
$Acklist^S_{B_i} = \emptyset$.

4.5.3. Algorithm running at user M_k
When the user receives a new message, accept it and acknowledge it:

For $Trans(B_i, M_k, d_l^S)$ or $Trans(B_i, All, d_l^S)$:
if $(h^S_{M_k} = l - 1)$ then
accept d_l^S,
$h^S_{M_k} = l$,
send $Ack(B_i, M_k, l, S)$ to B_i.

When asked to be joined by a station, kindly accept the invitation using *Connect*:

For $Join(M_k, B_i)$:
send $Connect(M_k, B_i, h^S_{M_k}, S)$,
$Laststation_{M_k} = B_i$.

Once in a while, or by any other trigger, send *Detect* message, to notify the near stations of your existence in their cell:

send $Detect(M_k, B_i, Laststation_{M_k})$ to local station B_i.

4.6. Properties of MPIF protocol

The properties of the MPIF protocol appear in the following theorems.

4.6.1. Correctness

Theorem 4. Suppose S initiates a propagation of $Msg(d_l^S)$ at time t_l, then

(1) there exists some finite time interval T_l so that *all* base stations B_i have already received $Msg(d_l^S)$ before $t = t_l + T_l$,

(2) if $M_k \in$ ACTIVE then M_k will eventually receive $Msg(d_l^S)$,

(3) M_k receives message d_l^S in the fastest way possible, under the assumptions of the model described above,

(4) if S received an acknowledgement for d_l^S at $t = t_l + t_l^{\text{ack}}$, then all users in ACTIVE at $t_l < t < t_l + t_l^{\text{ack}}$ already received d_l^S,

(5) for any number of users $M_k \in$ ACTIVE, S will eventually receive an acknowledgement for d_l^S.

4.6.2. Message complexity

Let us assume that the backbone communication network, connecting the stations, is fixed and its topology is known to each of the stations. We denote the message complexity cost of sending a message from B_i to B_j by $K^{B_j}_{B_i}$. We denote by K the *maximal* cost of the above. We also recall that Z denotes the number of handoff procedures being held at $t \in [t_l, t_l + T_l]$ by users in the network.

Theorem 5. For a broadcast of a single message d_l^S, the number of messages sent by MPIF protocol is at most $2(|E| + P) + Z(4 + K)$.

We recall that the message complexity of the regular static PIF protocol (when no handoffs are allowed) is $2(|E| + P)$. It can be easily shown by setting $Z = 0$ that, when the users are not performing any handoffs, the MPIF protocol message complexity converges to the ordinary PIF protocol's. Thus, the following is an important reduction.

Reduction 2. If no handoff procedure is performed at $t \in [t_l, t_l + T_l]$, i.e., $Z = 0$, then the message complexity of MPIF is $2(|E| + P)$.

A possible and natural assumption is that there exists a vertex between any two handoff adjacent cells. In this case

$K = 1$, and a single handoff procedure cost is only 5 messages, so the total message complexity is $5Z + 2(|E| + P)$.

Memory space complexity required at a user is $\mathcal{O}(1)$, and $\mathcal{O}(P)$ at a station. That means, the mobile user's instrument needs a fixed amount of memory, for any size of network topology, and no matter how many other mobile subscribers exist. This property, naturally, makes this algorithm applicable even for small mobile instruments.

4.6.3. Using MPIF protocol as a Mobilizer

So far we introduced the Mobile PIF protocol and its properties. We will show now how to use it as a Mobilizer for protocols that are based on broadcast with feedback. The meaning of broadcast based protocol is that some of the stations broadcast messages destined for all users. The users may reply these messages, while the source stations wait for a feedback.

Let A be such a protocol, originally designed for synchronous networks, we wish to execute over a cellular network with fast moving users. The only thing that is needed to be done is to combine A with the MPIF protocol. Now, any station wishes to initiate a broadcast within A, will use MPIF, while encapsulating its message within the data field d of the MPIF message *Msg*. A mobile user, wishing to initiate a broadcast as a part of algorithm A, will instruct its current station to do so. When this station receives a feedback, it will forward it to that user, using MPI protocol, or directly, if the user still remains in the cell.

By the above properties of the MPIF protocol, the message complexity overhead will be only linear (that is, $K + 4$) with the number Z of handoff procedures being held by users during the propagation of the message.

Hence, we developed a general mechanism for turning a distributed protocol designed for a network with static users into a protocol that can operate over a cellular network with mobile users, and the additional message complexity overhead is low compared to the original distributed protocol.

5. Summary

In this paper we introduced two distributed protocols designed for broadcasting over distributed networks with mobile users. The first protocol, Mobile PI, enables propagation of a series of messages that will be eventually received, at FIFO order, by the active users. The second protocol, Mobile PIF, gives the source station an indication after all connected users have received the propagated message correctly. Both protocols cope with unbounded rate of user movements, without a need to complete every handoff procedure. Both additional message complexity overhead over the static PI and PIF protocols is linear with user movements.

Having these properties, the MPI and MPIF protocols can supply an efficient novel way to convert any given synchronous protocol, based on PI or PIF, to a distributed asynchronous one, operating over networks with mobile users moving rapidly. We call this protocol and method the Mobilizer. An important step forward may be developing this Mobilizer mechanism to be able to convert *any* given synchronous protocol to a mobile environment. This can be done by developing a dynamic protocol for unicast in a cellular network containing highly mobile users.

Appendix. Proofs

The following appendix appeared also in [7]. See also [8].

Proof of theorem 1(1). Suppose base station S initiates the protocol lth phase by sending $Msg(d_l^S)$ at time t_l. We recall that the base stations B_i are connected with each other through a fixed network. This network is the same as in the regular static PI model. Thus, the properties of the PI protocol guarantee that there exists a finite time at which the last station has received the propagated message [5]. Let us denote this time by T_{last}, i.e., by the time $t_l + T_{\text{last}}$ *all* base stations B_i in the network have received $Msg(d_l^S)$. □

Proof of theorem 1(2). Consider now a mobile user $M_k \in$ ACTIVE at $t > t_l + T_{\text{last}}$. Denote by B_i the last station M_k completed a handoff procedure with, before $t = t_l$. By the assumption that $M_k \in$ ACTIVE at $t > t_l + T_{\text{last}}$, there are two possible cases: first, M_k did not leave B_i's cell. In this case, by the last theorem B_i received $Msg(d_l^S)$ before $t = t_l + T_{\text{last}}$, and by the MPI protocol it sent it to M_k, which will eventually receive it by the logic link assumption. In the second case, M_k started at least one handoff procedure. In this case, by the same assumption mentioned above, M_k has completed its last handoff procedure before $t = t_l + T_{\text{last}}$. Let us denote by B_j the last station received a *Connect* message from M_k. By the MPI protocol, if B_j received this message after $Msg(d_l^S)$ then as a reply to the *Connect* message, B_j sent M_k the message $Msg(d_l^S)$. Otherwise, if the *Connect* message was received before $Msg(d_l^S)$ then when the last one was received at B_j it was forwarded to its connected users, in particular to M_k. □

Proof of theorem 2(1). Let us recall a property of a static PI: if a series of messages is propagated by a source station S, then these messages are received at the backbone stations in FIFO order. By this property, if M_k remains connected to B_i, then it will receive the messages by FIFO order. Otherwise, when M_k completes a handoff procedure to some station B_j and receives a new message d_l^S, then by MPI $l > h_i^S$ and by the FIFO property at the backbone $l = h_i^S + 1$. Hence, FIFO order is preserved within the logic link to M_k. □

Proof of theorem 2(2). We prove now, that user M_k receives $Msg(d)$ in the shortest path available under the assumptions of our model. By the properties of the backbone model, by the fact that MPI is identical to PI in the backbone part, and by the properties of PI, it is guaranteed that the base stations get $Msg(d)$ in the shortest path possible, i.e., in the

fastest way. By MPI, each station transmits $Msg(d)$ to its connected users, as soon as it receives it for the first time. Hence, if M_k did not performed any handoff procedure, then M_k will receive $Msg(d)$ in the fastest way.

Else, if M_k moves along some path in between cells (stations), then by MPI M_k will receive $Msg(d)$ from the first station B_i, satisfying:

(1) B_i received $Msg(d)$ for the first time at $t = t_1$, and

(2) M_k completed a handoff procedure with B_i at some $t = t_2 > t_1$, and

(3) M_k did not complet any handoff procedure before $t = t_2$ with some station B_j, which already received $Msg(d)$.

So, there was no such earlier time where M_i could receive $Msg(d)$ from some station in our model. □

Proof of theorem 3. Let us denote by Z the number of handoff procedures held or partially held (i.e., never completed) by users $M_k \in$ ACTIVE during $t \in [t_l, t_l + T_l]$. Each of these procedures consisted of 4 messages at the most: *Detect*, *Join*, *Connect* and *Trans*. Multiplying by Z gives the total number of messages, induced by users' movements. The message complexity of a standard PI, the static one, is $2|E|$, where E is the static edges set. We recall that MPI operates like PI on the backbone network, and that the message *Trans* is sent only once towards the P users (if they don't move during the propagation of the information). From the above, the total number of message complexity is $2|E| + P + 4Z$, which is $\mathcal{O}(|E| + P + Z)$. □

Proof of theorem 4(1)–(3). These properties are proved the same way as in the MPI case (see also [5]). □

Proof of theorem 4(4). Let us examine a station B_i which returned an acknowledgement to the station from which it received d_l^S for the first time. By the protocol, B_i will not return $Msg(d_l^S)$ to $p_{B_i}^S$ unless either $M_k \in Acklist_{B_i}^S$ or M_k was removed from $Connlist_{B_i}$. The first condition will be fulfilled if M_k has returned an acknowledgement, thus, guaranteed that it eventually received the message *before* B_i returned an acknowledgement towards the source station S. The second condition will be fulfilled only if B_i has received the message $Free(B_i, B_j, M_k)$ from some station B_j.

It is now left to prove the following lemma:

Lemma 1. If a station B_j sends a message $Free(B_j, B_i, M_k)$ to station B_i, then either user M_k has already received the last message $Msg(d_l^S)$ propagated by S, or $M_k \in Connlist_{B_j}$ and B_j did not return an acknowledgement for the last message.

Proof. Let us examine the possible cases where a station B_j might send a message $Free(B_j, B_i, M_k)$. First case: B_j received $Connect(M_k, B_i, l, s)$, added M_k to $Connlist_{B_j}$ and at that moment $l \geqslant h_{B_i}^S$. If $l > h_{B_i}^S$ then M_k already received the message d_l^S before B_j did, so B_j has not returned an acknowledgement for d_l^S yet. If $l = h_{B_i}^S$ then, either they both received the last message and B_j did or did not returned an acknowledgement for it, or, they both did not received the last message so B_j could not have returned its acknowledgement yet. Second case: B_j received $Free(B_m, B_j, M_k)$ from some B_m. By induction, it means M_k is in some station's *Connlist*. Third case: B_j received $Ack(B_j, M_k, l, S)$. This means M_k has received the last message sent by B_j. Forth case: B_j received $Disconnect(M_k, B_m)$ sent by M_k. In this special case the protocol does not guarantee anything for M_k. □

Thus, we proved that when S gets an acknowledgement for d_l^S, all users in ACTIVE subset already received the message. □

Proof of theorem 4(5). It can be shown that messages sent by or from different users are not effecting each other. So, the proof of this theorem does not change if there is more than a single user $M \in$ ACTIVE in the network. Therefore, it is sufficient to prove lemma 2 below.

Lemma 2. If there is only a single user M in the network, and $M \in$ ACTIVE, then S will eventually receive an acknowledgement.

Proof. Assume M is connected to B_i just before S initiated its propagation of d_l^S, so $M \in Connlist_{B_i}$ and $Laststation_M = B_i$. Let us follow all possible scenario. If M remains in B_i's cell then by theorem 4.2 M will eventually receive d_l^S, and immediately send $Ack(B_i, M, l, S)$. Since $M \in$ ACTIVE, there are two possible cases: either B_i received this *Ack* message, or some station B_j received $Detect(M, B_j, B_i)$ some time later. Then, B_j updates $Laststation_{B_j}^M = B_i$. Now, if M will answer the $Join(M, B_j)$ message by $Connect(M, B_j, l, S)$ then B_j will send $Free(M_k, B_i, B_j)$ to B_i that will cause the last to remove M from $Connlist_{B_i}$, and eventually return an acknowledgement. In any additional partially held handoff procedure, if M changes its $Laststation_M$ from B_p to B_q then it holds at B_q $Laststation_{B_q}^M = B_p$. Due to several consecutive partially held handoff procedures there will be created a chain of pointers at the stations, each pointing the previous station which might contain M in its *Connlist*.

When some time later, M will complete a handoff procedure with some B_h, giving it the information that M already received d_l^S. Then B_h will send a *Free* message to the last station B_g M received the *Join* message from. The last one will forward the *Free* message to its $Laststation_{B_g}^M$, and so on, until the original station M was connected to, just before the propagation started, has received the *Free* message.

Thus, eventually, all stations are being able to return back an acknowledgement and S will get its feedback for d_l^S. □

Proof of theorem 5. For a broadcast of the single message d_l^S, upon each of the backbone edges two messages *Msg* are sent. Towards each of the users a *Trans* message

is sent followed by the reply of *Ack*. When a user performs a handoff procedure 4 additional messages at most, are being exchanged between the moving user and its new station: *Detect*, *Join*, *Connect* and *Trans*. Also a *Free* message, which cost K, might be sent. Therefore, if Z is the total number of users movements between cells, then the total number of messages sent by the protocol is $2(|E| + P) + Z(4 + K)$. □

References

[1] A. Acharaya and B. Badrinath, A framework for delivering multicast messages in networks with mobile hosts, Mobile Networks 1(2) (1996) 199–219.

[2] B. Awerbuch and D. Peleg, Network synchronization with polylogarithmic overhead, in: *31st IEEE Symp. on Foundations of Computer Science* (1990) pp. 514–522.

[3] B. Awerbuch and D. Peleg, Concurrent on-line tracking of mobile users, in: *SIGCOMM*, ACM, Zurich, Switzerland (September 1991) pp. 221–233. Computer Communication Review 21(4) (September 1991).

[4] B. Badrinath, A. Acharaya and T. Imielinski, Designing distributed algorithms for mobile computing networks, Computer Communications 19(4) (April 1996) 309–320.

[5] A. Segall, Distributed network protocols, IEEE Transactions on Information Theory 29(1) (January 1983) 23–35.

[6] A. Segall and B. Awerbuch, A reliable broadcast protocol, IEEE Transactions on Communications 31 (1983) 896–901.

[7] M. Sidi, B. Mizrachi and I. Cidon, PI and PIF based mobilizer, Technical report, EED, Technion, Israel, CC PUB(238) (March 1998).

[8] M. Sidi, B. Mizrachi and I. Cidon, PI and PIF based mobilizer, in: *IEEE Symp. on Computer and Communications*, Athens, Greece (March 1998) pp. 310–314.

Boaz Mizrachi received his B.Sc. and the M.Sc. degrees from the Technion – Israel Institute of Technology, Haifa, Israel, in 1996 and 1998, respectively, in electrical engineering. During 1996–1999 he designed and supervised student's hardware projects at the Technion, aimed to enable video conference calls over ATM and Ethernet. In 1998 he joined the R&D founders of Charlotte's Web Networks, Israel, where he currently works on the development of the next generation Tera-bit router. His research interests are in wireless networks protocols, hardware for video transmission over broadband networks and hardware architectures for Tera-bit routing and switching fabric.
E-mail: boazm@tx.technion.ac.il

Moshe Sidi received the B.Sc., M.Sc. and D.Sc. degrees from the Technion – Israel Institute of Technology, Haifa, Israel, in 1975, 1979 and 1982, respectively, all in electrical engineering. In 1982 he joined the faculty of Electrical Engineering Department at the Technion. During the academic year 1983–1984 he was a Post-Doctoral Associate at the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology, Cambridge, MA. During 1986–1987 he was a visiting scientist at IBM, Thomas J. Watson Research Center, Yorktown Heights, NY. He coauthors the book *Multiple Access Protocols: Performance and Analysis* (Springer Verlag, 1990). He served as the Editor for Communication Networks in the IEEE Transactions on Communications from 1989 until 1993, as the Associate Editor for Communication Networks and Computer Networks in the IEEE Transactions on Information Theory from 1991 until 1994, and as an Editor in the IEEE/ACM Transactions on Networking from 1993 until 1997. Currently he serves as an Editor in the Wireless Networks. His research interests are in wireless networks and multiple access protocols, traffic characterization and guaranteed grade of service in high-speed networks, queueing modeling and performance evaluation of computer communication networks.
E-mail: moshe@ee.technion.ac.il

Israel Cidon received the B.Sc. (summa cum laude) and the D.Sc. degrees from the Technion – Israel Institute of Technology in 1980 and 1984, respectively, both in electrical engineering. From 1984 to 1985 he was with the faculty of the Electrical Engineering Department at the Technion. In 1985 he joined the IBM T.J. Watson Research Center, NY, where he was a Research Staff Member and the manager of the Network Architectures and Algorithms group involved in various broadband networking projects such as the PARIS/Planet Gigabit networking testbeds, the Metaring/Orbit Gigabit LAN and the IBM BroadBand Networking architecture. In 1994 and 1995 he was with Sun Microsystems Labs in Mountains View, CA, as a manager of High-Speed Networking founding various ATM projects including Openet – an open and efficient ATM network control platform. Since 1990 he is with the Department of Electrical Engineering at the Technion. He was a founding editor for the IEEE/ACM Transactions on Networking between 1992 and 1997. Previously he served as the Editor for Network Algorithms for the IEEE Transactions on Communications and as a Guest Editor for Algorithmica. In 1989 and 1993 he received the IBM Outstanding Innovation Award for his work on the PARIS high speed network and topology update algorithms, respectively. His research interests are in networks architecture, distributed network applications and algorithms and mobile networks.
E-mail: cidon@ee.technion.ac.il