Multicommodity Multicast, Wireless and Fast

R. Ravi
Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA
https://www.contrib.andrew.cmu.edu/~ravi/
ravi@andrew.cmu.edu

Oleksandr Rudenko
Carnegie Mellon University, Pittsburgh, PA, USA
orudenko@andrew.cmu.edu

Abstract

We study rumor spreading in graphs, specifically multicommodity multicast problem under the wireless model: given source-destination pairs in the graph, one needs to find the fastest schedule to transfer information from each source to the corresponding destination. Under the wireless model, nodes can transmit to any subset of their neighbors in synchronous time steps, as long as they either transmit or receive from at most one transmitter during the same time step. We improve approximation ratio for this problem from $\tilde{O}(n^{\frac{2}{3}})$ to $\tilde{O}(n^{\frac{1}{2}+\varepsilon})$ on n-node graphs. We also design an algorithm that satisfies p given demand pairs in $O(\mathrm{OPT}+p)$ steps, where OPT is the length of an optimal schedule, by reducing it to the well-studied packet routing problem. In the case where underlying graph is an n-node tree, we improve the previously best-known approximation ratio of $O(\frac{\log n}{\log\log n})$ to 3. One consequence of our proof is a simple constructive rule for optimal broadcasting in a tree under a widely studied telephone model.

2012 ACM Subject Classification Networks → Routing protocols; Networks

Keywords and phrases Rumor, scheduling, broadcast, multicommodity, multicast, optimization algorithms, approximation, packet routing

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.78

Funding This material is based upon research supported in part by the U. S. Office of Naval Research under award number N00014-18-1-2099, and the U. S. National Science Foundation under award number CCF-1527032.

1 Introduction

1.1 Motivation and Formulation

Rumor spreading problems have been popular for decades motivated by applications ranging from increasing throughput in synchronizing networks [3], keeping object copies in distributed databases synchronized [11], to recreational mathematics [4]. Common objectives in rumor spreading involve the total number of messages, the total number of transmissions (especially when message sizes are bounded in transmissions) and the completion time. In this paper, we study the minimum completion time objective for the multicast version of the problem where a set of source terminals wish to send their messages to their respective subsets of sinks.

The requirements for rumor spreading problem range from sending a single message from a single source to all nodes (broadcast), a subset of nodes (multicast) or a more generalized multicommodity version of multicast that we study. The rules for message transmission that have been widely studied are synchronous and range from the telephone model [5], the radio model [1] and the recently introduced wireless [6] model that shares the features of the first two. We mainly study the wireless model in this paper.

Multicommodity Multicast Problem. Given a communication graph G and demand pairs of vertices $(s_1, t_1), \ldots, (s_p, t_p)$, where each source s_i has a unique piece of information (message or packet), the problem is to find a schedule that transfers the message from every source s_i to its corresponding destination t_i in a minimum number of steps. The rules under which the information is allowed to be sent in each step depend on the model.

Wireless Model. During each step, every node v in G is allowed to pick some of the vertices adjacent to v and send all information v has accumulated to those nodes at once (so there is no bound on message size). Nodes cannot receive from more than one source, nor can they send and receive during the same time step.

1.2 Related Work

The type of models that have been studied could be differentiated by the demand requirements and the rules under which information is spread. We highlight three different models relevant to our study, which roughly correspond to spanning, Steiner and generalized Steiner connectivity requirements studied in network design.

1. In the **Broadcast** problem, there is a single source (root) that must send its information to *all* nodes in the system.
2. In the more general **Multicast** problem, the root only needs its information to reach *a subset* of nodes in the system.
3. In the even more general **Multicommodity Multicast** problem, we are given source-destination pairs, and *every* source must transmit its message to the corresponding destination node.

We describe various models that define the rules of synchronous information transmission in rounds.

1. In the **Telephone model**, during one time step, every node with information is allowed to send it to *only one* of the adjacent nodes (neighbours) in the graph.
2. In the **Radio model**, a set of transmitting nodes send out information and only the other nodes that have unique transmitting neighbors can receive the information from that neighbor[1].
3. In the **Wireless model**, during one time step, every node with information is allowed to send it to *any* subset of neighbours.

In all models, nodes cannot receive from more than one source, nor can they send and receive during the same time step. The key difference between the radio and wireless models is that in the wireless model, receivers have tunable apparatus that allows them to listen in one exactly one of the neighbors that may be transmitting, whereas in the radio model, a node with two neighbors transmitting cannot receive any information due to interference. The most widely studied versions do not restrict the number of messages sent between a pair of nodes in each step, but capacitated versions are possible.

1.2.1 Telephone model

For the *broadcast/multicast* problems under the telephone model Ravi [10] showed a poly-logarithmic approximation in general graphs, by relating it to spanning/Steiner trees that simultaneously have small max degree and diameter (the so-called *poise* of the graph).

[1] We do not discuss this model further in this paper.

For this *broadcast/multicast* problem under the telephone model Elkin and Kortsarz [2] gave the best known $O(\frac{\log k}{\log \log k})$-approximation factor, where k is the number of terminals the information should be delivered to.

▶ **Theorem 1** (Multicast approximation [2]). *Given graph G, root r and k target sinks, there is a poly-time algorithm* TelephoneMulticast *that finds a schedule that sends information from the root to all sinks in the telephone model and has approximation ratio $O(\frac{\log k}{\log \log k})$.*

The very general *multicommodity multicast* under telephone model was studied as well, where the best approximating ratio $\tilde{O}(2^{\sqrt{\log k}})$ was achieved by Nikzad and Ravi [9]. Note that this ratio is super-polylogarithmic but sub polynomial in k, the number of terminals. No better approximations are known and part of the difficulty stems from the inability to relate this objective directly to graph parameters since the solution can in general be disconnected. This problem was also studied for special classes of graphs. First, it is known that if G is a tree, then a polynomial time algorithm to find optimal schedule for telephone broadcast can be derived through dynamic programming. Recently Iglesias, Rajaraman, Ravi and Sundaram [7] have shown a $O(\frac{\log^3 k \log n}{\log \log n})$-approximation for multicommodity multicast for planar graphs.

1.2.2 Wireless model

Given that there are no restrictions on the number of neighbors receiving a transmission, for both the *broadcast and multicast* problems, optimal schedules under wireless model can be found in polynomial time using a simple *Breadth-First Search* (BFS) algorithm.

For the *multicommodity multicast* problem the best known approximate ratio was given by Iglesias, Rajaraman, Ravi and Sundaram [6], who also introduced the wireless model. They also showed an $O(\frac{\log n}{\log \log n})$-approximation when the underlying graph is an n-node tree, as well as $\tilde{O}(n^{\frac{2}{3}})$-approximation for general n-node graphs. There were two main ideas used in their paper: (i) If there are many sources that want to send to the same sink, we can satisfy them quickly using approximation algorithm for multicast under the telephone model; (ii) If there are many short disjoint path from sources to their respective destinations, we can satisfy them in parallel, otherwise we can reduce the number of sources considerably.

1.3 Packet routing problem

Another widely studied problem related to our work is so-called *store-and-forward packet routing problem*, where given packets at different source nodes, one needs to find a synchronous schedule of transporting them to given destinations, with the only restriction that no two packets can traverse the same edge at the same time step.

The main difference between store-and-forward packet routing and multicommodity multicast under wireless model is that the capacity restriction is put on the edge, not on the node. This means that a node can do the following: (i) receive several packets at the same time; (ii) send several packets as long as they use different edges; (iii) send and receive at the same time. What is not allowed in this model is to accumulate several packets and send them along one edge altogether, in contrast with the wireless model which can do that.

The analysis of packet routing problem involves a trade-off between the "dilation" parameter d (maximum path length) and the "congestion" criterion c (maximum number of paths using any edge). Note, that both c and d are lower bounds on the length of the optimal schedule for packet routing. Srinivasan and Teo [12] showed a polynomial time algorithm to find a schedule of length $O(c + d)$, which achieves a constant approximation. We reduce wireless multicommodity multicast problem to a packet routing problem to derive one of our results.

When the paths for the packets are given, Leighton, Maggs and Rao [8] initiated a long line of work that showed the existence and efficient construction of schedules of length $O(c+d)$, which was used extensively in the work cited above on finding such near optimal routes when the paths are not specified in advance.

1.4 Our Contributions

In this paper we focus on *multicommodity multicast* problem under the *wireless model*.

- We show that for any given constant $\varepsilon > 0$, there is a polynomial time $\tilde{O}(n^{\frac{1}{2}+\varepsilon})$-approximation algorithm that, given an arbitrary graph G and pairs of source-destination nodes (s_i, t_i), finds a schedule to transfer information from source s_i to destination t_i for all i (Theorem 2). This algorithm generalizes the ideas of Iglesias et al. [6] and applies them recursively.
- We design a polynomial time algorithm that, given an arbitrary graph G and p pairs of source-destination nodes (s_i, t_i), finds a schedule of length $O(\mathrm{OPT} + p)$ to transfer information from source s_i to destination t_i for all i, where OPT is the length of the optimal one (Theorem 10). To prove this result, we reduce our problem to an instance of packet routing in an appropriately defined auxiliary digraph.
- We show that there is a polynomial time 3-approximation algorithm that, given a tree T and pairs of source-destination nodes (s_i, t_i), finds a schedule to transfer information from source s_i to destination t_i for all i (Theorem 16). This result decomposes the schedule into sending messages up the tree followed by a phase that sends it down the tree.
- We give a **simple** optimal schedule for the broadcast in the tree under telephone model (Theorem 23). This is not a new result but a conceptual improvement over the previous strategies that all rely on dynamic programming by providing a simple explicit rule for choosing the transmitting pairs.

The widely studied telephone and relatively new wireless models are related to each other in the following way. Even though spreading information from one source to several sinks under wireless model can be done efficiently using BFS, collecting information from several sources into one sink is equivalent under both models. This follows from the fact that while collecting to one sink every piece of information travels on a straight path. This observation implies that the *gossip problem*, where every node in the graph has to sent its message to all other nodes, is equivalent under both models, up to a constant factor. More precisely, the optimal schedule for the gossip problem under the wireless model is no longer than twice the optimal broadcast schedule under the telephone model, which can be used to sweep all information to a fixed root and spread it back in a BFS tree to all nodes. In the other direction, the optimal schedule for the gossip problem under the wireless model is at least as long as the optimal broadcast schedule under the telephone model since both models are required to collect information from all the nodes to a root under the same constraints.

The multicommodity multicast problem is a generalization of both multicast and gossip problems, so combines the difficulty of both. Finding approximation algorithms for this problem under wireless model includes difficulties for multicommodity multicast under telephone model. As mentioned earlier, the best-known approximation ratio of $\tilde{O}(2^{\sqrt{\log n}})$ [9] for multicommodity multicast under telephone model is sub polynomial in n, whereas the best-known ratio for this problem under the wireless model is $\tilde{O}(n^{\frac{2}{3}})$ [6]. Thus, the very general multicommodity multicast problem under the wireless model appears to be quite hard to approximate. Our work significantly improve this ratio down to roughly $O(\sqrt{n})$.

2 General graphs

In this section, we study general multicommodity-multicast problem in arbitrary graphs. The best known approximation ratio so far is $\tilde{O}(n^{\frac{2}{3}})$ [6]. We improve this bound to roughly $\tilde{O}(\sqrt{n})$.

▶ **Theorem 2.** *Algorithm 3 is a polynomial algorithm that, given any $\varepsilon > 0$ and arbitrary graph with demand pairs, can find a multicommodity-multicast schedule in the wireless model with approximation ratio $O\left(n^{\frac{1}{2}+\varepsilon}\right)$.*

In the sequel, we assume that we are given the length L of the optimal rumor spreading schedule. Note that there is always a wireless multicast schedule of linear length in a connected graph by using any spanning tree, and traversing an Euler tour of the tree twice and using the edges in the traversal in order. Thus, we can guess the length L at the very beginning in the range $[1, 4(n-1)]$.

2.1 Algorithm

Our algorithm exploits two main observations that were introduced by Iglesias, Rajaraman, Ravi and Sundaram [6].

2.1.1 Idea 1

The first key idea is the following. If there are numerous sources that want to send to the same sink, we can satisfy **all demands** that originate in those sources. Algorithm 1 shows how this can be done. The analysis of the algorithm is summarized in Lemma 3.

▶ **Lemma 3** (Big in-demand [6]). *Given a vertex t that is a sink in demand pairs $(s_1, t), \ldots, (s_d, t)$, Algorithm 1 satisfies **all** demands from nodes $s_1, \ldots, s_d$ in $\tilde{O}(L)$ steps.*

Note that we are not only satisfying d demand pairs, but potentialy up to $d \cdot n$ demands, because every s_i could be a source of up to n demand pairs.

Algorithm 1 Algorithm for satisfying all demands in the "demand-neighbourhood" of a given vertex.

procedure InDemand(G, S, t)
Input: graph G, source nodes $S = \{s_1, \ldots, s_d\}$ with the same sink t.

Run TelephoneMulticast(G, t, S) to get schedule TM that spreads information from t to all nodes in S under telephone model.
Reverse schedule TM and run it on G to collect all information from $s_1, \ldots, s_d$ in t.
Run BFS(t) to spread information from t to all sinks that correspond to sources in S.
end procedure

2.1.2 Idea 2

The key observation here is the following. If we can find a lot of disjoint source-destination paths, we can satisfy all of them **in parallel**, and hence in at most the length of the longest path. We will try to greedily find as many paths of length at most L as possible from given sources to corresponding destinations. Algorithm 2 summarizes this procedure.

Algorithm 2 Algorithm for finding disjoint paths between sources and destinations.

procedure DISJOINTPATH(G, D, L)
Input: graph G, demand pairs $D = \{(s_1, t_1), \ldots, (s_p, t_p)\}$ and the length of the optimal algorithm L.

$P = \emptyset$
while there is a demand pair $(s, t) \in D$ such that the distance between s and t is at most L **do**
Add shortest path from s to t to P.
Delete all vertices in this path from G.
end while
Return P.
end procedure

2.1.3 Our idea

Iglesias, Rajaraman, Ravi and Sundaram [6] used both ideas and balanced parameters in them to achieve $\tilde{O}(n^{\frac{2}{3}})$-approximation. We will combine and generalize these ideas in a different way.

Idea 2.1.1 is useful when there is a vertex with high in-demand. More precisely, it allows us to reduce the number of distinct sources while there are a lot of demand pairs.

Idea 2.1.2 instead shows how we can either satisfy a lot of demand pairs in parallel or significantly reduce the number of distinct sources.

Our contribution comes from generalizing both ideas and combining them alternatively. We use the first idea to reduce the number of demand pairs, then we use the second idea to reduce the number of distinct sources. Now, given smaller number of sources we exploit the first idea again to reduce the number of demand pairs even further, etc. Algorithm 4 corresponds to the first idea of reducing the number of demand pairs given the number of distinct sources, whereas Algorithm 5 corresponds to the second idea of reducing the number of distinct sources given fixed number of demand pairs. These two algorithms call each other recursively. If we keep doing this infinitely long, we will achieve $\tilde{O}(\sqrt{n})$-approximation, so the main Algorithm 3 stops when it gets ε-close to it.

The analysis of the algorithm is presented in the main Lemma 7. The key idea in the analysis is that the first observation can improve the performance of the second one, and vice versa. So we alternatively apply each observation to improve the previous one.

2.2 Correctness and Complexity

▷ Claim 4. Algorithm 3 satisfies all given demands in the graph under wireless model rules.

First, note that the length L of the optimal schedule is at most $4(n-1)$, where n is the number of vertices, so one of the runs of $S(G, D, k, L)$ will use the correct guess of L and hence will satisfy all demands as long as Algorithm 4 works properly.

▷ Claim 5. Algorithms 4 and 5 satisfies all given demands in the graph under wireless model rules.

Proof. First, note that algorithms 4 and 5 call each other alternatively, but every second time, the parameter k (which keeps track of the number of iterations) decreases by 1, so it will eventually reach 0. Also, note that every time any of these algorithms modify the set of demand pairs D, they correctly satisfy or split demands. After k reaches 0, Algorithm 4 just runs BFS from every source node, which clearly satisfies all demands. ◁

Algorithm 3 Algorithm for rumor spreading in the general graph.

procedure GENERALMM(G, D, ε)
Input: unweighted graph G, demand pairs $D = \{(s_1, t_1), \ldots, (s_p, t_p)\}$ and precision $\varepsilon > 0$.

Take integer $k = \lceil \frac{1}{4\varepsilon} \rceil$.
for $L = 1, 2, \ldots, 4(n-1)$ **do** ▷ guess for the length of the optimal schedule
if $S(G, D, k, L)$ produces a shorter schedule **then**
Save $S(G, D, k, L)$ as current answer in BestSchedule.
end if
end for
Return BestSchedule.
end procedure

Algorithm 4 Algorithm for general graphs based on the number of sources.

procedure S(G, D, k, L)
Input: graph G, demand pairs $D = \{(s_1, t_1), \ldots, (s_p, t_p)\}$, iteration number k and the length of the optimal schedule L.

if $k = 0$ **then** ▷ condition to exit the recursion
for every distinct source s **do**
Run BFS(s). ▷ to satisfy all demands in D that start in s
end for
Exit.
end if

Compute s to be the number of vertices in the graph that are sources in at least one demand pair in D.
Take $\alpha = \frac{1+k \log_s n}{2k+1}$.
while there is a vertex t with at least $s^{1-\alpha}$ demands going into it **do**
Compute S_t to be those vertices that needs to send information to t. So $S_t = \{s \mid (s, t) \in D\}$.
Run INDEMAND(G, S_t, t) to satisfy **all demands** that originate in S_t.
Delete all demands that originate in S_t from D.
end while
Run $P(G, D, k-1, L)$.
end procedure

▷ Claim 6. Algorithm 3 runs in polynomial time($n, \frac{1}{\varepsilon}$).

Proof. First, note that Algorithm 1 runs in polynomial time, because TELEPHONEMULTICAST as well as BFS run in polynomial time. Since we can find shortest path between any two nodes in a graph is polynomial time, Greedy Algorithm 2 runs in polynomial time as well since we decrease the number of vertices in the graph at each iteration.

Then, Algorithm 3 runs Algorithm 4 $4(n-1)$ times, so it is polynomial time as long as Algorithm 4 is.

Algorithms 4 and 5 call each other alternatively, at most $k = \lceil \frac{1}{4\varepsilon} \rceil = O(\frac{1}{\varepsilon})$ times each. So it is enough to show that both of these algorithms each run in polynomial time.

Algorithm 5 Algorithm for general graph based on the number of demand pairs.

procedure P(G, D, k, L)

Input: graph G, demand pairs $D = \{(s_1, t_1), \ldots, (s_p, t_p)\}$ and iteration number k and the length of the optimal algorithm L.

Take $\beta = \frac{1+k\log_p n}{2k+2}$.

while the number of disjoint paths $d = |\textsc{DisjointPath}(G, D, L)|$ is at least $p^{1-\beta}$ **do**

Satisfy all d demands in parallel following paths in DisjointPath(G, D, L).

Delete all those demands from D.

end while

Denote $\textsc{DisjointPath}(G, D, L)$ to be the disjoint paths for demand pairs $\{(s_1, t_1), \ldots, (s_d, t_d)\}$.

Let a new graph G' be G with added complete binary tree with leaves $t_1, \ldots, t_d$ and root r.

Compute S to be all vertices in G that are sources in at least one demand pair in D.

TM $= \textsc{TelephoneMulticast}(G, r, S)$. ▷ TM spreads information from the root r to all sources in D under telephone model

Reverse schedule TM and run it on G. ▷ It will collect all information from sources in D into $t_1, \ldots, t_d$

$D_{new} = \emptyset$

for $(s, t) \in D$ **do**

Find t_i among $t_1, \ldots, t_d$ that has information from source s.

Add demand pair (t_i, t) into D_{new}.

end for

Run $S(G, D_{new}, k, L)$.

end procedure

Algorithms 4 runs Algorithm 1 at most $\frac{s}{s^{1-\alpha}} = s^\alpha$ times. Given that $\alpha = \frac{1+k\log_s n}{2k+1}$ we have that $s^\alpha = \sqrt[2k+1]{sn^k} \leq \sqrt[2k+1]{n^{k+1}} \leq n$. As we have already noted, Algorithm 1 also runs in poly-time, so overall 4 is poly-time in terms of n and $\frac{1}{\varepsilon}$.

Algorithms 5 runs Algorithm 2 at most $\frac{p}{p^{1-\beta}} = p^\beta$ times. Given that $\beta = \frac{1+k\log_p n}{2k+2}$ we have that $p^\beta = \sqrt[2k+2]{pn^k} \leq \sqrt[2k+2]{n^{k+1}} \leq n$. As we have already noted, Algorithm 2 also runs in poly-time. So overall Algorithm 4 is polynomial time in terms of n and $\frac{1}{\varepsilon}$. ◁

2.3 Analysis

First, assume that for the given graph and demand pairs the optimal schedule takes L time steps. The main Theorem 2 will follow from the following key lemma:

▶ **Lemma 7.** *For every $k \in \mathbb{N}_0$ the following holds:*

(S_k) *If the number of distinct sources is s, then there is a $\tilde{O}(\sqrt[2k+1]{sn^k})$-approximation schedule.*

(P_k) *If the total number of demand pairs is p, then there is a $\tilde{O}(\sqrt[2k+2]{pn^k})$-approximation schedule.*

Note that $s \leq n$, so for every $k \in \mathbb{N}_0$ from S_k we can have $\tilde{O}(\sqrt[2k+1]{sn^k}) = \tilde{O}(n^{\frac{k+1}{2k+1}})$ approximation schedule. As $k \to \infty$, $\frac{k+1}{2k+1} \to \frac{1}{2}$, and hence we can find a schedule that has approximation ratio $O(n^{\frac{1}{2}+\varepsilon})$ for any positive ε. Thus, Lemma 7 implies the main Theorem 2.

To prove the main lemma, we use an important observation that handles aggregated messages with a few sinks.

▶ **Lemma 8.** *If there is only* one sink *in multicommodity multicast under wireless model, then there is an* $O(\frac{\log p}{\log \log p})$*-approximation algorithm for satisfying all demands, where* p *is the number of sources.*

Proof. The proof is based on the fact that collecting information from several sources to *one* sink t under wireless model is equivalent to multicasting information from t to those sources under telephone model (in reverse).

Note that because we are collecting information at one node, every piece of information travels on the path (the node never sends information to several neighbours at the same time) and so every step of the schedule is a matching - in other words, we have never used the power of wireless model over telephone model. Hence, collecting information at one node under wireless model is equivalent to collecting under telephone model, which is further equivalent to multicast under telephone model by reversing the schedule.

Using the known approximation result for multicasting under the telephone model given in Theorem 1, we can find a multicast schedule of length at most $O(L \cdot \frac{\log p}{\log \log p})$. Reversing it leads to a schedule that collects all information into t in time $O(L \cdot \frac{\log p}{\log \log p})$. ◀

We can now prove our initial lemma about satisfying all the demands of sources corresponding to a single sink.

Proof of the Lemma 3. We will produce a schedule that will satisfy all demands from nodes $s_1, \ldots, s_d$ in $\tilde{O}(L)$ steps. It will go in two stages:

1. Collect all information from $s_1, \ldots, s_d$ in t using Lemma 8.
2. Send it all together from t to all sinks of $s_1, \ldots, s_d$ using BFS.

Note that the second step could be done in $O(L)$ by BFS from t, because the distance from t to any sink of s_i is at most the distance from t to s_i plus the distance from s_i to sink, which is at most $L + L = 2L$. ◀

Proof of key Lemma 7. We prove this by induction on k. Moreover, the structure will be as follows.

1. $S_k \Rightarrow P_k$ for every $k \geq 0$.
2. $P_k \Rightarrow S_{k+1}$ for every $k \geq 0$.

Base case. It is enough to prove base case S_0, in other words that it is possible to satisfy all demands with approximation ratio $O(s)$. Note that if we fix an arbitrary source v, we can satisfy all its demands in $O(L)$ steps by BFS starting at v (exploiting the power of wireless model where we can send to several neighbours at the same time). By repeating this over the s sources, it is possible to satisfy all demands in $O(L \cdot s)$ steps.

Induction step. Fix an arbitrary $k \geq 0$.

1. We want to prove that $S_k \Rightarrow P_k$.
   To recall, this means that: if there is a $\tilde{O}(\sqrt[2k+1]{sn^k})$ approximation schedule for s sources, then there is a $\tilde{O}(\sqrt[2k+2]{pn^k})$ approximation schedule for p demand pairs.
   Assume that we are given p demand pairs.
   Forming greedy paths: Start with an arbitrary source and try to find a path of length at most L to one of its destinations. If found, delete all vertices in this path from the graph and repeat. This corresponds to Algorithm 2.

After this stage we have some maximal number d of disjoint paths $(s_1, t_1), \ldots, (s_d, t_d)$. Note that because they are disjoint we can satisfy all of them in parallel in L steps.
Let $\beta \in [0, 1]$ be a constant we will set afterwards. The idea of Algorithm 5 is the following.

a. **If the number of greedy paths d is at least $p^{1-\beta}$**, then satisfy those demands in parallel and repeat.
b. **If the number of greedy paths d is less than $p^{1-\beta}$**, we will reroute/collect all information at $t_1, \ldots, t_d$ using Lemma 9 and finish everything in one shot by applying S_k to it.

We cannot repeat the first case more that $\frac{p}{p^{1-\beta}} = p^\beta$ times, and so the total number of rounds in the schedule is at most $\tilde{O}(L \cdot p^\beta)$.
In the second case, there are $d \leq p^{1-\beta}$ sources now, so by applying S_k we get that we can send all information in $\tilde{O}(L \cdot \sqrt[2k+1]{p^{1-\beta} n^k})$. Combining the two cases gives approximation ratio $\tilde{O}(p^\beta + \sqrt[2k+1]{p^{1-\beta} n^k})$. Minimizing this leads to $\beta = \frac{1+k \log_p n}{2k+2}$ and approximation ratio $\tilde{O}(\sqrt[2k+2]{p n^k})$, which is exactly what we wanted.
Technical note: we need to ensure that $\beta \in [0, 1]$. Clearly $\beta \geq 0$. Now, $k = 0$ leads to $\beta = \frac{1}{2}$, giving a $\sqrt{p}$ approximation ratio. For $k \geq 1$ we have that $p \geq n \Rightarrow \beta \leq 1$, but in the case $p < n$ the approximation ratio $\sqrt{p}$ is better than $\tilde{O}(\sqrt[2k+2]{p n^k})$ for any k.
The only thing left is to prove the following lemma.

▶ **Lemma 9** ([6]). *If there are d greedy paths $(s_1, t_1), \ldots, (s_d, t_d)$, we can collect all information at one of the $t_1, \ldots, t_d$ in $\tilde{O}(L)$ time.*

Figure 1 gives an example illustration of the Lemma 8.

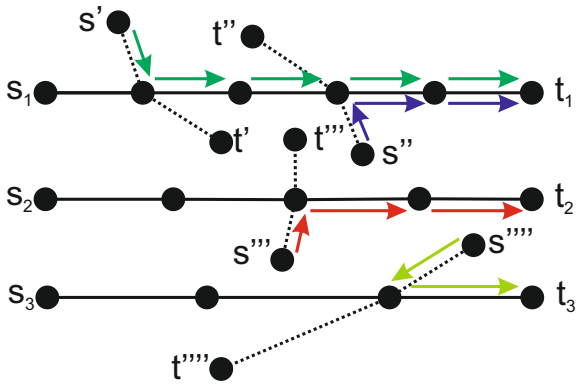

Figure 1 Collecting information from all sources into $t_1, \ldots, t_k$.

Proof. For the simplicity of the argument (and to use results about broadcast time, which is the same as collecting information at one node), we will add to our graph a dummy complete binary tree with leaves $t_1, \ldots, t_d$ and root r and our goal will be to collect all information at r in $\tilde{O}(L)$ time. Obviously, if we can do this, then we also will be able to collect all information at one of the $t_1, \ldots, t_d$ in the same amount of time.
Consider an arbitrary source s and pick a fixed sink t for it (only one). Consider the path between $s \to t$ in the optimal schedule. This path has length at most L and hence it should intersect with path $s_i \to t_i$ for some i (otherwise we can find one more disjoint path of length at most L). This means that if we run the optimal schedule, then nodes on the greedy paths will contain information from all sources. Hence if we run it for L more steps by transmitting information only along greedy paths, information from all

sources will eventually be at one of $t_1, \ldots, t_d$. The dummy binary tree has depth at most $\log d \leq \log n$, so if we run the schedule for $2 \log n$ more rounds (two parallel rounds per level), the root r will contain all the information.
This means that it is possible to collect all information in r in at most $2L + 2 \log n$ steps. Using Lemma 8 we can find schedule that collects all information into r in at most $O((2L + 2 \log n) \frac{\log n}{\log \log n})) = \tilde{O}(L)$ steps. ◀

2. We want to prove that $P_k \Rightarrow S_{k+1}$.
Here we will expoit Lemma 3. Note that we are not just satisfying d demand pairs, but potentialy up to $d \cdot n$ demands, because every source s_i could be in up to n demand pairs. Also, by the base case we know that this could be done in $O(dL)$ steps, but this lemma gives a much better bound.
Now, assume that we have s sources, and let $\alpha \in [0, 1]$ be number that we will define afterwards. The main idea of Algorithm 4 is the following.
   a. **If there is a vertex with at least $s^{1-\alpha}$ demands** going into it, then satisfy those demands by Lemma 3 and repeat.
   b. **If every vertex has less than $s^{1-\alpha}$ demands** going into it, then the total number p of demand pairs is small and we will finish in one shot by applying P_k.

   In the first case we can satisfy all demands in $\tilde{O}(L)$ steps by the lemma, and also we will repeat this at most $\frac{s}{s^{1-\alpha}} = s^\alpha$ times, so the total number of steps is $\tilde{O}(L \cdot s^\alpha)$.
   In the second case, if every vertex has less than $s^{1-\alpha}$ demands going into it, then the total number of demands is $p \leq s^{1-\alpha} n$. Applying P_k gives that we can find a schedule to satisfy all demands in $\tilde{O}(L \cdot \sqrt[2k+2]{s^{1-\alpha} n^{k+1}})$.
   Combining the two yields to approximation ratio $\tilde{O}(s^\alpha + \sqrt[2k+2]{s^{1-\alpha} n^{k+1}})$. By minimizing this we get $\alpha = \frac{1+(k+1)\log_s n}{2k+3}$ for the call to $S(G, D, k+1, L)$ and corresponding approximation ratio $\tilde{O}(\sqrt[2k+3]{s n^{k+1}})$, which is what we wanted.
   Technical note: we need to ensure that $\alpha \in [0, 1]$. Clearly $\alpha \geq 0$. Note that $\alpha \leq 1$ if and only if $s \geq \sqrt{n}$, but in the case $s < \sqrt{n}$ the approximation ratio $O(s)$ in the base case is better than $\tilde{O}(\sqrt[2k+3]{s n^{k+1}})$ for any k. ◀

3 Reduction to packet routing

In this section we draw a connection between multicommodity multicast problem and extensively studied store-and-forward packet routing problem. We further derive an algorithm that satisfies all demands in O(OPT + number of demand pair) for multicommodity multicast under wireless model using approximation results from packet routing problem. Note that if the number of given demand pairs is small or the optimal schedule is long, this gives a better approximation result compared to the one we have shown in Section 2.

▶ **Theorem 10.** *There is a polynomial time algorithm that finds a schedule of length $O(L+p)$ for multicommodity multicast problem with p demand pairs under wireless model.*

3.1 Packet routing problem

We are going to relate wireless multicommodity multicast and packet routing problem.

▶ **Definition 11** (Store-and-forward packet routing problem). *Given an arbitrary (directed) graph G and source-sink pairs of vertices $(s_1, t_1), \ldots, (s_p, t_p)$, where each source s_i has a unique packet, the problem is to find a schedule that transfers packets from every source s_i to every destination t_i in a minimum amount of steps, so that no two packets can traverse the same edge at the same unit of time.*

Note that under wireless model we are allowed to send several pieces of information at once along one edge, which is not the case for packet routing problem. On the other hand, packet routing problem has no other restrictions. This implies that a node can: (i) receive several packets at the same time (ii) send several packets as long as they use different edges (iii) send and receive at the same time.

The analysis of packet routing problem involves a trade-off between the "dilation" parameter d (maximum path length) and the "congestion" criterion c (maximum number of paths using any edge). Note that both c and d are lower bounds on the length of the optimal schedule for packet routing. Since the paths for the packets are not specified a priori, note that the problem also first involves finding such paths minimizing the sum of the dilation and congestion and then constructing an actual (offline centralized) schedule that completes in about this many rounds. Srinivasan and Teo [12] do just that and show how to construct a schedule of length $O(c + d)$, which achieves constant approximation.

▶ **Theorem 12** ([12]). *There is a polynomial time algorithm that finds a schedule for store-and-forward packet routing problem and gives a constant approximation.*

3.2 Reduction from wireless to packet routing

We prove Theorem 10 by reducing multicommodity multicast under the wireless model to packet routing problem and using Theorem 12. There are two superior properties in the packet routing problem: (i) a node can receive from several sources per time step; (ii) a node can send and receive at the same time step.

First, we will show how to eliminate (i). The number of incoming messages is a lower bound under wireless model (because a node can receive only from one source), whereas it is not the case in packet routing problem. Given an instance for wireless multicommodity multicast, we are going to modify it to create another instance for packet routing and use its schedule to design a fast wireless schedule that satisfies given demand pairs. Formally speaking, given an arbitrary undirected graph G we construct *directed* graph G' in the following way. For every node $v \in G$ we create two nodes v_{in} and v_{out} in G' and put a directed edge $v_{in} \to v_{out}$. Also, for every edge $(u, v) \in G$ we put directed edges $u_{out} \to v_{in}$ and $v_{out} \to u_{in}$ in G'. Then, every demand pair (s, t) in G correspond to demand pair (s_{in}, t_{out}) in G'. We split every original vertex into "only receiving" and "only sending" part. In this way the inbound restriction of v in G is translated into congestion restriction of edge $v_{in} \to v_{out}$ in G'.

We still need to address the second difference (ii) between packet routing and wireless. But for now we relax this condition and show how to transform wireless multicommodity multicast to packet routing allowing nodes to send and receive at the same time under wireless model.

▶ **Definition 13.** *The **Upgraded Wireless model** is the relaxation of the wireless model in which a node is allowed to send and receive information at the same time.*

▶ **Lemma 14** (Packet to information routing). *Given a schedule for packet routing problem on the graph G', it is possible to transform it in polynomial time to a schedule of the same length on the graph G under the upgraded wireless model.*

Proof. First, given a packet routing schedule R on G' we modify R to schedule R' such that the following is true: (i) R' sends information through the same path as R (ii) R' finishes in the same amount of steps as R (iii) the graph G' does not contain any nodes that receive from more than one source at any time step. Then we show how given R' one can simply construct an upgraded wireless schedule of the same length on G.

Let's fix an arbitrary time step in R. Every node that sends to several destinations, or sends and receives at the same time can still do so under upgraded wireless model. The only issue is that a node cannot receive from several sources. Note that v_{out} can receive only from v_{in}, so we will show how to alter the schedule R so that every node v_{in} receives a packet from at most one neighbor. The main idea is that instead of accumulating packets at 'receiving' vertices v_{in}, we will accumulate information at 'sending' vertices u_{out}. More precisely, if vertex u_{out} sends packet p_i to v_{in} at some point t in time in R, then the new schedule R' does the following: vertex u_{out} will hold packet p_i and send it to v_{in} *right before* the time step when v_{in} sends p_i to v_{out} in R. So we send information only when needed, otherwise hold it in the previous node.

Note that in R' there is no vertex that receives from more than one node at a time. Moreover, the time at which every packet arrives to any vertex v_{out} under R is the same as in R'. This follows from the fact that whenever some vertex v_{out} needs a packet p_i from v_{in}, vertex v_{in} would have already collected packet p_i by construction of R'.

We are left to show that packet routing R' on graph G' could be transformed into upgraded wireless schedule W of the same length on graph G. Every time vertex u_{out} sends a packet p_i to v_{in}, let W send information i from u to v. Indeed, W has the same number of steps as R' and satisfies all demands, because it just mirrors packet schedule of R'. Moreover, W is a valid upgraded wireless schedule, since in R' every node receives from at most one node at every time step. ◀

▶ **Lemma 15.** *Given a schedule under upgraded wireless model it is possible to transform it in polynomial time to a schedule under wireless model, which has at most* 3 *times as many steps.*

Proof. Fix a time step and let S to be the directed graph of information flow of the given schedule under upgraded wireless model at this specific time step. So there is a directed edge $u \to v$ in S if and only if node u sends information to node v during this time step. We will show how to send all information that S does using wireless model in three time steps.

Note that the in-degree of every node in S is at most 1, because under upgraded wireless model a node still cannot receive from more than one source per time step. Each connected component in any directed graph with in-degree bounded by 1 has a simple characterization. It is a directed cycle with outbound trees hanging from it. More precisely, it is a union of the cycle $v_1 \to v_2 \to \ldots \to v_k \to v_1$ with several trees each rooted at one of v_i's, where every edge in a tree is directed out from the root. Note that it is enough to show how to transfer information in one connected component of schedule S using the wireless model and then run this in parallel across different connected components.

Deleting an edge $v_k \to v_1$ from the cycle (any other edge works as well) leads us to a tree rooted at v_1, where every edge is directed out from the root. We show how to send all information along the edges in such a tree. We apply the following trick: let the *level* of the vertex be its distance to the root of the tree. Now, call the vertex *even* if its level is even and *odd* if its level is odd. First, all odd nodes that need to send information will do so (and so even ones will be able to receive). Then, all even nodes will be allowed to send information (and so odd ones will receive). In this way every node is either receiving or sending information. Finally, we send information along the deleted edge $v_k \to v_1$ from the cycle. Hence instead of sending all information in one time step, we did it in three. This blows up the total number of time steps by a factor of 3. ◀

▶ Remark. The result in Lemma 15 is sharp, for instance when underlying demand graph S is a directed cycle on 3 vertices. Under thre upgraded wireless model information could be sent in 1 step, whereas under the usual wireless model it requires 3 steps.

We are going to combine these results to prove Theorem 10. Given multicommodity multicast problem under wireless model on graph G with p demand pairs, we construct graph G' as described. Also, for every demand pair (s_i, t_i) in G we create a separate packet that originates at s_i and needs to be delivered to t_i. Note that even if there is one piece of information that originates at source s and needs to be sent to different sinks $t_1, \ldots, t_k$, we create *different* packets $p_1, \ldots, p_k$ for each of these demands. If we consider optimal wireless schedule of length L on G and trace the path of every packet p_i, then the length of every such path is at most L, and on every edge the number of paths that uses that edge is at most p (because there are p paths in total). Hence, dilation $d \leq L$ and congestion $c \leq p$. We do not know the optimal wireless schedule, but both c and d are lower bound on the length of optimal packet routing schedule, and hence $\frac{c+d}{2}$ is also a lower bound. Theorem 12 gives a constant-approximation schedule for packet routing, and hence this schedule satisfy all demands in $O(c + d) = O(L + p)$ time steps. Then we exploit Lemma 14 and Lemma 15 to transform it into a $O(L + p)$ schedule for multicommodity multicast problem under the wireless model, which proves Theorem 10.

4 Rumor spreading in trees

In this section we present a simple polynomial algorithm Tree that gives a constant approximation for wireless rumor spreading in trees, which improves the best known $O(\frac{\log n}{\log \log n})$-approximation ratio for n-node trees.

▶ **Theorem 16.** *Algorithm* Tree *runs in polynomial time and gives* 3*-approximation for multicommodity multicast problem in a tree under wireless model.*

The algorithm consists of two parts. We use the same idea as Iglesias, Rajaraman, Ravi and Sundaram [6], by rooting the tree and splitting rumor spreading into sending all information *up the tree* to ancestors, and then sending it *down the tree* to descendants.

4.1 Algorithm

Algorithm 6 Algorithm for rumor spreading in the tree.

procedure Tree(T, D)
  Input: Tree T and demand pairs $D = \{(s_1, t_1), \ldots, (s_p, t_p)\}$.
  Pick an arbitrary vertex r and root the tree T at r.
  $D_{up} := \emptyset$
  $D_{down} := \emptyset$
  for $(s, t) \in D$ **do**
    Find the least common ancestor $lca(s, t)$ of s and t.
    Add demand pair $(s, lca(s, t))$ to D_{up}.
    Add demand pair $(lca(s, t), t)$ to D_{down}.
  end for
  Run TreeUp(T, D_{up}).
  Run TreeDown(T, D_{down}).
end procedure

In order to send information *up* the tree, algorithm TreeUp uses a greedy approach to send information that should go the furthest first. This strategy satisfies all demand pairs in at most 2·OPT time steps, where OPT is the length of the optimal schedule. Lemma 19 provides an analysis of the algorithm.

Algorithm 7 Algorithm for rumor spreading, when all demands go up the tree.

procedure TreeUp(T, D)
Input: Rooted tree T and demand pairs $D = \{(s_1, t_1), \ldots, (s_p, t_p)\}$, where t_i is an ancestor of s_i in the tree, for every i.
$D_{clean} := \emptyset$ ▷ This will contain a subset of demand pairs D with non-repeating sources.
for every vertex $s \in G$ that is a source in D **do**
Among all demand pairs in D with source s consider the one, which has the longest distance between s and corresponding sink in the tree.
Add this pair in D_{clean}.
end for
Set the depth of every node v, $d(v)$ to be the length of the path from v to the root r.
Set current number of steps $i := 0$.
while not all demands in D_{clean} are satisfied **do**
for $v \in G$ **do**
if $d(v)$ has same parity as i **then**
Find child u of v with the piece of information that at the current state needs to travel the longest distance up in the tree to its sink.
Send all information u has up to v.
end if
end for
$i = i + 1$
end while
end procedure

The algorithm TreeDown sends information *down* the tree very fast, using two observations. First, it exploits the fact that under wireless model information could be sent to several neighbours simultaneously. Second, we note that any two nodes in a tree have disjoint set of children. We show that this strategy shows satisfies all demands in at most OPT+1 steps. Lemma 20 gives this analysis of algorithm TreeDown 8.

Algorithm 8 Algorithm for rumor spreading, when all demands go down the tree.

procedure TreeDown(T, D)
Input: Rooted tree T and demand pairs $D = \{(s_1, t_1), \ldots, (s_p, t_p)\}$, where t_i is a descendant of s_i in the tree, for every i.
Set the depth of every node v, $d(v)$ to be the length of the path from v to the root r.
Set current number of steps $i := 0$.
while not all demands are satisfied **do**
for $v \in G$ **do**
if $d(v)$ has same parity as i **then**
send all information v has to all its children in the tree.
end if
end for
$i = i + 1$
end while
end procedure

Combining these two algorithms, we get the overall algorithm Tree for wireless multicommodity multicast in the tree.

4.2 Correctness and Complexity

▷ Claim 17. Algorithm Tree 6 satisfies all given demands in the tree under wireless model rules.

Proof. First, it is clear that if we first send information from source s to $lca(s,t)$ for every demand pair (s,t), and then from $lca(s,t)$ to t for all (s,t), then we would have sent information from every s to corresponding t.

Now, we need to show that algorithms TreeUp 7 and TreeDown 8 correctly send information up and down the tree respectively. First, let's show that both algorithms obey wireless rules. Note that in both algorithms only nodes with the same parity of depth can send information, and so node with opposite parity of depth will receive information, hence there is no vertex that is trying to send and receive information at the same time.

Next, we show that none of the nodes is trying to receive information from more than one vertex. Every vertex v in algorithm TreeUp 7 receives from only one if its children (if any). For algorithm TreeDown 8, observe that nodes with even (or odd) depth are pairwise not connected, and so sets of theirs children are disjoint.

Finally, note that both algorithms will run until all demands are satisfied. Since at every step at least one piece of information becomes closer to the destination, it should satisfy all demands in finite number of steps. ◁

▷ Claim 18. Algorithm Tree runs in polynomial time.

Proof. It is enough to show that both algorithm TreeUp 7 and algorithm TreeDown 8 run in polynomial time. Note that since at every step at least one piece of information becomes closer to the destination, the number of while loops is at most (number of demand pairs)·(maximum distance in the tree), which is polynomial in the number of vertices in the tree. In algorithm TreeUp 7 at each iteration of the while loop, we also need to loop through all children of v and find the one that needs to send information the furthest. This could be done in poly-time as well, because the number of children, pieces of information and diameter of the tree are all polynomial in n. ◁

We show that algorithm Tree outputs a rumor spreading schedule that is at most 3 times longer than the optimal one. Let us denote the length of the optimal schedule to be L.

It is enough to analyze approximation ratios of both algorithm TreeUp and algorithm TreeDown. We show that the first one gives a 2-approximation and the second one is almost optimal.

▶ **Lemma 19.** *Algorithm* TreeUp *7 produces the schedule that satisfies all demands in at most* $2L$ *steps.*

▶ **Lemma 20** ([6])**.** *Algorithm* TreeDown *8 produces the schedule that satisfies all demands in at most* $L+1$ *steps.*

These two lemmas clearly imply that algorithm Tree 6 has approximation ratio 3.

Proof of Lemma 19. Note that for every source s, all of its sinks could only be the vertices along the path from s to the root. Observe that for each $s-t$ pair the path from s to t is unique and hence we can also define a distance from s to t.

If s has to send to several sinks, we can leave only the furthest sink and remove others - because the tree path from s to the furthest sink also passes through closer sinks. So now we assume that every source has exactly one sink, which corresponds to the new set of demand pairs D_{clear}.

First, let us assume that the vertex is allowed to send and receive at the same time, but still is not allowed to receive from more than one source. Under this assumption, algorithm TreeUp 7 (after removing the line *"If $d(v)$ has same parity as i"*) achieves the optimal spreading time! Figure 2 illustrates the execution of this modified version of algorithm TreeUp 7.

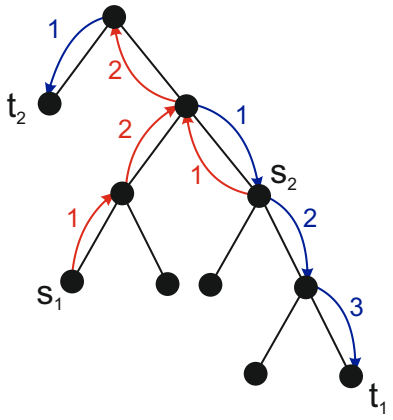

Figure 2 Illustration of precursor to algorithm Tree 6.

Recall that the algorithm is the following: if node v contains some information, it will try to send it up. So if no other vertices are trying to send information to the parent of v, then v is allowed to send information and it is obviously *the best possible move*.

▶ **Definition 21** (The best/optimal move). *Given a multicommodity-multicast problem, the move is called optimal if there exists a subsequent sequence of moves that will achieve optimal rumor spreading time.*

Now, imagine that at the specific point in time there are several nodes $v_1, \ldots, v_k$ who want to send to their common parent t.

▶ **Definition 22** (The priority of the vertex). *Given a vertex v that has information from sources $s_1, \ldots, s_k$, define the priority of v at this particular state of the system to be the maximum distance from v to all sinks $t_1, \ldots, t_k$ that correspond to sources $s_1, \ldots, s_k$.*

The rule will be that vertices with larger priorities always send first. More precisely, if there are several nodes $v_1, \ldots, v_k$ who want to send information to their common parent t at this point in time, then pick the vertex v_i with the largest priority and allow it to propagate information to t (we will break any ties arbitrarily).

We argue that this is an optimal move. Consider the optimal algorithm and the first moment in time when the state is different from the state in our greedy algorithm. We will show that optimal algorithm could be modified, so that it still finishes rumor spreading in the same number of steps, but also does the same move as our greedy algorithm.

Assume at this state vertices $v_1, \ldots, v_k$ want to send to their parent t. One of the vertices v_i should be allowed to send info to t since one more piece of information will move closer to the sink this way. Consider what the optimal algorithm will do. Say it allows vertex v_j to send information to its parent. If $v_i = v_j$, then we are done (meaning our current move is optimal). If not, then consider a different algorithm, where we swap v_i in place of v_j, and move this step where v_j sends information to its parent to the next time when v_i is scheduled to do so in the optimal algorithm. We claim that this modified algorithm is still optimal.

To see this, consider the optimal algorithm again. Both vertices v_i and v_j have common path to the top (except the very first edge to their parent, which we will ignore). Moreover, because the priority of v_i is at least the priority of v_j, then the demand from v_j is a subpath of the demand for v_i (remember we associate demand with both the source-sink pair and the path between them). Because v_j goes first and takes the same route as v_i, but v_i goes further, we have that at some point during the optimal schedule the current information from vertices v_i and v_j will be at the same vertex (somewhere along the demand path). Hence at the end of the optimal schedule both sinks that correspond to v_i and v_j will contain the current information from both v_i, v_j. And so if we swap v_i and v_j in our algorithm, after the same number of time steps as in the optimal schedule again, both sinks that correspond to v_i, v_j will contain current information from both v_i, v_j. Hence our modified algorithm is optimal as well.

To sum up, this proves that our greedy algorithm which allows vertices with higher priority to send first is optimal.

To finish the proof we need to get rid of the assumption that a node can send and receive information at the same time. For this, we use the following idea: let the *level* of the vertex be its distance to the root of the tree. Now, call the vertex *even* if its level is even and *odd* if its level is odd. We will alternate between even/odd levels to send/receive. First, all odd nodes that need to send information will do so (and so even ones will be able to receive). Then, all even nodes will be allowed to send information (and so odd ones will receive). This is then repeated. In this way every node is either receiving or sending information at every step. Clearly, we have increased the number of time step by at most factor 2. Therefore our algorithm will finish in at most $2L$ steps. ◀

Proof of Lemma 20. First, note that the distance between any source and sink is at most L, because in the optimal schedule all distances are at most L. But we cannot start greedily send information down, because a vertex cannot both send and receive information at the same time.

Hence we will use the same idea again - vertices with even/odd levels will alternate to send/receive. Note that our algorithm TreeDown 8 does the following: first, all even nodes that need to send information will do so (and so odd ones will be able to receive). Then, all odd nodes will be allowed to send information (and so even ones will receive). This is repeated as before.

Consider a fixed source. If it is even, then it will start sending information down right away, if it is odd, there will be a delay of one step. But after the vertex started sending information, there will be no delays! For example, if the vertex is even, then it sends information to the odd one, and at the next iteration all odd vertices are allowed to send information. Hence, it will take at most $L + 1$ steps to deliver all information. ◀

4.3 Optimal Telephone Broadcast in a Tree

▶ **Corollary 23** (Optimal broadcast in the tree under telephone model)**.** *There is a* ***simple*** *optimal schedule for broadcasting in the tree from any root under telephone model.*

Note that it is well-known that the optimal schedule for broadcast in the tree under telephone model can be found in polynomial time. For instance, one can find such a schedule using dynamic programming. But this approach only gives an implicit algorithm to find such a schedule. We will present a very **simple explicit optimal schedule** algorithm TreeBroadcast 9.

Algorithm 9 Algorithm for optimal broadcast in the tree under telephone model.

procedure TreeBroadcast(T)
  Input: rooted tree T.

  OptimalSchedule $= \emptyset$
  while T is not empty **do**
    Matching $= \emptyset$
    for every vertex $v \in T$ **do**
      If v is not connected to any leaf, then continue to another v.
      Among all children of v, pick an arbitrary leaf u.
      Add edge $u \to v$ to Matching.
    end for
    Add Matching to OptimalSchedule.
    Delete Matching from T and any isolated vertices.
  end while

  Return reversed OptimalSchedule
end procedure

▷ Claim 24. Algorithm TreeBroadcast 9 outputs the optimal broadcast schedule in the tree.

Proof. Every broadcast schedule that sends information from the root to all nodes is equivalent to a schedule that sends information from the root to all leaves. This is further equivalent to a schedule that collects information from all leaves to the root - by reversing the direction of information flow and hence the schedule. We will present such a schedule.

Assume we are given that each leaf has a unique piece of information that it wants to transmit to the root. The schedule is very simple: every leaf that has some information (one or several pieces collected from other nodes) tries to send it to its parent. After the leaf successfully propagates the information up, we delete this vertex from the tree. If there are several leaves who try to send to the same parent, then choose one arbitrarily.

This is an optimal move because in each conflict when several leaves are trying to send to the parent, the priority of each of these conflicting leaves is the same (since their distance to the root is the same), so the argument is exactly the same as in the analysis of Algorithm TreeUp.

The only additional point here is that we do not allow for non-leaf vertices to send information, even though they might have some information available. This is because if there is an internal node v, such that in its subtree there are leaves with information that they haven't sent up to v yet, there is no need in an optimal scheme for v to try to send available information up, since it can wait until the very last information in its subtree will reach v. More formally, the priority of the node v is less than the priority of a leaf in a subtree of v, and hence we should give a priority to the leaf (by the same argument as in that of Algorithm TreeUp). This rule makes sure that there is no vertex that is trying to send and receive information and the same time. ◁

5 Conclusion

We have studied well-known rumor spreading problem under the wireless model. We designed approximation algorithms for the most general multicommodity multicast set-up that improve approximation ratios both for general graphs and when the underlying graph is a tree. For

general graphs we improve approximation ratio for this problem from $\tilde{O}(n^{\frac{2}{3}})$ to $\tilde{O}(n^{\frac{1}{2}+\varepsilon})$ on n-node graphs. We also design an algorithm that satisfies p demand pairs in $O(\mathrm{OPT}+p)$ steps, by reducing it to the well-studied packet routing problem. When underlying graph is an n-node tree, we improved approximation ratio from $O(\frac{\log n}{\log\log n})$ to a constant 3-approximation. A consequence of our algorithm is the simple constructive rule for optimal broadcasting in a tree under a widely studied telephone model.

References

1. Marek Chrobak, Leszek Gaasieniec, and Wojciech Rytter. Fast broadcasting and gossiping in radio networks. *Journal of Algorithms*, pages 177–189, 2002. doi:10.1016/S0196-6774(02)00004-4.
2. Michael Elkin and Guy Kortsarz. Sublogarithmic Approximation for Telephone Multiast. *SODA Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 76–85, 2003.
3. Arthur M. Farley. Broadcast Time in Communication Networks. *SIAM J. Appl. Math.*, pages 385–390, 1980. doi:10.1137/0139032.
4. András Hajnal, Eric C Milner, and Endre Szemerédi. A cure for the telephone disease. *Canadian Mathematical Bulletin*, 15(3):447–450, 1972.
5. Sandra M. Hedetniemi, Stephen T. Hedetniemi, and Arthur L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, pages 319–349, 1988. doi:10.1002/net.3230180406.
6. Jennifer Iglesias, Rajmohan Rajaraman, R Ravi, and Ravi Sundaram. Rumors across radio, wireless, telephone. In *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.
7. Jennifer Iglesias, Rajmohan Rajaraman, R Ravi, and Ravi Sundaram. Plane gossip: Approximating rumor spread in planar graphs. In *Latin American Symposium on Theoretical Informatics*, pages 611–624. Springer, 2018.
8. Tom Leighton, Bruce Maggs, and Satish Rao. Universal packet routing algorithms. In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pages 256–269. IEEE, 1988.
9. Afshin Nikzad and R Ravi. Sending secrets swiftly: Approximation algorithms for generalized multicast problems. In *International Colloquium on Automata, Languages, and Programming*, pages 568–607. Springer, 2014.
10. R Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 202–213. IEEE, 1994.
11. Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis II. System level concurrency control for distributed database systems. *ACM Transactions on Database Systems (TODS)*, pages 178–198, 1978.
12. Aravind Srinivasan and Chung-Piaw Teo. A constant-factor approximation algorithm for packet routing and balancing local vs. global criteria. In *STOC '97 Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 636–643, 1997. doi:10.1145/258533.258658.