Faster Communication in Known Topology Radio Networks

EXTENDED ABSTRACT

Leszek Gasieniec
Dept. of Computer Science
The University of Liverpool
Liverpool L69 7ZF, UK
leszek@csc.liv.ac.uk

David Peleg *
Dept. of Computer Science
The Weizmann Institute
Rehovot 76100, Israel
david.peleg@weizmann.ac.il

Qin Xin
Dept. of Computer Science
The University of Liverpool
Liverpool L69 7ZF, UK
qinxin@csc.liv.ac.uk

ABSTRACT

This paper concerns the communication primitives of broadcasting (one-to-all communication) and gossiping (all-to-all communication) in radio networks with known topology, i.e., where for each primitive the schedule of transmissions is precomputed based on full knowledge about the size and the topology of the network.

The first part of the paper examines the two communication primitives in general graphs. In particular, it proposes a new (efficiently computable) deterministic schedule that uses $O(D + \Delta \log n)$ time units to complete the gossiping task in any radio network with size n, diameter D and max-degree Δ. Our new schedule improves and simplifies the currently best known gossiping schedule, requiring time $O(D + \sqrt[i+2]{D}\Delta \log^{i+1} n)$, for any network with the diameter $D = \Omega(\log^{i+4} n)$, where i is an arbitrary integer constant $i \geq 0$, see [17]. For the broadcast task we deliver two new results: a deterministic efficient algorithm for computing a radio schedule of length $D + O(\log^3 n)$, and a randomized algorithm for computing a radio schedule of length $D + O(\log^2 n)$. These results improve on the best currently known $D + O(\log^4 n)$ time schedule due to Elkin and Kortsarz [12].

The second part of the paper focuses on radio communication in planar graphs, devising a new broadcasting schedule using fewer than $3D$ time slots. This result improves, for small values of D, on currently best known $D + O(\log^3 n)$ time schedule proposed by Elkin and Kortsarz in [12]. Our new algorithm should be also seen as the separation result between the planar and the general graphs with a small diameter due to the polylogarithmic inapproximability result in general graphs due to Elkin and Kortsarz, see [11].

*Supported in part by a grant from the Israel Science Foundation and by the Royal Academy of Engineering.

PODC'05, July 17–20, 2005, Las Vegas, Nevada, USA.

Categories & Subject Descriptors:
F.2.2[Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems;
G.2.2[Discrete Mathematics]: Graph Theory;
General Terms: Algorithms, Theory.
Keywords: broadcasting, gossiping, algorithms, radio networks, planar graphs.

1. INTRODUCTION

1.1 Background

The two classical problems of information dissemination in computer networks are the *broadcasting* problem and the *gossiping* problem. The broadcasting problem requires distributing a particular message from a distinguished *source* node to all other nodes in the network. In the gossiping problem, each node v in the network initially holds a message m_v, and it is required to distribute all messages m_v to all nodes in the network. In both problems, the efficiency criterion is very often to minimize the time needed to complete the task.

The paper concerns the following model of a radio network. A network is an undirected connected graph $G = (V, E)$, where V represents the set of nodes of the network and E contains unordered pairs of distinct nodes, such that $(v, w) \in E$ iff the transmissions of node v can directly reach node w and vice versa (the reachability of transmissions is assumed to be a symmetric relation). In this case, we say that the nodes v and w are *neighbours* in G. One of the properties of radio network is that a message transmitted by a node is always sent to all its neighbors.

The number of neighbours of a node w is called its *degree*, and the maximum degree of any node in the network is called the *max-degree* of the network and is denoted by Δ. The *size of the network* is the number of nodes $n = |V|$. The communication is synchronous and consists of a sequence of communication steps. During each step, each node v either transmits or listens. If v transmits, then the transmitted message reaches each of its neighbours by the end of this step. However, a node w adjacent to v successfully receives this message iff in this step w is listening and v is the only transmitting node among w's neighbors. If node w is adjacent to a transmitting node but is not listening, or is adjacent to more than one transmitting node, then a *collision* occurs and w does not retrieve any message in this step.

The running time of any communication schedule is determined by the number of the time steps required to complete the communication task. That is, we do not account for any internal computation within individual nodes. Another abstraction of the model is that no limit is placed on the length of a message which one node can transmit in one step. This assumption is used here in the context of the gossiping problem where it is assumed that if a node transmits in the current step, then it transmits all the information it currently has.

In this paper we focus on gossiping algorithms that rely on using complete information about the network topology. This type of topology-wise communication algorithms are useful in radio networks that have a reasonably stable topology/infrastructure. As long as no changes occur in the network topology during the execution of the algorithm, the tasks of broadcasting and gossiping will be completed successfully. Note also that our main goal was the design of time efficient communication procedures. However it would not be difficult to increase the level of fault-tolerance in our algorithm at the expense of some extra time consumption.

1.2 Communication in radio networks with known topology

The work on communication in known topology radio networks was initiated in the context of the broadcasting problem. In [13], Gaber and Mansour prove that the broadcasting task can be completed in time $O(D\log^2 n)$. An $\Omega(\log^2 n)$ time lower bound was proved for the family of graphs of radius 2, see [2]. While it was known for quite a while [3] that for every n-vertex radio network of diameter D there exists a deterministic broadcasting schedule of length $O(D\log n+\log^2 n)$, an appropriate efficient construction for such a schedule was proposed only very recently in [19]. More recently, an efficient deterministic construction for a broadcasting schedule of length $D+O(\log^4 n)$ was proposed by Elkin and Kortsarz [12]. They have also presented an efficient deterministic construction for a broadcasting schedule of length $D+O(\log^3 n)$ for planar graphs.

Efficient radio broadcasting algorithms for various particular types of network topologies can be found in Diks *et al.* [10]. For general networks, however, it is known that the computation of an optimal (radio) broadcast schedule is NP-hard, even if the underlying graph is embedded in the plane [4, 21].

Radio gossiping in networks with known topology was first studied in the context of radio communication with messages of limited size, see [6]. In this model the authors proposed several optimal or close to optimal $O(n)$-time gossiping procedures for various standard network topologies, including lines, rings, stars and free trees. They also proved that there exists a radio network topology in which the gossiping (with unit size messages) requires $\Omega(n\log n)$ time. The first work on radio gossiping in known topology networks with arbitrarily large messages is [17], where the authors propose several optimal gossiping schedules for a wide range of radio network topologies.

1.3 Our results

In the first part of the paper we examine the communication primitives in general graphs. In particular, we propose a new efficiently computable deterministic schedule that uses $O(D+\Delta\log n)$ time units to complete the gossiping task in any radio network with size n, diameter D and max-degree Δ. Our new schedule improves and simplifies the currently best known gossiping schedule requiring time $O(D+\sqrt[i+2]{D}\Delta\log^{i+1} n)$, for any network with the diameter $D=\Omega(\log^{i+4} n)$, where i is an arbitrary integer constant $i\geq 0$, see [17]. For the broadcast task we show two new results: a deterministic efficient algorithm for computing a radio schedule of length $D+O(\log^3 n)$, and a randomized algorithm for computing a radio schedule of length $D+O(\log^2 n)$. These results improve on the best currently known $D+O(\log^4 n)$ time schedule due to [12].

In the second part of the paper we focus on radio communication in planar graphs, devising a new broadcasting schedule using fewer than $3D$ time slots. This result improves, for small values of D, on currently best known $D+O(\log^3 n)$ time schedule proposed in [12]. Our new algorithm should be also seen as the separation result between the planar and the general graphs with a small diameter due to the polylogarithmic inapproximability result in general graphs due to Elkin and Kortsarz [11].

2. GOSSIPING IN GENERAL GRAPHS WITH KNOWN TOPOLOGY

The gossiping task can be performed in two consecutive stages. During the first stage we gather all individual messages in one (central) point of the graph. Then, during the second stage, the collection of individual messages is broadcast to all nodes in the network. We start this section with the presentation of a simple gathering procedure that works in time $O((D+\Delta)\log n)$ in free trees. Later we show how to choose a spanning breadth-first (BFS) tree in an arbitrary graph G in order to gather (along its branches) all messages in G also in time $O((D+\Delta)\log n)$, despite the additional edges in G which might potentially cause additional collisions. Finally, we show how the gathering process can be pipelined and sped up to time $O(D+\Delta\log n)$.

2.1 Ranking procedure

Given an arbitrary tree, we choose as the root its central node c. The nodes in a tree rooted in c are partitioned into consecutive layers $L_i=\{v \mid dist(c,v)=i\}$, for $i=0,..,r$ where r is a radius of the tree. We denote the size of each layer L_i by $|L_i|$.

We adopt a standard definition of the rank of nodes in a rooted tree, used earlier in the context of radio communication in known topology networks in [13]. This ranking was used even earlier in defining the *Strahler number* of binary trees, introduced in hydro-geology [23] and later studied extensively in computer science (cf. [24] and the references therein).

Specifically, every leaf v has $\texttt{rank}(v)=1$. A non-leaf node determines its rank according to the rank of its children as follows. Given the ranks of the children of a node v, say $r_1,\cdots,r_k$, let $r_{max}=\max(r_i)$. If v has a unique child whose rank is r_{max}, then the rank of node v is set to $\texttt{rank}(v)=r_{max}$. Otherwise, there are at least two children with the rank r_{max}, in which case the rank of node v is set to $\texttt{rank}(v)=r_{max}+1$. For an illustration see Figure 1.

LEMMA 2.1. *The largest rank in a tree of size n is bounded by $\lceil\log n\rceil$. (see [8]).*

The schedule is now defined in stages using the ranked tree. This is done by partitioning the nodes into different *rank sets* $R_i = \{v \mid \texttt{rank}(v) = i\}$, where $1 \le i \le r_{max} \le \lceil \log n \rceil$. The meaning of this partition is that the nodes in R_i are involved in transmissions only during the stage i.

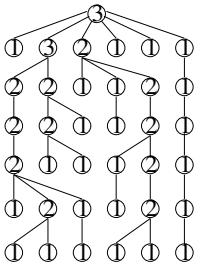

Figure 1: A ranked tree of size $n = 37$

We now define two key subsets of nodes for each stage. The *fast transmission set*:

$F_i^k = \{v \mid v \in L_k \cap R_i \text{ and } parent(v) \in R_i\}$.

Also define $F_i = \bigcup_{k=1}^{D} F_i^k$ and $F = \bigcup_{i=1}^{r_{max} \le \lceil \log n \rceil} F_i$.

The *slow transmission set*:

$S_i^k = \{v \mid v \in L_k \cap R_i \text{ and } parent(v) \in R_j, j > i\}$.

Also define $S_i = \bigcup_{k=1}^{D} S_i^k$ and $S = \bigcup_{i=1}^{r_{max} \le \lceil \log n \rceil} S_i$.

LEMMA 2.2. *During the ith stage, all nodes in F_i^k can transmit to their parents simultaneously without any collisions, for $i = 1, .., r_{max} \le \lceil \log n \rceil$ and $k = 1, .., D$.*

Proof. Consider any two distinct nodes u and v in F_i^k, and suppose they interfere with each other, namely, have a neighbor in common. Obviously, these two nodes must have the same parent x in the tree. Moreover, according to the definition of the fast transmission set F_i^k, the ranks of u, v and x are i. However, according to the definition of the ranking procedure, if $\texttt{rank}(u) = \texttt{rank}(v) = i$ then the rank of the node x must be at least $i+1$. Hence the nodes u and v cannot both belong to F_i^k, which leads to contradiction. ∎

The following procedure moves messages from all nodes with rank i into their parents with ranks $i+1$ or higher.

Procedure GATHERING(i);

1. Move messages from nodes in F_i to S_i;
   from F_i^D down to F_i^1 layer by layer.
2. Move messages from nodes in S_i to their parents;
   all parents collect their messages from their children in S_i one by one.

The time complexity of step 1 is $O(D)$ due to Lemma 2.2. The time complexity of step 2 is bounded by $O(\Delta)$ where Δ is the maximum degree of the tree.

Due to Lemma 2.1, procedure GATHERING completes the gathering stage in time $O((D + \Delta) \log n)$.

THEOREM 2.3. *In any tree of size n, diameter D and maximum degree Δ, the gossiping task can be completed in time $O((D + \Delta) \log n)$.*

Proof. Gathering all messages at the root of the tree is done in time $O((D+\Delta) \log n)$. This is followed by the trivial broadcasting stage in time D. ∎

2.2 Gathering messages in arbitrary graphs

We start this section with an introduction of a novel concept of a *gathering spanning tree* (GST). These trees play a crucial role in time efficient gossiping in arbitrary graphs. Indeed, we show that an arbitrary graph G contains a GST. We also propose an $O(n^2)$-time algorithm that constructs the GST for any graph of size n and diameter D. In the concluding part of this section, we propose a new more efficient schedule that completes message gathering in time $O(D + \Delta \log n)$.

2.2.1 The construction

We start this subsection with the definition of the gathering spanning tree.

In an arbitrary graph $G = (V, E)$, any BFS spanning tree T_G of G s.t.

(1) T_G is rooted at the central node c of G,

(2) T_G is ranked, and

(3) all nodes in F_i^k of T_G are able to transmit their messages to their parents simultaneously without any collision, for all $1 \le k \le D$ and $1 \le i \le r_{max} \le \lceil \log n \rceil$,

is called a *gathering spanning tree*, or simply *GST*.

In a graph $G = (V, E)$, a *pre-gathering-tree* $T_{PGT} = (V, E_{PGT})$ is an arbitrary ranked BFS spanning tree rooted in the central node c. The procedure for constructing a gathering spanning tree from a pre-gathering-tree T_{PGT} is based on the *pruning process*. During the pruning process, a function CHECK-COLLISION(i, j) is used to detect two nodes in F_j^i (if such a pair exists) that are not able to transmit simultaneously their messages to their parents in T_{PGT}. The function returns either a pair of nodes (u, v) whose transmissions may cause a collision or '*null*' if such a pair does not exist. The outcome is illustrated in Figure 2.

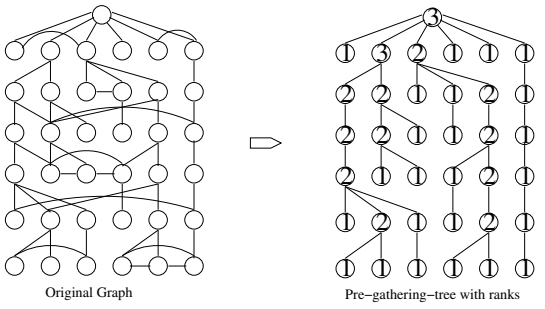

Figure 2: Creating ranked pre-gathering-tree

Function CHECK-COLLISION(i, j): A PAIR OF NODES;
(1) if $\exists u, v \in F_j^i$ and $(u,\ parent(v)) \in E$, where $u \ne v$
then return(u, v);
else return('*null*');

Procedure GATHERING-SPANNING-TREE constructs a gathering-spanning-tree GST $\subseteq G$ on the basis of $T_{PGT} \subseteq G$. The pruning process is performed layer by layer starting from the bottom (layer D) of T_{PGT} (outer loop). At each layer we gradually *fix* the parents of all nodes who are potentially involved in collisions starting with nodes with

the highest rank at the layer (inner loop). The outcome is illustrated in Figure 3.

Procedure GATHERING-SPANNING-TREE(T_{PGT});
(1) For $i := D$ down to 1 do
(2) begin
(3) For $j := r_{max}$ down to 1 do
(4) begin
(5) While CHECK-COLLISION$(i, j) \neq$ '*null*' do
(6) begin
(7) **rank**$(parent(v)) = j + 1$;
(8) $F_j^i = F_j^i - \{v, u\}$;
(9) $S_j^i = S_j^i \cup \{v, u\}$;
(10) $E_{PGT} = E_{PGT} - \{(u, parent(u))\}$;
(11) $E_{PGT} = E_{PGT} \cup \{(u, parent(v))\}$;
(12) re-rank T_{PGT} only at the top BFS layers from $i - 1$ down to 0;
(13) recompute sets in F and S in new T_{PGT};
(14) end
(15) end
(16) end

In what follows, we use an inductive argument to show that the Procedure GATHERING-SPANNING-TREE constructs the GST of an arbitrary graph $G = (V, E)$ in time $O(n^2)$.

LEMMA 2.4. *After completing the pruning process at layer i in T_{PGT}, the structure of edges in T_{PGT} between layers $i-1, .., D$ is fixed, i.e., the transmissions within layers $i, .., D$ in all sets F_j, for $j = 1, .., r_{max} \leq \lceil \log n \rceil$, are free of collisions.*

Proof. We rely on the assumption that before the ith execution of the outer loop, all edges in T_{PGT} between layers from i through D are already fixed and will never change again. Note that during the pruning process at layer i, the updates involve only some edges between layers i and $i - 1$. Note also that the updates at layer i are executed always first at the nodes with higher ranks. Thus, after a pair of nodes u, v with the same rank j gets pruned and their now joint parent $parent(v)$ gets a higher rank $j + 1$, neither the pair u, v nor the edges leading to their new parent will be considered again. This is because the pair u, v no longer belongs to the set F and further updates at this layer at nodes with ranks j or lower cannot change this property. On the other hand, the former parent of u ($parent(u)$) might be downgraded to a lower rank after losing its child u. But this poses no problem to the correctness of the construction since the pruning process is performed at nodes with gradually decreasing ranks and the former parent of u will be reconsidered (if necessary) during some later stage of the pruning process at the layer i. The lemma follows. ∎

THEOREM 2.5. *There exists an efficient polynomial time construction of a GST on an arbitrary graph G.*

Proof. The claim follows directly from Lemma 2.4. ∎

2.2.2 $O((D + \Delta) \log n)$-time gossiping

Using the ranks of the GST nodes (constructed in the previous section), all nodes get partitioned into distinct *rank sets* $R_i = F_i \cup S_i$, where $1 \leq i \leq r_{max} \leq \lceil \log n \rceil$. Initially, all messages are gathered into the central node c, stage by

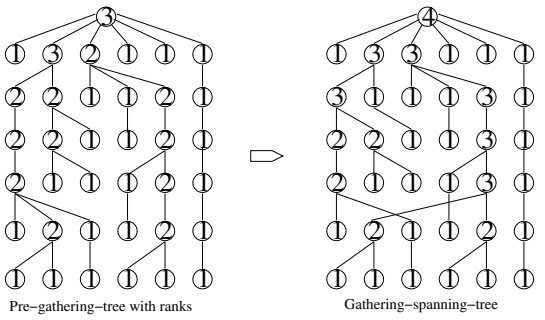

Figure 3: From pre-gathering-tree to gathering-spanning-tree

stage, using the structure of the GST. This is done as follows. During the ith stage, all messages from nodes in F_i are first moved to the nodes in S_i. In order to avoid collisions between transmissions originating at neighbouring BFS layers we divide the sequence of transmission time slots into three separate (interleaved) subsequences of time slots. Specifically, the nodes in S_i transmit in time slots: $t \equiv 0 \pmod 3$ iff $i \equiv 0 \pmod 3$; $t \equiv 1 \pmod 3$ iff $i \equiv 1 \pmod 3$; and $t \equiv 2 \pmod 3$ iff $i \equiv 2 \pmod 3$. Later, we move all messages from nodes in S_i to their parents in GST.

LEMMA 2.6. *In stage i, nodes in set S_i of the GST transmit their messages to the parents in time $O(\Delta)$.*

Proof. Lemma 4 in [17] states that one can move all messages between two partitions of a bipartite graph with max-degree Δ (in this case two consecutive BFS layers) in time Δ. The solution is based on the use of the *minimal covering set*. Please note that during this process a (possibly) combined message m sent by a node $v \in S_i$ may be delivered to the parent of another transmitting node $w \in S_i$ rather then to $parent(v)$. But this is fine, since now the time of delivery of the message m to the root of the tree is controlled by the delivery mechanism from the node w. Obviously this flipping effect can be observed a number of times in various parts of the tree, though each change of the route does not change the delivering mechanism at all.

In order to avoid extra collisions caused by nodes at neighbouring BFS layers, we use the solution with three separate interleaved subsequences of time slots incurring a slowdown with a multiplicative factor of 3. ∎

Upon the completion of the gathering stage, the gossiping problem is reduced to the broadcasting problem. We distribute all messages to every node in the network by reversing the direction and the time of transmission of the gathering stage. In section 3 we prove that the broadcasting stage can be performed faster in graphs with large Δ, i.e., in time $D + O(\log^3 n)$.

THEOREM 2.7. *In any graph G, the gossiping task can be completed in time $O((D + \Delta) \log n)$.*

Proof. During the ith stage, all messages from F_i are moved to S_i in time $O(D)$. This is feasible due to the fact that the maximum distance between any two nodes in GST is limited to D and the properties of the GST. According to Lemma 2.6, all nodes in the set S_i are able to move their messages to their parents in T_G in time $O(\Delta)$. Since this

process has to be repeated not more than $\log n$ times (the number of different ranks, see Lemma 2.1), we conclude that the gossiping time can be bounded by $O((D+\Delta)\log n)$. ∎

2.3 $O(D + \Delta \log n)$-time gossiping

In the previous section, the transmission process was split into $\lceil \log n \rceil$ separate stages, each costing $O(D + \Delta)$ units of time. In this section we show how to pipeline the transmissions of different stages. This will allow a new gossiping schedule of length $O(D + \Delta \log n)$.

In order to achieve this, the communication process is split into consecutive blocks of 6 time units each. The first 3 units of each block will be used for fast transmissions from the set F, and the remaining 3 will be used for slow transmissions of nodes from the set S. We use 3 units of time for each type of transmission to avoid collisions between neighbouring BFS layers, similarly to what was done in the last section. Recall that due to Lemma 4 in [17] we can move all messages between two consecutive BFS layers in time Δ. We compute for each node $v \in S$ at layer i a number of a step $1 \leq s(v) \leq \Delta$ in which the node v can transmit without interruption from other nodes in S at the layer i.

The pattern of transmissions of a node v at layer i and with rank j in GST depends on whether it belongs to the set F or to the set S, and it is as follows:

(1) if $v \in F$, then v transmits within the time block $(D - i) + j \cdot \Delta$

(2) otherwise ($v \in S$), v transmits within the time block $(D - i) + j \cdot \Delta + s(v)$.

LEMMA 2.8. *A node v transmits its message as well as all messages collected from its descendants towards its parent in GST successfully during the time block allocated to it by the pattern of transmissions.*

Proof. First note that according to the pattern of transmissions all descendants of the node v transmit in earlier time blocks. E.g., if a descendant w of v is at layer $i' < i$ and the same rank j, then the first term of the expression $(D-i)+j \cdot \Delta$ is smaller for w. If w has also smaller rank j', then both terms of the expression are smaller. This means that the node w transmits earlier than the node v.

We now prove that any node v following the pattern of transmissions will transmit to its parent without being interrupted by anyone else. First note that there will be no collisions between neighbouring BFS layers thanks to the separation into three subsequences, ensuring that three time units are available within each block. Also there will be no collision between transmissions coming from different types of transmission (fast and slow), thanks to the two parts of each time block. Hence we have to consider only potential collisions within the same type of transmission at the same BFS layer in GST. Assume that $v, w \in F$ and they are at the same BFS layer i in GST. If v and w have also the same rank j, then they do not interrupt each other due to the property of GST. If they have different ranks j and j' respectively, then they transmit in different time blocks $(D - i) + j \cdot \Delta$ and $(D - i) + j' \cdot \Delta$, so they do not interrupt each other. Now assume that $v, w \in S$ and they are at the same BFS layer i in GST. Then they do not interrupt each other since either their ranks are different and both $s(v), s(w) \leq \Delta$, or if they have the same rank j, then they have different values of $s(v)$ and $s(w)$. This completes the proof. ∎

Since the number of blocks used in the schedule is limited to $D + \Delta \log n$, the following theorem holds.

THEOREM 2.9. *In any graph G, the gossiping task can be completed in time $O(D + \Delta \log n)$.*

Using this theorem we can extend the class of graphs from [17] in which the gossiping can be completed in time $O(D)$ as follows.

COROLLARY 2.10. *The gossiping can be completed in time $O(D)$ in all graphs with $\Delta = O(\frac{D}{\log n})$.*

3. BROADCASTING IN GRAPHS WITH KNOWN TOPOLOGY

In this section we present the idea of a new deterministic algorithm B that generates a schedule for completing the broadcasting task in a general known topology radio network in time $D + O(\log^3 n)$. We then show a randomized variant B' of this algorithm that with high probability broadcasts the message in time $D + O(\log^2 n)$. This establishes the existence of a broadcasting schedule of length $D + O(\log^2 n)$ for every n node radio network of diameter D. As far as constructive results are concerned, this is likely to be asymptotically optimal, in view of the polylogarithmic additive inapproximability result of [11].

The deterministic algorithm B uses the concept of the ranked gathering spanning tree (on this occasion rooted in the *source node* s) introduced in section 2. Similarly to the gossiping case the algorithm uses *fast* and *slow* transmission that partition the set of nodes to the same sets F_i and S_i. However, this time the broadcast message is disseminated from the root towards the leaves of the tree. Correspondingly, to each node $v \in F_i$ we associate the *fast edge* connecting it to its parent, $(parent(v), v)$. Note that after reversing the direction of fast transmissions within the same stage (i.e., letting the transmissions between nodes of the same rank go from $parent(v)$ to v), all deliveries must still be successful. Otherwise there would have to be a crossing edge causing collisions. But such an edge would cause also collisions during the transmission in the gathering direction, which is not possible

Let us start with an overview of the broadcast process from the point of view of a copy of the message that was eventually received at some leaf a of the tree. Note that this message does not necessarily have to follow the unique shortest path $p(a)$ leading from the root of the tree to a. In fact, there are many paths along which the message could be forwarded, some of which do not even need to be shortest paths. For the sake of the time complexity analysis, however, we fix our attention on the path $p(a)$ and argue about the potential progress of the message along this path.

Conceptually, the path $p(a)$ consists in segments

$$p(a) \;=\; \langle p_1^F(a), p_1^S(a), p_2^F(a), p_2^S(a), \ldots, p_q^F(a), p_q^S(a) \rangle \;,$$

where each $p_i^F(a)$ is a segment consisting of fast transmission edges (i.e., edges leading from $parent(v)$ to v of the same rank) and each $p_i^S(a)$ is an edge (u, w) where u is a node

on layer L_k for some k, w is a node on layer L_{k+1} and $\texttt{rank}(u) > \texttt{rank}(w)$. We refer to such edges (u,w) as slow transmission edges. (Note that some of the segments $p_i^F(a)$ may be null.)

Again, we stress that in reality, the message need not follow this path. Nevertheless, we may consider the "progress" of the message along this path, by measuring the delay from the time the message is already available at some node v on the path $p(a)$ to the time the message has already reached the following node w on the path (though not necessarily via a transmission from v). Hence conceptually, the message progress can be viewed as traversing the path $p(a)$ by alternating (flipping) between chains $p_i^F(a)$ of fast transmission edges connecting nodes of the same rank and slow transmission steps over edges $p_i^S(a)$, connecting high rank nodes to lower rank nodes.

Let us next describe the schedule governing these transmissions. During the broadcasting process the nodes in the tree use the following pattern of transmissions. Let $r_{max} \leq \lceil \log n \rceil$ be the largest rank in the tree. Consider a node v of rank $1 \leq j \leq r_{max}$ on BFS layer L_i with a child w of the same rank at the next BFS layer. Then v is set to perform a fast transmission to w in time steps t satisfying $t \equiv i + 6j \pmod{6r_{max}}$. Observe that in real terms, v will perform such a fast transmission exactly once, on the first appropriate time slot after it receives the message for the first time. The slow transmissions at the BFS layer L_i are performed in time steps t satisfying $t \equiv i+3 \pmod 6$. Note that this pattern of transmissions separates the fast and the slow transmissions at any BFS layer by three units of time. Thus there are no collisions between the fast and the slow transmissions at the same BFS layer. The pattern also ensures that at any time step, transmissions are performed on BFS layers at distances that are multiples of 3 apart. Thus there will be no conflicts between transmissions coming from different BFS layers.

Note also that once the broadcast message arrives at the first node v of a fast segment $p_i^F(a)$ of the route (with a particular rank), it may have to wait for as many as $6r_{max} = O(\log n)$ time steps, but then, when finally transmitted to the next BFS layer, it will be forwarded through the fast segment $p_i^F(a)$ without further delays.

Once reaching the end node u of the fast segment $p_i^F(a)$, the message has to be transmitted from some node on u's BFS layer to the next node w on $p(a)$, which is of lower rank, using a slow transmissions mechanism. For slow transmissions, algorithm B uses the $O(\log^2 n)$ transmission Procedure CW proposed by Chlamtac and Weinstein in [5]. Procedure CW allows to move uniform information from one partition of a bipartite graph of size n (here, an entire BFS layer L_j of the tree) to the other (here, the next layer L_{j+1}) in time $O(\log^2 n)$. The slow transmission mechanism based on Procedure CW is run repeatedly in a periodic manner at every BFS layer of the tree. In particular, at any BFS layer, the steps of the slow transmission procedure CW are performed in every 6th step of the broadcasting schedule.

Hence, suppose the broadcast message traversing towards any destination a in the tree has reached a node u of BFS layer L_j on its path $p(a)$, such that the next edge (u,w) on the path is a slow transmission edge. It is possible that neither u nor any other neighbor of w on BFS layer L_j participates in the current activation of procedure CW on L_j (possibly because neither of those nodes had the message at the last time the procedure was activated). Nevertheless, u will participate in the next activation of procedure CW on BFS layer L_j, which will be started within at most $O(\log^2 n)$ time (namely, the time required for the current activation to terminate). Moreover, it is guaranteed that by the time that activation of procedure CW terminates, w will have the message (although it may get it from any of its neighbors in L_j, and not necessarily directly from u). Hence this entire stage can be thought of as a slow transmission operation on the edge (u,w), taking a total of at most $O(\log^2 n)$ time steps.

Thus the total time required for the broadcast message to reach a leaf a in the tree can be bounded as follows. Let D_i, for $1 \leq i \leq r_{max}$, denote the length of $p^F(a)$, the ith fast segment of the route $p(a)$ used by the broadcast message that has reached a. Thus the time required to communicate a is bounded by $O(\log n)+D_1+\ldots+O(\log n)+D_{r_{max}} \leq D+O(\log^2 n)$ for the fast transmissions plus $r_{max} \cdot O(\log^2 n) = O(\log^3 n)$ for the slow transmissions, yielding a total of $D + O(\log^3 n)$.

THEOREM 3.1. *There exists a deterministic polynomial time algorithm that constructs, for any n node radio network of diameter D, a broadcasting schedule of length $D + O(\log^3 n)$.*

We replace now Procedure CW with the following randomized procedure RCW. Procedure RCW works for a block of $\lceil \log n \rceil$ rounds, and is activated repeatedly in a periodic manner every block of $K = \lceil \log n \rceil$ rounds at every BFS layer L_j of the tree. At the beginning of each block on layer L_j of the tree, the nodes of L_j that have a copy of the message join the activation and participate in the procedure. (Nodes that get the message during the time block will join the process only in the next activation and after.) Each participating node v in each round $1 \leq i \leq K$ decides whether to transmit the message randomly and uniformly with probability $1/2^i$. We observe the following.

CLAIM 3.2. *Consider an uninformed node w in L_{j+1}. Suppose that at the beginning of the current activation of procedure RCW, w has some informed neighbors on layer L_j. Then w will get the message during the current activation of procedure RCW with constant probability $p \geq 1/(4e)$.*

Proof. Let $d(w)$ denote the number of informed neighbors w has on layer L_j at the beginning of the current activation. Let $1 \leq i \leq K$ be an integer satisfying $2^{i-1} < d(w) \leq 2^i$. Then in step i of the activation, w will get the message with probability

$$p \;\geq\; d(v) \cdot \frac{1}{2^i} \cdot \left(1 - \frac{1}{2^i}\right)^{d(w)-1} \;>\; 2^{i-1} \cdot \frac{1}{2^i} \cdot \left(1 - \frac{1}{2^i}\right)^{2^i - 1}$$

$$> \;\frac{1}{2} \cdot \left(1 - \frac{1}{2^i}\right) \cdot \frac{1}{e} \;>\; \frac{1}{4e} \;.$$

∎

Now consider an arbitrary node v in the graph, and consider the path along which it is supposed to get the message from the root. This path is divided into "fast segments" and "slow steps" as discussed above. As argued, the fast segments require $D + O(\log^2 n)$ steps. We now claim that the total number of time cycles spent by the message for

the slow steps on its way to v is at most $O(\log^2 n)$ as well. More precisely, we argue that with high probability, the message will participate in at most $O(\log n)$ activations of procedure RCW. To see this, note that each participation of the message in an activation of procedure RCW succeeds (i.e., the message crosses from its current node to the next node on the path to v) independently with constant probability $p \geq 1/(4e)$. Hence, letting X be a random variable denoting the number of successes of the message during $24eK$ participations in procedure RCW, the expected value of X is $\mu = 6K$. Clearly, the message arrives at v after at most $K = \lceil \log n \rceil$ successes. By Chernoff's bound, the probability $P_{fail}(v)$ that the message will not reach v after $24eK$ participations in procedure RCW can be bounded from above by

$$P_{fail}(v) \;\leq\; P(X < K) \;=\; P(X < (1 - 5/6)\mu)$$

$$< \;\exp\left(-\frac{1}{2}\left(\frac{5}{6}\right)^2 \mu\right) \;<\; n^{-2}\,.$$

Subsequently, the probability that for *some* vertex v in G, the message will require more than $24eK$ participations in procedure RCW until it reaches v, is smaller than $1/n$. We thus have the following result.

THEOREM 3.3. *There exists a randomised algorithm that for any known topology of n node radio network of diameter D and any source node s, following a polynomial preprocessing stage, broadcasts a message from s with high probability in time $D + O(\log^2 n)$.*

COROLLARY 3.4. *For any known topology n nodes radio network of diameter D, there exists a broadcasting schedule of length $D + O(log^2 n)$.*

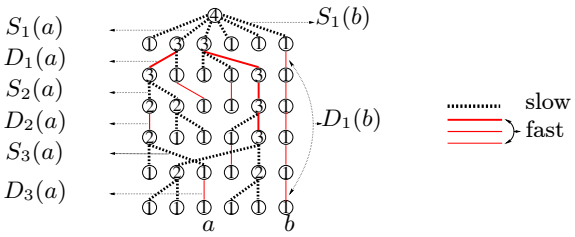

Figure 4: An example of broadcasting schedule

4. ASYMPTOTICALLY OPTIMAL RADIO BROADCAST ON PLANAR GRAPHS

In this section we sketch an algorithm for constructing a transmission schedule for performing broadcast from a given source s on a known planar radio network G in asymptotically optimal $O(D)$ time. The schedule consists of D phases, each of up to 3 rounds, where the ith phase involves transmitting the message to the vertices at distance i from the source s. Layering the graph G by distance from s, let L_p denote the set of vertices at distance p from s. The first phase of the schedule consists of a single round where only s transmits, and by the end of this round, all the vertices of L_1 are informed. Assuming all the vertices of layer $U = L_{p-1}$ are informed, let us now describe the algorithm for constructing the sub-schedule of phase p, designed to inform all the vertices of $D = L_p$.

The algorithm starts with a preprocessing stage, whose purpose is to construct a bipartite graph consisting only of the nodes of the two layers U and D and the edges connecting them. All the layers below D in the original graph are discarded, and all the layers above U in the original graph are replaced by a star connecting the source s to the nodes of layer U. See Figure 6(b) for the intended final output of the preprocessing stage.

The preprocessing stage operates in a number of substages. Let us start by constructing a planar embedding of G with s at the top (on the outer face) and all other vertices below it. Erase from the graph all the vertices of layers L_j for $j > p$ and their edges, as well as all the edges connecting vertices of D. In the resulting graph, each vertex of D is connected only to vertices of U. Now mark on the graph a shortest paths tree T rooted at s and leading to all the vertices of U. The leaves of this tree are precisely the vertices of U. An example outcome of this process is illustrated in Figure 5(a). Next, erase from the graph all the

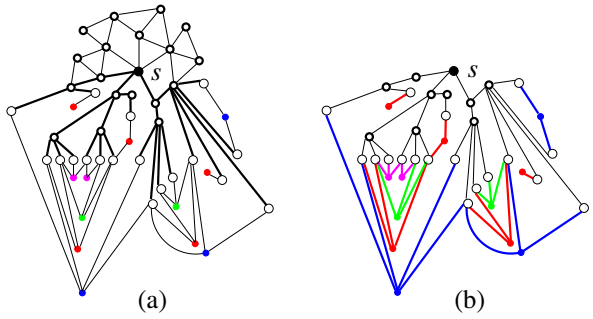

Figure 5: Example: Phase p=4, where the vertices of layer $U = L_3$ (plain hollow circles) must transmit to the vertices of layer $D = L_4$ (solid circles). Bold hollow circles are vertices of layers L_1 and L_2. (a) Bold lines mark the edges of the tree T leading from s to layer U. (b) Picture after removal of irrelevant vertices and edges from layers L_1 and L_2. Bold lines mark the edges connecting U to L.

vertices of layers other than D and U that do not participate in this tree. An outcome of this process is illustrated in Figure 5(b).

Next, we replace the tree T by a star connecting s directly to the vertices of layer U. To see that this transformation does not affect the planarity of the graph, observe that it can be done by gradually contracting edges in the tree T and gluing together adjacent vertices while moving the vertices on the plane so as to preserve the planarity of the embedding. The outcome of this on the graph of Figure 5(b) is illustrated in Figure 6(a).

Next, we modify the embedding so that the vertices of layer U occur on a straight horizontal line and the vertices of layer D occur below this line. For our example, this yields the embedding depicted in Figure 6(b).

We now assign depth values to the vertices of D. This is done recursively as follows. Let $d = 1$. Assign each vertex $v \in D$ on the outer face a depth value $depth(v) = d$. Now erase all the vertices of D on the outer face and their edges, and increase d by 1. If D is still nonempty then recurse.

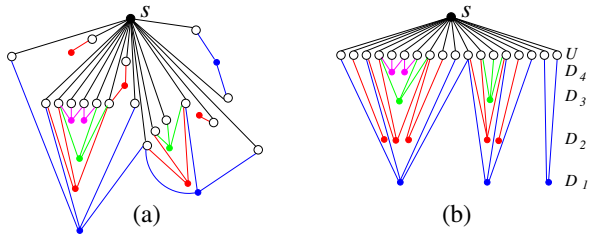

Figure 6: (a) Transforming the shortest paths tree T from s to the vertices of U into a star. (b) The embedding with the vertices of U on a straight line.

The depth values partition the set D into subsets D_ℓ for $\ell \geq 1$ such that D_ℓ contains all the D vertices of depth ℓ. The resulting depth values and sets D_ℓ for our example are depicted in Figure 6(b).

For each vertex $v \in D$, denote its leftmost U neighbour by $\texttt{left}(v)$, its rightmost U neighbour by $\texttt{right}(v)$, and the list of its remaining neighbours (if any) by $\texttt{rest}(v)$, taken from left to right. Also, order the vertices of each set D_ℓ as an ordered list going from left to right consistently with the order of their U neighbours, i.e., so that v_1 occurs to the left of v_2 if $\texttt{right}(v_1)$ is the same as or to the left of $\texttt{left}(v_2)$.

Finally, the schedule is defined as follows. The three time slots of the current phase p are $t_1 = 3p - 4$, $t_2 = 3p - 3$ and $t_3 = 3p - 2$. The procedure operates in stages, where stage ℓ constructs the part of the schedule responsible for informing the depth ℓ vertices of D. In particular, stage ℓ starts with the depth 1 vertices, kept in the ordered list $D_1 = \langle v_1, \ldots, v_k \rangle$. Construct the ordered list of "breakpoint vertices"

$$B = \langle \texttt{left}(v_1), \texttt{right}(v_1), \ldots, \texttt{left}(v_k), \texttt{right}(v_k) \rangle.$$

Note that some of the vertices in this list may coincide, namely, $\texttt{right}(v_i)$ may equal $\texttt{left}(v_{i+1})$ for some i's, in which case only one copy is kept in the list B. Assign time slots t_1 and t_2 alternately to the vertices of the list B. Next, for each v_i with nonempty list $\texttt{rest}(v_i)$, assign time slots to the vertices of that list as follows. If $\texttt{left}(v_i)$ was assigned the time slot t_1 (hence $\texttt{right}(v_i)$ was assigned the time slot t_2), then assign the time slots t_3 and t_1 alternately to the nodes of $\texttt{rest}(v_i)$ from left to right. Similarly, if $\texttt{left}(v_i)$ was assigned the time slot t_2 (and $\texttt{right}(v_i)$ was assigned the time slot t_1), then assign the time slots t_3 and t_2 alternately to the nodes of $\texttt{rest}(v_i)$ from left to right. See Figure 7(a) for an illustration. Observe that at this point in

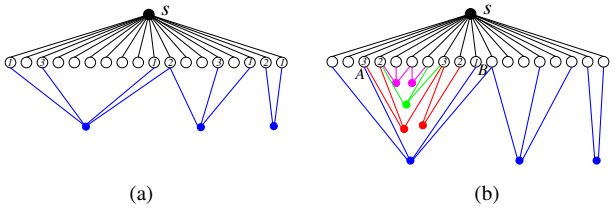

Figure 7: (a) Assigning transmission times to depth 1 vertices of D. (Here i represents time slot t_i for $i = 1, 2, 3$.) (b) Assigning transmission times to depth 2 vertices of D connected to U vertices between A and B.

time, for every vertex $v \in D_1$, we have the following properties. First, all the U neighbours of v have been assigned a time slot. Second, this assignment ensures that v gets the message at some time slot among $\{t_1, t_2, t_3\}$. Third, letting $U_{assigned}$ denote the list of U vertices that have already been assigned time slots, ordered from left to right, we have the property that every two consecutive vertices in $U_{assigned}$ are assigned different time slots.

Let us now describe stage $\ell \geq 2$, showing how to assign time slots to some U neighbours of vertices in D_ℓ so as to ensure that they get the message during the current phase. The inductive hypotheses we rely on at the beginning of stage ℓ are the following:

(P1) At the end of stage $\ell - 1$, all the neighbours of vertices of D_k for $k < \ell$ were already assigned time slots,

(P2) these previously made assignments ensure that all the vertices of D_k for $k < \ell$ receive the message during the phase, and

(P3) at the end of stage $\ell - 1$, every two consecutive vertices in $U_{assigned}$ are assigned different time slots.

Consider some vertex $v \in D_\ell$. Let A be the rightmost U vertex to the left of $\texttt{left}(v)$ that has already been assigned a time slot t_A previously. (A can possibly be $\texttt{left}(v)$ itself.) Similarly, let B be the leftmost U vertex to the right of $\texttt{right}(v)$ (possibly $\texttt{right}(v)$ itself) that has already been assigned a time slot t_B previously. (See Figure 7(b), where the assigned time slots are $t_A = t_3$ and $t_B = t_1$ respectively.) Note that $\texttt{left}(v)$ and $\texttt{right}(v)$ may have already been assigned a time slot previously, but the vertices of $\texttt{rest}(v)$ (if any exist) are necessarily still unassigned at the beginning of stage ℓ. Moreover, if both $\texttt{left}(v)$ and $\texttt{right}(v)$ have been assigned a time slot previously (in which case $A = \texttt{left}(v)$ and $B = \texttt{right}(v)$), then these time slots must be different by the inductive hypothesis (P3), as A and B occur consecutively in $U_{assigned}$.

Now assign time slots to all unassigned U neighbours of v as follows. Let $t_C \in \{t_1, t_2, t_3\}$ be a time slot different from t_A and t_B. Go over the unassigned U neighbours of v from left to right, and assign them the time slots t_C and t_A alternately. See Figure 7(b).

Note that once this is done, the inductive hypotheses are guaranteed for v, namely, (P1) all the neighbours of v are already assigned time slots, (P2) v is ensured to receive the message during the current phase, and (P3) every two consecutive vertices in $U_{assigned}$ are assigned different time slots.

The final assignment of time slots for layer U vertices in our example graph is depicted in Figure 8. By the fact that the inductive hypotheses are maintained throughout the process of assigning time slots to the vertices of U, it is clear that the resulting 3-round schedule ensures that all the vertices of D receive the message by the end of the phase. We have the following.

THEOREM 4.1. *Given a known planar radio network G of diameter D and a source vertex s in G, it is possible to construct (in polynomial time) a transmission schedule for performing broadcast from s in $O(D)$ time (which is asymptotically optimal).*

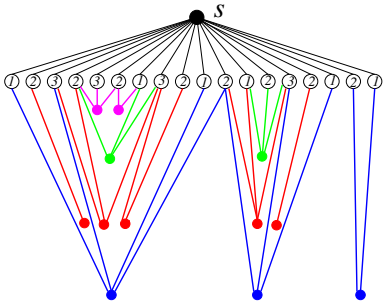

Figure 8: Final assignment of transmission times in D.

5. CONCLUSION

We proposed here new efficient (polynomial time) construction of the deterministic schedule that performs the gossiping task in radio networks in time $O(D + \Delta \log n)$. The solution is based on the new concept of the gathering spanning tree. The new gossiping schedule is asymptotically optimal if $\Delta = O(\frac{D}{\log n})$. Unfortunately, when Δ is larger, our schedule might be far from being optimal. Though it provides the best known upper bound for all $\Delta = O(\frac{n}{\log n})$. The search for the optimal deterministic gossiping schedule for a wider classes of graphs remains the main unsolved problem here. The evident open problem regarding broadcast is whether there exists a deterministic broadcast schedule of time $O(D + \log^2 n)$ for any n-node graph G of diameter D.

6. REFERENCES

[1] B. Awerbuch and D. Peleg. Sparse partitions. *Proc. 31st Symp. on Foundations of Computer Science*, 1990, pp. 503-513.

[2] N. Alon, A. Bar-Noy, N. Linial and D. Peleg. A lower bound for radio broadcast. *J. Computer and System Sciences* 43, (1991), 290 - 298.

[3] R. Bar-Yehuda, O. Goldreich and A. Itai. On the time complexity of broadcasting in radio networks: an exponential gap between determinism and randomization. *Proc. 5th Symp. on Principles of Distributed Computing*, 1986, 98 - 107.

[4] I. Chlamtac and S. Kutten. On broadcasting in radio networks-problem analysis and protocol design. *IEEE Trans. on Communications* 33, (1985), pp. 1240-1246.

[5] I. Chlamtac and O. Weinstein. The wave expansion approach to broadcasting in multihop radio networks. Proc. *Proc. INFOCOM*, 1987.

[6] M. Christersson, L. Gąsieniec and A. Lingas. Gossiping with bounded size messages in ad-hoc radio networks. *Proc. 29th Int. Colloq. on Automata, Languages and Programming*, 2002, pp. 377-389.

[7] M. Chrobak, L. Gąsieniec and W. Rytter, Fast broadcasting and gossiping in radio networks. *J. of Algorithms* 43(2), (2002), pp. 177-189.

[8] T.H. Cormen, C.E. Leiserson and R.L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[9] A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. *Proc. 44th Symp. on Foundations of Computer Science*, 2003, pp. 492-501.

[10] K. Diks, E. Kranakis and A. Pelc. The impact of knowledge on broadcasting time in radio networks. *Proc. 7th European Symp. on Algorithms*, 1999, pp. 41-52.

[11] M. Elkin and G. Kortsarz. Polylogarithmic inapproximability of the radio broadcast problem. *Proc. APPROX*, 2004, LNCS 3122.

[12] M. Elkin and G. Kortsarz. Improved broadcast schedule for radio networks. *Proc. 16th ACM-SIAM Symp. on Discrete Algorithms*, 2005.

[13] I. Gaber and Y. Mansour. Broadcast in radio networks. *Proc. 6th ACM-SIAM Symp. on Discrete Algorithms*, 1995, pp. 577-585.

[14] L. Gąsieniec and A. Lingas. On adaptive deterministic gossiping in ad hoc radio networks, *Information Processing Letters* 2(83), 2002, pp. 89-94.

[15] L. Gąsieniec, E. Kranakis, A. Pelc and Q. Xin. Deterministic M2M multicast in radio networks. *Proc. 31st Int. Colloq. on Automata, Languages and Programming*, 2004, LNCS 3142, pp. 670-682.

[16] L. Gąsieniec and I. Potapov, Gossiping with unit messages in known radio networks. *Proc. 2nd IFIP Int. Conference on Theoretical Computer Science*, 2002, pp. 193-205.

[17] L. Gąsieniec, I. Potapov and Q. Xin. Efficient gossiping in known radio networks. *Proc. 11th Int. Colloq. on Structural Information and Communication Complexity*, 2004, LNCS 3104, pp. 173-184.

[18] L. Gąsieniec, T. Radzik and Q. Xin. Faster deterministic gossiping in ad-hoc radio networks. *Proc. 9th Scandinavian Workshop on Algorithm Theory*, 2004, LNCS 3111, pp. 397-407.

[19] D. Kowalski and A. Pelc. Centralized deterministic broadcasting in undirected multi-hop radio networks. *Proc. APPROX*, 2004, LNCS 3122, pp. 171-182.

[20] D. Liu and M. Prabhakaran. On randomized broadcasting and gossiping in radio networks. *Proc. 8th Int. Conf. on Computing and Combinatorics*, 2002, pp. 340-349.

[21] A. Sen and M.L. Huson. A new model for scheduling packet radio networks. *Proc. 15th Joint Conf. of IEEE Computer and Communication Societies*, 1996, pp. 1116-1124.

[22] P.J. Slater, E.J.Cockayne and S.T. Hedetniemi. Information dissemination in trees. *SIAM J. on Computing* 10, (1981), pp. 892–701.

[23] A.N. Strahler. Hypsometric (area-altitude) analysis of erosional topology. *Bull. Geol. Soc. Amer.* 63, (1952), pp. 117–1142.

[24] X.G. Viennot. A Strahler bijection between Dyck paths and planar trees. *Discrete Mathematics* 246, (2002), pp. 317–329.

[25] Y. Xu. An $O(n^{1.5})$ deterministic gossiping algorithm for radio networks. *Algorithmica*, 36(1), (2003), pp. 93–96.