

Publish/Subscribe Scheme for Mobile Networks

Emmanuelle Anceaume*

Ajoy K. Datta‡

Maria Gradinariu*

Gwendal Simon† *

* IRISA/INRIA-CNRS, Université Rennes 1, France
{anceaume,mgradina}@irisa.fr

† France Telecom R & D, Issy Moulineaux, France
gwendal.simon@rd.francetelecom.com

‡ Department of Computer Science, University of Nevada Las Vegas
datta@cs.unlv.edu

ABSTRACT

The information dissemination in mobile networks is an important but complex and challenging problem. Designing suitable communication primitives for these systems is critical. One of these primitives is the publish/subscribe paradigm. The publish/subscribe is a strategy to establish communication between the information providers (publishers) and information consumers (subscribers) in a distributed system. Our work focuses on an appropriate distributed infrastructure suitable for a scalable implementation of a publish/subscribe system. We present a formal model which is adapted for the peer-based particular subscription criteria of publish/subscribe systems. Moreover, we propose a general deterministic information diffusion scheme for mobile systems. The three main features of our communication scheme are the following: First, our scheme is well-adapted to scalable systems without compromising any subscription criteria or network reorganization. Second, we maintain the anonymity of the distributed system — in order to maintain the network structure, we need only local information. Third, our solution is fully decentralized and modular, thus making it appropriate for practical implementations.

Keywords

anonymous publish/subscribe, peer-to-peer, mobile systems

1. INTRODUCTION

Many recent distributed systems experience frequent and unpredictable changes of size and topology. In a mobile ad-hoc network, nodes communicate by sending messages over wireless links. Traditional dynamic networks are wired networks. However, both ad-hoc and dynamic networks go through the unpredictable failure and recovery of processors and links. Peer-to-peer systems have received a lot of attention recently. The behavior of these systems is much more

complex than most other distributed systems. For example, many processors may choose to connect to some super-processors (i.e., processors with some important resources) causing a polarization of the network. So, maintaining the knowledge of these super-processors is very critical. But, a loss of connection to a name server or some changes in the environment may make this knowledge invalid, causing the super-processors unreachable by some processors in the network.

In order to handle the dynamic behavior of the networks, classical distributed algorithms have been modified or extended to make them adaptable for the ad-hoc and dynamic networks. The research on ad-hoc networks mainly includes the routing protocols ([8, 9, 11, 16, 20]), wireless channel allocation ([12]), and protocols for broadcasting and multicasting ([10, 19, 21]). The work on dynamic networks focused on the message ordering ([15]) and routing ([1, 4]). Recently, the problems of mutual exclusion and k-mutual exclusion for the ad-hoc networks also have been addressed [22, 23]. The leader election problem for the mobile ad-hoc networks has been discussed in [18]. The resource discovery in networks where the processors do not have a global view of the dynamic of the network topology was presented in [13, 17].

Designing a suitable scheme for information dissemination in mobile systems is an important topic of research. One of the paradigms is the publish/subscribe scheme which is used to establish communication between the information providers and the information consumers [5]. The publishers must guarantee timely delivery of events to all the subscribers who subscribed to the topic of those events. There are two main approaches to the implementation of the publish/subscribe scheme. In the topic-based scheme [2], events are marked based on a fixed set of topics/subjects. In the content-based subscription systems [3, 5], the subscribers can refine their subscription choosing filtering criteria along multiple dimensions without requiring the pre-definition of groups.

A publish/subscribe scheme for the mobile environments was proposed in [14]. The authors presented a centralized solution, then discussed issues for the distributed systems. Both approaches are based on the broker concept. In the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

POMC'02, October 30-31, 2002, Toulouse, France.

Copyright 2002 ACM 1-58113-511-4/02/0010 ...\$5.00

centralized approach, there is only one broker in the system, whereas in the distributed solution, the number of brokers is more than one and each broker is responsible for some subset of subscribers. In order to disseminate information to its subscribers, a broker maintains a spanning tree covering the interested subscribers. Both schemes assume that only the source and displays events are mobile, but the brokers belong to a fixed network.

Our Contributions. We are interested in designing a publish/subscribe scheme which can cope with the anonymity and mobility of both publishers and subscribers, weak-connectivity, and polarization (which are some of the characteristics of peer-to-peer networks). Moreover, we investigate the systems where every node could play a role of both publisher and subscriber. To the best of our knowledge, no such a scheme has been proposed. Our scheme clearly separates the logical organization and maintenance of the network from the physical aspects. We model a dynamic network as a logical multi-layered system not necessarily organized in a hierarchical manner. Each layer is represented by a weakly connected graph. This organization of the processors in virtual multi-layers supports a proper model not only for the network mobility but also for the polarization or weak connection of the network. In our model, different layers may run different algorithms to cater different interests of the subscribers. So, our model supports both the topic-based and content-based publish/subscribe schemes — the former can be modeled by a pair of independent layers, while the latter matches the hierarchical organization of the network. We present a deterministic communication scheme which supports the publish/subscribe paradigm with anonymity and fairness constraints. The proposed scheme uses an underlying logical DAG orientation of the network. We show that the proposed DAG orientation scheme is space optimal under the weakest hypothesis — each node is aware only of its neighbors. Our scheme copes with network partitions in the following sense: If the network gets partitioned, the proposed algorithm will still be able to maintain the communication between the publishers and subscribers in the individual partitions.

Outline of the paper. In Section 2, we present the model of a logically organized multi-layer network. Section 3 describes the underlying DAG orientation scheme used by the publish/subscribe scheme (presented in Section 4). In Section 3, we also prove the optimality of the DAG orientation scheme. We conclude the paper in Section 5.

2. PRELIMINARIES

We consider an asynchronous network subject to topology changes. Processors can leave or join the system. Processors and links can fail temporarily (transient faults) or permanently (crash failures). The processors communicate through reliable channels — messages are not corrupted, not lost, and not delivered more than once.

The network is logically organized as a multi-layer system, each logical layer l being a weakly connected graph, also referred to as the communication graph at Layer l . In order to connect to a particular layer l , the processors execute an underlying connection protocol. A processor p is called

active at a layer l if there exists at least one processor q which is connected at l and aware of p . A logical link between two processors p and q at a layer l could be in one of the following states: *up* — the two processors are aware of each other at l ; *down* — p and q are not aware of each other at l ; *forming* — at least one of the processors has initiated the connection protocol for l . The logical *neighbors* of a processor p at a layer l (denoted by $\mathcal{N}^l(p)$) is the set of processors q such that the logical link (p, q) is up.

Note that a processor i may belong to several layers simultaneously. So, i may have different set of neighbors at different logical layers. For example, the network presented in Figure 2.1 has two logical layers. The links of the two layers are distinguished from one another by using two types of lines. The logical neighbors of Processor 5 in Layer 1 are Processors 2, 3, and 4, and its neighbor at Layer 2 is Processor 1.

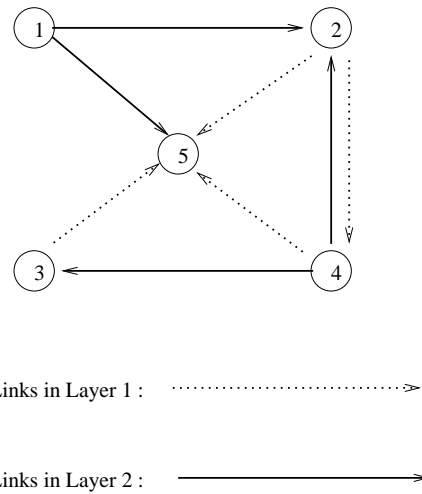


Figure 2.1: Logical Multi-Layer Network

A logical layer l at time t is characterized by:

- the active nodes at l at time t , denoted by $V^l(t)$;
- the logical links up at time t at l , denoted by $E^l(t)$;
- the logical orientation of the communication graph, $G^l(t) = (V^l(t), E^l(t))$, denoted by \mathcal{R}^l .

Note that since we consider dynamic, ad hoc, or peer-to-peer networks in this paper, the communication graph(s) at time t may be different from that at time $t + 1$.

Processors may run different algorithms at different layers. The state of a processor p at layer l at time t is given by the values of p 's variables and the state of its adjacent logical links at time t . The state of the processors at a layer l at time t is denoted by $State(V^l(t))$. The configuration of a layer l at time t consists of the state of the active processors in l at t , the communication graph $(G^l(t) = (V^l(t), E^l(t)))$, and the logical orientation of the communication graph, \mathcal{R}^l . Formally, $c_t^l = (State(V^l(t)), \{(G^l(t) = (V^l(t), E^l(t)), \mathcal{R}^l)\})$.

The configuration of a multi-layer network at time t is represented by the state of all active processors at time t , the set of communication graphs corresponding to all the layers, and the logical orientation for all logical layers $l \in \mathcal{L}$, where \mathcal{L} is a finite set of logical orientations. Formally, $c_t = (c_t^{l_1}, \dots, c_t^{l_k})$ where $c_t^{l_i}$ is the configuration of layer $l_i \in \mathcal{L}$ at time t .

The system transitions correspond to the topological changes or some internal actions executed by some processors. A transition is labeled with the labels of the layers l involved. A system execution is a maximal sequence of transitions.

We now show that if every layer in the system satisfies a property \mathcal{P} and the layers are pairwise independent (the actions executed at a layer do not influence any other layer), then the multi-layer system also satisfies \mathcal{P} .

Definition 1. (pairwise independent layers) Let \mathcal{S} be a system logically organized in a set \mathcal{L} of logical layers. Two layers l_1 and l_2 in \mathcal{L} are called pairwise independent if no action executed by a process at l_1 causes a transition labeled l_2 .

THEOREM 1. *Let \mathcal{S} be a logical multi-layered system with pairwise independent layers, \mathcal{L} the set of layers, and SP^l the specification of the algorithm executed at layer $l \in \mathcal{L}$. \mathcal{S} satisfies $\bigwedge_{l \in \mathcal{L}} SP^l$.*

PROOF. Let e be an execution of \mathcal{S} . Assume that e does not satisfy $\bigwedge_{l \in \mathcal{L}} SP^l$. So, there exists at least one $k \in \mathcal{L}$ such that e does not satisfy SP^k . Let e_k be the projection of e on k . Since e does not satisfy SP^k , then e_k does not satisfy SP^k , which contradicts the hypothesis of the theorem. \square

COROLLARY 1. *Let \mathcal{S} be a logical multi-layer system with pairwise independent layers and SP the specification of the algorithms executed at every layer. Then \mathcal{S} satisfies SP .*

3. MAINTAINING DAG IN MOBILE NETWORKS

The publish/subscribe scheme proposed in Section 4 assumes the underlying logical communication graph to be a DAG. Thus our scheme must maintain the acyclic property of the communication graphs of all layers at all times. In Section 3.2, we first define the logical orientation of the processors such that the communication graphs are acyclic. Then we present an algorithm to maintain the acyclicity when new processors are added to the system. The publish/subscribe scheme is based on the edge reversal concept [6, 7], which can be informally described as follows: Only the sink processors are privileged to execute some task. After a sink processor executes its task, it reverses the direction of all its incident edges. In our scheme, the task to be executed is the information diffusion — a processor is allowed to diffuse information if and only if it is a sink. In Section 3.3, we present a logical re-orientation scheme satisfying the fairness of the processors involved in the diffusion process. We also show that this orientation scheme is space

optimal. In both algorithms (in Sections 3.2 and 3.3), processors use the set of their neighbors as an input. An outline of a neighborhood maintenance protocol suitable for mobile systems is described in Section 3.1.

3.1 Neighborhood Maintenance in Mobile Networks

In this section, we present the primitives executed by the processors to join or leave a network.

Definition 2. (Connection point) Let G^l be the graph representing layer l of a distributed system. Assume that processor i is trying to join the network at layer l by connecting itself to Processor j . Then, j is called a connection point and the edge connecting i to j is referred as a connection edge.

The connection points must be chosen to avoid the critical point creation. A point is called critical if its failure disconnects¹ the network. We now informally describe the idea of the connection primitives. We assume that processors maintain a list of f connection points. The parameter f is application dependent and may be related with the number $(f - 1)$ of failures or departures which can be tolerated. A processor to join the network sends a special signal (*request_link_up*) to all its connection points, and sets the set of all the links to be formed as “forming”. A processor p upon receiving a *request_link_up* signal responds by *link_up_OK* if it accepts the link or by *link_up_KO* otherwise. If p receives *link_up_OK* from a connection point q , then it adds q to its neighbors list $\mathcal{N}(p)$. A processor p to disconnect itself from the network sends a message *request_link_down* to all the processors in its neighbors list $\mathcal{N}(p)$. A processor q receiving a *request_link_down* from a neighbor p resets $\mathcal{N}(q)$ to $\mathcal{N}(q) \setminus \{p\}$ and sends a signal *link_down_OK* to p .

The above connection/disconnection scheme can be easily extended to a multi-layered network. For each layer l , a processor maintains a list of neighbors \mathcal{N}^l .

3.2 Orientation of Communication Graphs

In this section, we define the logical orientation on edges to maintain a DAG. Then we present an algorithm to maintain this orientation when new processors are added. Every processor maintains two variables: an identifier (*lid*) and an integer variable ($val \in \{0, 1, 2\}$). We assume that the processor identifiers are unique in their neighborhood. The variable *val* will be used later in Section 3.3 to re-orient links. We will prove (in Section 3.3) that the state space (2) of *val* matches the lower bound to maintain a fair execution of the edge reversal scheme.

Definition 3. The relation \prec is defined as $x \prec y$ iff $0 < (y - x) < 2$ or $(x = 2 \wedge y = 0)$.

Definition 4. Let $G(V, E)$ be a communication graph. The logical orientation of the edges \rightarrow is defined as follows: the

¹A network is disconnected when there exist some nodes p_i and p_j such that there is no path between them.

edge (p, q) is logically oriented from q to p ($q \rightarrow p$) iff $(val_p < val_q)$ or $(val_p = val_q \wedge lid_p < lid_q)$.

LEMMA 1. Let $G(V, E)$ be a communication graph and \rightarrow a logical orientation of the edges of G . G with the orientation \rightarrow is acyclic.

PROOF. Assume that the graph has a cycle, namely $p_1 \rightarrow \dots \rightarrow p_s \rightarrow p_1$. By Definition 4, that $(val_{p1} < val_{p1})$ or $(lid_{p1} < lid_{p1})$, which is impossible. \square

The orientation algorithm, LLO, is shown as Table 1. A new processor p joins the system by running the neighborhood algorithm discussed in Section 3.1. Processor p then orients its adjacent links in the following manner: If all its adjacent (connected) processors have the same value of val , then p orients all its adjacent edges outward (Rule \mathcal{R}_1) (see Figure 3.2 (a)). Otherwise (if not all neighbors have the same value of val), p orients in a way to maintain the acyclicity of the graph (Rules \mathcal{R}_2 and \mathcal{R}_3). If the set of values of the adjacent processors is $\{0, 1\}$, $\{1, 2\}$, or $\{0, 2\}$, then the new processor p executes Rule \mathcal{R}_2 and sets val_p to the minimal value in the set (see Figure 3.2 (b)). The situation where the adjacent processors of a new processor p have all three possible values of val (i.e., 0, 1, and 2), Rule \mathcal{R}_3 is executed by p . Note that Rule \mathcal{R}_3 may introduce cycles if there are at least two adjacent processors j and k such that $val_j = 1$, $val_k = 2$, and $j \in \mathcal{N}(k)$. We assume that the underlying neighborhood maintenance protocol ensures the following: (1) $\forall (j, k) \in \mathcal{N}(i)$, $|val_j - val_k| \leq 1$ or (2) $\forall (j, k) \in \mathcal{N}(i)$ such that $val_j = 1$ and $val_k = 2$, $j \notin \mathcal{N}(k)$. We will discuss another alternate strategy later.

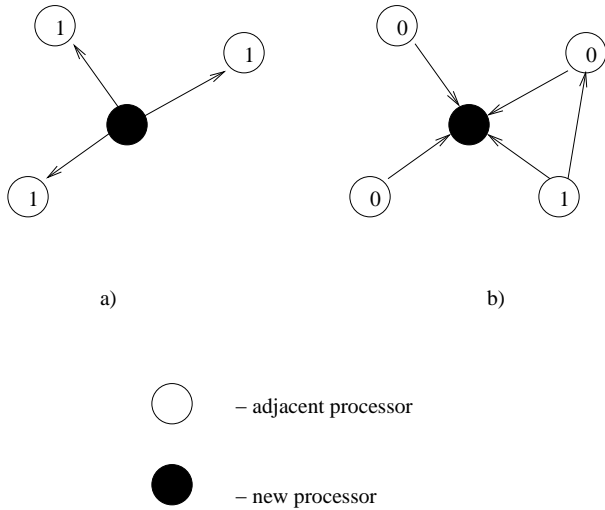


Figure 3.2: Logical link orientation after a new processor joins the system.

LEMMA 2. Let $G(t)$ be the communication graph at time t . Assume that $G(t)$ is acyclic. Let i be a new processor joining $G(t)$ and $G(t+1)$ the communication graph after i joined the system by executing LLO (Table 1). Then $G(t+1)$ is acyclic.

Parameters :	
$\mathcal{N}(i)$:	set of neighbors computed by the neighborhood maintenance protocol;
$val_i \in \{0, 1, 2\}$:	integer, 0 by default;
lid_i :	integer, the value of the local identifier of i after joining the system;
Functions:	
$IdS()$	$= [\min(\mathcal{N}(i)), \max(\mathcal{N}(i))] \cap IN \setminus \mathcal{N}(i)$
$Id_{av}()$	$= \begin{cases} true, & IdS() \neq \emptyset \\ false, & IdS() = \emptyset \end{cases}$
Macro:	
$Choose_{id}$: returns the first identifier available in $\mathcal{N}(i)$ or the maximum of the identifiers plus 1 when there is no identifier available in the list of neighbors. Formally:
$Choose_{id}(i)$	$= \begin{cases} \min(IdS) & , Id_{av}() = true \\ \max(\mathcal{N}(i)) + 1 & , Id_{av}() = false \end{cases}$
Actions :	
\mathcal{R}_1	: if $(\forall (j, k) \in \mathcal{N}(i), val_j = val_k)$ $val_i = (val_j + 1) \bmod 3;$ $lid_i = Choose_{id}(i)$
\mathcal{R}_2	: if $(\exists (j, k) \in \mathcal{N}(i), val_j \neq val_k) \wedge (\bigcup_{j \in \mathcal{N}(i)} \{val_j\} \neq \{0, 1, 2\})$ $val_i = \min_{j \in \mathcal{N}(i)} val_j;$ $lid_i = Choose_{id}(i)$
\mathcal{R}_3	: if $(\exists (j, k) \in \mathcal{N}(i), val_j \neq val_k) \wedge (\bigcup_{j \in \mathcal{N}(i)} \{val_j\} = \{0, 1, 2\})$ $val_i = 0;$ $lid_i = \max_{j \in \mathcal{N}(i), val_j = 0} \{lid_j\} + 1$

Table 1: LLO - Logical Link Orientation Algorithm (Processor i)

PROOF. Assume that $G(t+1)$ contains a cycle. Since $G(t)$ was acyclic, the cycle must contain i . Hence, there exist two edges adjacent to i — one oriented to i (edge (i, j)) and the other towards a neighbor k (edge (i, k)) — and all three processors i, j, k are included in the cycle, which is impossible by LLO. \square

NOTE 1. When a processor leaves the system, it does not affect the acyclicity of the communication graph.

NOTE 2. One drawback of LLO is the two assumptions we made earlier on the values of val of the neighboring processors. It is possible to remove this restriction in the following way: A new processor p , in order to orient its edge (p, q) for an adjacent processor q waits until q becomes a sink. Until then, the edge (p, q) is maintained in the state forming by both p and q . Although this solution removes the earlier restrictions, but it has another problem. A new processor in this new scheme needs to wait for every neighbor to become a sink. This waiting time depends on the dynamic behavior of the network.

3.3 Space Optimal Link Re-orientation Scheme

In this section, we first present a link re-orientation scheme when the network is logically organized in one virtual layer.

Parameters :

$\mathcal{N}(i)$: set of neighbors computed by the neighborhood maintenance protocol;
 $val_i \in \{0, 1, 2\}$: integer, 0 by default;

Function :

$sink(i) : \forall j \in \mathcal{N}(i), j \rightarrow i$

Actions :

$\mathcal{R}_1 : \text{if } sink(i) \wedge (\forall j \in \mathcal{N}(i), val_j = val_i)$
 $val_i = (val_i + 1) \bmod 3;$
 $\mathcal{R}_2 : \text{if } sink(i) \wedge (\exists j \in \mathcal{N}(i), val_i < val_j)$
 $val_i = \max_{j \in \mathcal{N}(i)}(val_j);$

Table 2: LLRO - Logical Link Re-orientation Scheme in a One-Layer Network (Processor i)

Then we generalize our scheme to multi-layer networks. Our orientation scheme is based on an edge reversal algorithm. Only the sink processor is privileged to execute a task (publish an event). After the sink processor finished its task, its adjacent edges are re-oriented to ensure the fairness among all neighboring processors.

3.3.1 Logical Link Reorientation Scheme in a One Layer Network

Definition 5. (Sink node) Let $G(t)$ be the communication graph at time t . A processor p is called a sink node if for all neighbors q of p , $q \rightarrow p$.

The idea of the re-orientation algorithm is simple. By Definition 4, the network oriented according to the relation \rightarrow contains at least one sink processor. These sink processors are enabled to execute a rule of LLRO (Table 2). After they execute a rule, they re-orient their adjacent edges toward their neighbors. So, the new orientation remains acyclic. The edge reversal is implemented by changing val to the maximal value in the neighborhood. If all neighbors of the sink processor have the same value v of val , then the sink sets its val to $(v + 1) \bmod 3$ (see Rule \mathcal{R}_1). If the sink processor i has at least one neighbor j such that $val_i < val_j$, then i sets its val to the maximal value of val of all the neighbors of i by executing \mathcal{R}_2 .

LEMMA 3. *Let $G(t)$ be the communication graph at time t and $G(t + 1)$ the communication graph after the execution of LLRO by a sink processor p . Then $G(t + 1)$ is acyclic according to the relation \rightarrow .*

PROOF. Assume the contradictory, i.e., $G(t + 1)$ contains a cycle. Since $G(t)$ was initially acyclic, then this cycle must contain p . Let $p \rightarrow p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_s \rightarrow p$ be the cycle. Since after the execution of the algorithm by p , the edges adjacent to p are oriented towards its neighbors, the only way this cycle can be created is that p_s executed the algorithm in parallel with p and also reversed its adjacent edges. However, that is impossible since the orientation relation is a total order. \square

LEMMA 4. *Let e be an execution of LLRO algorithm. Every processor in the system executes its algorithm infinitely often even in the presence of change of topology (i.e., the addition and deletion of processors).*

PROOF. We show that in a finite number of computation steps, a processor gets its adjacent edges oriented toward it, thus will execute LLRO.

Assume that a processor p never executes LLRO. So, there must exist a neighbor q of p such that the logical link (p, q) remains oriented from p to q forever. This may happen due to one the following reasons: (a) The network contains a cycle $p \rightarrow p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_s \rightarrow p$, which is impossible by Lemma 1. (b) The addition of a new processor in the network created a cycle, which is also impossible by Lemma 2. \square

NOTE 3. *Lemma 4 proves two important properties of LLRO — starvation freedom (i.e., every processor eventually executes its actions) and liveness (i.e., a processor executes its actions infinitely often).*

NOTE 4. *Note that since the underlying neighborhood maintenance protocol updates the list of neighbors, even if the network becomes partitioned, the edge reversal scheme works with no additional cost in every individual partition.*

3.3.2 Optimality of the Logical Link Reorientation Scheme

In this section, we will show that the edge reversal scheme (Table 2) is state optimal. We show that the minimal number of states in order to construct an edge reversal scheme which ensures the starvation freedom is 2. Note that the number of states does not include the variable lid because it is well-known that using identifiers is the only way to break the symmetry of a deterministic system.

LEMMA 5. *LLRO is state optimal.*

PROOF. We prove this lemma by contradiction. Assume that we can design a starvation-free edge reversal scheme using only two states. Assume that the two states are 0 and 1. To arrive at the contradiction, we will construct an execution where one of the processors never executes any action.

Assume that the system contains two processors p_1 and p_2 such that the local identifier of p_1 is less than the local identifier of p_2 , and the states of p_1 and p_2 are 0 and 1, respectively. In this configuration, the edge between them is oriented from p_2 to p_1 . p_1 executes Rule \mathcal{R}_2 and changes its state to 1. Now, in this new configuration, the edge is still oriented from p_2 to p_1 (the orientation in this case is decided by the value of the identifiers). Therefore, p_1 executes \mathcal{R}_1 and changes its state to 0. This configuration is the same as the first configuration. Thus, we obtain an execution where p_2 starves (i.e., it never executes any action of LLRO). \square

<p>Parameters :</p> <p>\mathcal{L}: the set of layers i belongs to;</p> <p>$Val = \{val_i^l \in \{0, 1, 2\} \mid l \in \mathcal{L}\}$: set of integers, 0 by default;</p> <p>$\mathcal{N} = \{\mathcal{N}^l(i) \mid l \in \mathcal{L}\}$: the set of neighbors of i for all levels in \mathcal{L};</p> <p>Relation:</p> <p>$\forall l \in \mathcal{L}, \rightarrow^l$ is relation \rightarrow with respect to val^l and identifiers;</p> <p>Function :</p> <p>$sink^l(i) : \forall j \in \mathcal{N}^l(i), j \rightarrow^l i$</p> <p>Action :</p> <p>$\mathcal{R} : \text{if } \exists l \in \mathcal{L}, sink^l(i)$ Execute LLRO with parameters val_i^l and $\mathcal{N}^l(i)$</p>

Table 3: LLRM - Logical Link Reorientation Scheme in a Multi-layer Network (Processor i)

3.3.3 Logical Link Reorientation Scheme in a Multi-Layer Network

In this section, we generalize the scheme for networks where processors are organized in multiple layers, each layer characterized by an oriented communication graph. Every processor maintains a separate list of neighbors for every layer it belongs to.

LEMMA 6. *Let \mathcal{P} be a predicate satisfied by the computations of a system executing LLRO algorithm (Table 2) then \mathcal{P} is also satisfied by the computations of LLRM (Table 3).*

PROOF. Follows directly from Theorem 1. \square

4. ANONYMOUS PUBLISH / SUBSCRIBE SCHEME

In this section, we use LLO algorithm (Table 1) and LLRO algorithm (Table 2) to design a new deterministic publish/subscribe scheme which preserves the anonymity of the publishers. We discuss both single topic and multiple topic algorithms below. Note that a processor can be both a publisher and a subscriber for some news. We now define the properties a deterministic publish/subscribe scheme must satisfy:

- **Safety:** If a publisher generates a news m on a topic t , then m is eventually delivered.
- **Agreement:** If a publisher generates the news m on the topic t , then every subscriber for the topic t receives m .
- **Liveness:** Every publisher delivers infinitely often.
- **Anonymity:** The information is diffused in the network in an anonymous way.

4.1 Anonymous Publish / Subscribe One Topic Algorithm

<p>Volatile Variables:</p> <p>$Input_Messages_i$: list of messages received from i's neighbors, initially empty;</p> <p>$Local_Messages_i$: list of messages generated locally;</p> <p>Primitives:</p> <p>$receive(Input_Messages_i)$: returns the list of messages received from the input buffer;</p> <p>$collect(Local_Messages_i)$: reads the local buffer and returns messages in $Local_Messages_i$. The invocation of this primitive empties the local buffer;</p> <p>$send(New_Messages_i)$: sends the messages from $New_Messages_i$;</p> <p>Action:</p> <p>$\mathcal{R}1 : \text{if } sink(i)$ $receive(Input_Messages_i);$ $collect(Local_Messages_i);$ $send(Input_Messages_i \cup Local_Messages_i);$ execute LLRO (Table 2)</p>

Table 4: Publish/subscribe Single Topic Scheme (Processor i)

In a publish/subscribe system, subscribers subscribe to some specific categories of events. They subscribe to a single category or multiple categories in single topic or multiple topic systems, respectively. The subscriptions are modeled using links in the communication graph representing the publish/subscribe system. In Section 3, we implemented the communication graphs as DAGs. Moreover, we also have shown that the DAGs will remain as DAGs forever in spite of any topological changes and any actions by any processor. That guarantees the presence of at least one sink at any time. The sink processors behave as privileged processors to send or forward any information regarding any event of the publish/subscribe system. They are allowed to send messages that are generated locally and received from the neighbors. After sending the messages, they re-orient their adjacent edges to give a chance to their neighbors to do the same. A non-privileged (i.e., not a sink) processor can receive some messages, but is not allowed to forward the information until it becomes a sink. So, only the sink processors are points of diffusion, thus avoiding the network flooding. The publish/subscribe scheme (shown as Table 4) uses two buffers and two communication primitives, **send()** and **receive()** to disseminate all the events to the interested subscribers using the underlying oriented communication graph. The two buffers are called input and local buffers. In Table 4, $Input_Messages$ and $Local_Messages$ refer to the messages stored in the input and local buffers, respectively. The input buffer is used to record all the messages received from the neighbors, and the local buffer saves the messages generated locally. The **receive()** primitive when invoked reads the input buffer and returns the messages collected between the previous and current invocation of **receive()**. Processors write their own (i.e., generated locally) messages in their local buffer. A processor by calling the **send()** primitive sends to all its neighbors the messages returned by the receive primitive and the ones produced locally (stored in the local buffer).

LEMMA 7 (SAFETY). *Let p be a publisher generating a*

message m . m is eventually delivered.

PROOF. Follows from Lemma 4. The processor representing the publisher eventually becomes a sink. Hence, it delivers its messages at that point. \square

LEMMA 8 (AGREEMENT). *Let p be a publisher generating a message m . m is received by every active subscriber.*

PROOF. The proof follows from Lemma 4 and the assumption that the sink processors are not allowed to leave the system. \square

LEMMA 9 (LIVENESS). *A processor p delivers infinitely often.*

PROOF. The proof directly follows from Lemma 4. \square

NOTE 5 (ANONYMITY). *The multi topic publish/subscriber algorithm (Table 4) ensures the diffusion of the information in an anonymous way. The sender of a message may not be the publisher, it could simply be a forwarding subscriber.*

4.2 Anonymous Publish / Subscribe Multiple Topic algorithm

In this section, we generalize the algorithm presented in Table 4 for the multiple topic publish/subscribe system. Every processor maintains a list \mathcal{L} of topics to which it is a member of. Now, the processors maintain l sets of the two buffers (input and local). $send()$, $receive$, and $collect$ primitives now refer to specific layers. The messages are also maintained in l sets of lists $Input_Messages^l$ and $Local_Messages^l$. Note that all the properties proved for the algorithm presented in Table 4 are also valid for the multiple topic publish/subscriber algorithm (Table 5).

5. CONCLUSIONS

We studied the anonymous publisher/subscriber communication scheme in mobile networks. We presented a model for the organization of a mobile system. An optimal space information dissemination scheme is introduced. We also demonstrated that the logical multi-layer organization is the most inclusive approach to modeling the mobile systems. Moreover, we formally stated the relation between the properties of a logically organized multi-layer system function and that of its sub-layers. We proposed a completely decentralized, deterministic publish/subscribe scheme which preserves the anonymity of the publishers and copes with the dynamic systems which go through topological changes including network partitioning.

6. REFERENCES

[1] Y. Afek, E. Gafni, and A. Rosen. The slide mechanism with applications in dynamic networks. *Proceedings of the Eleventh Annual ACM Symposium on Principles of Distributed Computing (PODC'92)*, pages 35–46, 1992.

Volatile Variables :

$Input_Messages_i^l$: list of messages received from i 's neighbors at layer l , initially empty;
 $Local_Messages_i^l$: list of messages generated locally at layer l ;

Primitives:

$receive(Input_Messages_i)$: returns the list of messages received from the input buffer;
 $collect(Local_Messages_i)$: reads the local buffer and returns messages in $Local_Messages_i$. The invocation of this primitive empties the local buffer for the layer l ;
 $send(New_Messages_i)$: sends the messages from $New_Messages_i$;

Action:

$R1$: *if* $sink^l(i)$
 $receive(Input_Messages_i^l)$;
 $collect(Local_Messages_i^l)$;
 $send(Input_Messages_i^l \cup Local_Messages_i^l)$;
 execute LLRM (Table 3)

Table 5: Publish/Subscribe Multiple Topic Scheme (Processor i)

- [2] K. M. Aguilera and R. Strom. Efficient atomic broadcast using deterministic merge. In *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing (PODC'99)*, 1999.
- [3] M. Aguilera, R. Strom, D. Sturman, M. Astley, and T. Chandra. Matching events in a content-based subscription system. In *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing (PODC'99)*, pages 53–61, 1999.
- [4] B. Awerbuch, Y. Masour, and N. Shavit. Polynomial end-to-end communication. *Proceedings of the Thirty-th Annual Symposium Foundation of Computer Science (FOCS'89)*, pages 358–363, 1989.
- [5] G. Banavar, T. Chandra, B. Mukherjee, and J. Nagarajarao. An efficient protocol for content based publish subscribe systems. In *Proceedings of the 19th International Conference on Distributed Computing Systems (ICDCS'99)*, 1999.
- [6] V. Barbosa and E. Gafni. Concurrency in heavily loaded neighborhood-constrained systems. *ACM Transactions on Programming Languages and Systems*, 11(4):562–584, 1989.
- [7] M. Chandy and J. Misra. The drinking philosophers problem. *ACM Transactions on Programming Languages and Systems*, 4(6):632–646, october 1984.
- [8] M. Corson and A. Ephremides. A distributed routing algorithm for mobile wireless networks. *ACM Journal of Wireless Networks*, 1(1):61–81, 1997.
- [9] R. Dube, C. D. Rais, K. Wang, and T. S.K. Signal stability based adaptive routing (ssa) for ad-hoc mobile networks. *IEEE Personal communications*, pages 36–45, february 1997.

- [10] A. Ephremides and T. Truong. Scheduling broadcasts in multihop radio networks. *IEEE Transactions on Communications*, 4(38):456–460, 1990.
- [11] E. Gafni and D. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, C-29(1):11–18, 1981.
- [12] M. Gerla and T. Tsai. Multicluster, mobile, multimedia radio networks. *Wireless Networks*, pages 255–265, 1995.
- [13] M. Harchol-Balter, T. Leighton, and D. Lewin. Resource discovery in distributed networks. *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing (PODC'99)*, pages 229–237, 1999.
- [14] Y. Huang and H. Garcia-Molina. Publish/subscribe in a mobile environment. *ACM Int. Workshop on Data Engineering for wireless and mobile access (MOBIDE'01)*, pages 27–34, 2001.
- [15] I. Keidar and D. Dolev. Efficient message ordering in dynamic networks. *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing (PODC'96)*, pages 68–76, 1996.
- [16] Y. Ko and V. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *Proc. of 4th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 66–75, 1998.
- [17] S. Kutten, D. Peleg, and U. Vishkin. Deterministic resource discovery in distributed networks. *The Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '01)*, pages 77–83, 2001.
- [18] N. Malpani, J. Welch, and N. H. Vaidya. Leader election algorithms for mobile ad hoc networks. *Proceedings of fourth international workshop on discrete algorithms and methods for mobile computing and communications*, pages 96–103, 2000.
- [19] E. Pagani and G. Rossi. Reliable broadcast in mobile multihop packet networks. *In Proc. of 3rd ACM/IEEE International Conference on Mobile Computing and Networking*, pages 34–42, 1997.
- [20] C. E. Perkins and R. E. M. Ad-hoc on demand distance vector routing. *In Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.
- [21] E. Royer and C. Perkins. Multicast operation of the ad-hoc on demand distance vector routing protocol. *In Proc. of 5th ACM/IEEE International on Mobile Computing and Networking*, pages 207–218, 1999.
- [22] J. Walter, G. Cao, and M. Mohanty. A k-mutual exclusion algorithm for ad-hoc mobile networks. *ACM Int. Workshop of principals of mobile computing (POMC'01)*, 2001.
- [23] J. Walter, J. Welch, and N. Vaidya. A mutual exclusion algorithm for ad-hoc mobile networks. *ACM and Baltzer Wireless Networks journal, special issues on DialM papers*, 2001.