Efficient Threshold Detection in a Distributed Environment

Yuval Emek [*] Amos Korman [†]

Abstract

Consider a distributed network in which *events* occur at arbitrary nodes and at unpredicted times. An event occurring at node u is sensed only by u which in turn may invoke a communication protocol that allows nodes to exchange messages with their neighbors. We are interested in the following *threshold detection (TD)* problem inherent to distributed computing: Given some threshold k, the goal of a TD protocol is to broadcast a *termination* signal when at least k events have occurred (throughout the network).

In this paper we develop a randomized TD protocol that may fail with negligible probability but which significantly improves previous results in terms of the message complexity, namely, the total number of messages sent by all participating nodes. With the right choice of parameters our randomized protocol turns into a deterministic one that guarantees low communication burden for any node. This is a principal complexity measure in many applications of wireless networks and which, to the best of our knowledge, has not been bounded before in the context of such problems.

[*]Microsoft Israel R&D Center, Herzelia, Israel and School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel. E-mail: yuvale@eng.tau.ac.il.

[†]CNRS and Université Paris Diderot - Paris 7, France. E-mail: amos.korman@gmail.com. Supported in part by the ANR project ALADDIN, by the INRIA project GANG, and by COST Action 295 DYNAMO.

1 Introduction

The problem. Consider a distributed network modeled as an undirected graph in which nodes are allowed to exchange messages with their neighbors. *Events* may occur at arbitrary nodes and at unpredicted times. It is assumed that an event occurring at node u is sensed only by u, however, other nodes may learn of this event via messages sent from u. Let κ be a (global) variable denoting the number of events that occurred throughout the network. Given some *threshold* k, we wish to know when κ reaches k. More formally, a protocol for the *threshold detection (TD)* problem satisfies the following requirements: (1) if $\kappa \geq k$, then a *termination* signal will be broadcast within finite time; and (2) if a termination signal was broadcast, then $\kappa \geq k$.

We also consider a randomized variant of the TD problem that tolerates a small (one-sided) failure probability q. Formally, given some threshold k, a protocol for the *randomized TD* problem satisfies the following two requirements: (1) if $\kappa \geq k$, then with probability at least $1 - q$, a termination signal will be broadcast within finite time; and (2) if a termination signal was broadcast, then $\kappa \geq k$. Note that the probability in requirement (1) is taken over the coin tosses of the randomized protocol; we do not make any assumptions on the distribution of the input. In fact, in this paper it is assumed that the input is constructed by an adaptive adversary whose decisions may depend on previous coin tosses of the randomized protocol (but not on future ones).

The TD problem finds applications in various contexts of distributed computing. As an example, consider a sensor network designed for detecting a flood. The sensors (nodes) of the network are spread over some terrain; each sensor measures the local amount of rain-fall (measuring one unit of rain-fall corresponds to an event occurring at the node). Clearly, a single unit of rain-fall measured at some sensor does not indicate that a flood is imminent. However, if many (say, k) units of rain-fall are measured across the terrain, then a flood warning (corresponding to the termination signal) should be broadcast.

From a theoretical point of view, the TD problem combines an online flavor with a distributed one. Specifically, the problem aims at capturing an inherent difficulty of distributed computing: the high cost of knowing what happened in remote places. Indeed, the current paper focuses on this difficulty and aspires to the constructions of TD protocols incurring low communication costs.

Communication models. We consider two different communication models. The first one is the standard *message passing* model [14] in which node u can send a message to node v if v is a neighbor of u in the network. The complexity measure we focus on under this model is the *message complexity* defined as the total number of messages sent by all participating nodes throughout the execution of the protocol (i.e., until the termination signal reaches all nodes). Actually, it will be more convenient to consider the *average message complexity* defined as the ratio of the message complexity to the number of nodes in the network. This captures the amount of work exerted by a node for communication purposes on average.

In the second communication model we assume that if a node *transmits* some message, then this message is delivered to *all* its neighbors in the network. We refer to this communication model as the *transmissions* model. The complexity measure we focus on under this model is the *maximum transmission complexity* defined as the maximum number of transmissions made by any single node. This captures the amount of work exerted by a node for communication purposes in the worst case. We assume that both the message passing and transmissions models operate in an asynchronous environment, which means that the delivery of each message may incur an arbitrary, yet finite, delay.

The transmissions model reflects the behavior of wireless networks, where messages are not delivered via physical links, but rather transmitted in the open air and received by all neighbors of the transmitting node. We would like to point out that many works on wireless networks take into consideration the issue of colliding transmissions which may prevent the successful decoding of a message received by some node.

We abstract away this issue. Such an abstraction can be justified by assuming that a designated feature that handles collisions is constructed in a preprocessing stage[1]. Note that in many applications of wireless networks the energy resources of the nodes are limited and in particular, the amount of energy spent by any node for communication purposes should be minimized. Therefore the maximum transmissions complexity seems to be the "right" complexity measure under the transmissions model.

Previous work. There is a long history of research studying various related problems under the message passing model [6, 4, 8, 2, 9, 3, 13, 1, 11, 12, 10, 5]. Among these problems, the *token collection* problem introduced in [2] is probably the closest one to threshold detection. This problem was subsequently generalized in [1] to introduce the *controller* problem. Indeed, the current state of the art TD protocols are the ones derived from the controllers presented in [1, 10, 5]. Specifically, the average message complexity of the TD protocols derived from [1, 10] is $O(\log k \log^2 n)$, where n denotes the number of nodes in the network. For small values of k, this bound was improved to $O(\log k \log^2 k)$ in [5]. It is also (implicitly) shown in [5] that any TD protocol must have average message complexity of $\Omega(\log k)$. This lower bound also holds for the maximum transmission complexity.

To the best of our knowledge, there is no non-trivial upper bound on the maximum transmission complexity of the TD problem under the transmissions model. The techniques in [1, 10, 5] can be used to derive TD protocols that operate under the transmissions model[2], however, an $\Omega(\min\{k, n\})$ factor is inherent to the resulting transmission complexity.

Our contribution. We begin by observing a very simple $\Omega(\log k)$ lower bound on the expected values of both the average message complexity (under the message passing model) and the maximum transmission complexity (under the transmissions model) of any randomized TD protocol with constant failure probability. Our main contribution, however, is the construction of a generic randomized TD protocol that can be adapted to both the message passing and transmissions communication models.

Under the message passing model, the protocol guarantees a negligible (that is, smaller than the reciprocal of any polynomial in n) failure probability and an average message complexity of $O(\log k \log^2 \log n)$ on expectation. In comparison, the rather trivial randomized TD protocol that is based on tossing a biased coin on each event and checking all nodes if this coin turns head must have message complexity $\omega(\log k \log n)$ in order to guarantee a negligible failure probability. Thus, factoring away the inherent $O(\log k)$ factor of the message complexity, our randomized protocol incurs an exponential improvement over the known deterministic protocols and the trivial randomized one. Under the transmissions model, the internal parameters of our randomized protocol can be set to obtain a deterministic protocol with maximum transmission complexity of $O(\log^2 k \log^2 n)$. This again implies an exponential improvement over the simple linear solutions. A summary of the results is presented in Table 1.

2 Preliminaries

Let n denote the number of nodes in the network. For either one of the above two models of communication, we assume that each message is encoded using $\log^{O(1)}(n)$ bits. As commonly assumed in this context (e.g.,

[1] In fact, such a feature is typically implemented in (real-world) wireless networks by the MAC layer. For example, the currently implemented MAC protocols (e.g., WiFi) use carrier-sensing (a node first listens whether someone else is talking, and only if no one is talking, it is allowed to talk) and backoffs (if someone else is talking, one goes into an exponential backoff before attempting to transmit again) to ensure fair and efficient medium access.

[2] To obtain a protocol that operates under the transmissions model from a protocol that operates under the message passing model, simply attach to each message from u to v the identity of the destination node v. When v receives this message it continues its actions accordingly and every other neighbor of u ignores the message.

	Previously known	New
Average message complexity	(deterministic) $O(\log k \log^2 n)$ [1, 10]	randomized $O(\log k \log^2 \log n)$ with negligible failure probability
	$\Omega(\log k)$ for any (deterministic) TD protocol [5]	$\Omega(\log k)$ for any randomized TD protocol with constant failure probability
Maximum transmission complexity	$\tilde{O}(\min\{k, n\})$ (trivial)	(deterministic) $O(\log^2 k \log^2 n)$
	$\Omega(\log k)$ for any (deterministic) TD protocol [5]	$\Omega(\log k)$ for any randomized TD protocol with constant failure probability

Table 1: A summary of the results.

[1, 5, 10]), we too assume that when the protocol starts, a spanning tree T is already constructed over the network and the protocol's communication is restricted to the links of T. Actually, in what follows we assume that the network itself is the tree T.

3 Lower bounds

As a warm-up, we start with simple lower bounds. Consider the message passing model. In this section we establish an $\Omega(\log k)$ lower bound on the expected average message complexity of any randomized TD protocol designed for any constant failure probability $0 \leq q < 1$. An almost identical proof can be used for bounding the maximum transmission complexity in the transmissions model.

Let P be an n-node path. Consider some randomized TD protocol on P designed for an integer k and some constant failure probability $0 \leq q < 1$. Our goal is to find a scenario Γ of events for which Ψ sends $\Omega(n \log k)$ messages on expectation. For simplicity, we assume that n is even and $k+1$ is a power of 2.

Let l and r be the leftmost and rightmost nodes in P, respectively. Each event in Γ is presented either at l or at r. The scenario Γ is a concatenation of $\lambda = \log k - 1$ subscenarios, namely, $\Gamma = \gamma_1 \cdots \gamma_\lambda$. For $1 \leq i \leq \lambda$, the i^{th} subscenario γ_i consists of $(k+1)/2^i$ events, all of them are presented at the same node (either l or r). The scenario Γ is designed so that in response to γ_i, the randomized controller Ψ sends $\Omega(n)$ messages on expectation.

The subscenarios γ_i are defined by induction on i. Let γ_0 be the empty scenario (no messages are sent). Given $0 < i \leq \lambda$ such that the scenario $\Gamma_{i-1} = \gamma_0 \cdot \gamma_1 \cdots \gamma_{i-1}$ is already defined, we define the subscenario γ_i as follows. Let $\gamma_i(l)$ (respectively, $\gamma_i(r)$) be a scenario of $(k+1)/2^i$ events presented at l (resp., at r). Let $A_i(l)$ (respectively, $A_i(r)$) denote the probabilistic event that Ψ sends at least $n/2$ messages in response to the suffix $\gamma_i(l)$ (resp., the suffix $\gamma_i(r)$) of the scenario $\Gamma_i(l) = \Gamma_{i-1} \cdot \gamma_i(l)$ (resp., the scenario $\Gamma_i(r) = \Gamma_{i-1} \cdot \gamma_i(r)$). If $\mathbb{P}[A_i(l)] \geq (1-q)/2$, then we fix $\Gamma_i = \Gamma_i(l)$; otherwise, we fix $\Gamma_i = \Gamma_i(r)$.

Assume by way of contradiction that $\mathbb{P}[A_i(l)] < (1-q)/2$ and $\mathbb{P}[A_i(r)] < (1-q)/2$. Observe first that the number of events in Γ_i is $|\Gamma_i| = \sum_{j=1}^{i} \frac{k+1}{2^i} = (k+1)(1 - 1/2^i)$. Since $i < \log k$, it follows that $|\Gamma_i| < k$. Therefore Ψ cannot signal termination in response to Γ_i. Now consider the scenario $\Gamma_i(l, r) = \Gamma_{i-1} \cdot \gamma_i(l) \cdot \gamma_i(r)$ and let B_i denote the probabilistic event that Ψ sends less than $n/2$ messages in response to the subscenario $\gamma_i(l)$ and less than $n/2$ messages in response to the subscenario $\gamma_i(r)$ in $\Gamma_i(l, r)$.

We argue that B_i occurs with probability greater than q. This is justified as every sequence of coin tosses that resulted in the probabilistic event $\neg A_i(l)$ (in the scenario $\Gamma_i(l)$) and in the probabilistic event $\neg A_i(r)$ (in the scenario $\Gamma_i(r)$), must have resulted in the probabilistic event B_i (in the scenario $\Gamma_i(l,r)$). By a union bound argument on $\mathbb{P}[A_i(l) \vee A_i(r)]$, we conclude that $\mathbb{P}[B_i] \geq \mathbb{P}[\neg A_i(l) \wedge \neg A_i(r)] > q$.

Now, if B_i occurs, then no node in the left side of the path can distinguish the scenario $\Gamma_i(l,r)$ from the scenario $\Gamma_i(l)$ and no node in the right side of the path can distinguish the scenario $\Gamma_i(l,r)$ from the scenario $\Gamma_i(r)$, hence termination is not signaled. On the other hand, the number of events in $\Gamma_i(l,r)$ is $|\Gamma_i(l,r)| = (k+1)(1-1/2^{i-1}) + 2(k+1)/2^i = k+1$ and Ψ should have signaled termination, in contradiction to the assumption that Ψ fails with probability at most q. It follows that Ψ sends $\Omega(n)$ messages in response to the i^{th} subscenario in Γ with probability at least $(1-q)/2$ for every $1 \leq i \leq \lambda$. Assuming that $(1-q)/2 = \Omega(1)$, this sums up to an expected message complexity of $\Omega(n \log k)$ as promised, i.e., the expected average message complexity is $\Omega(\log k)$. This establishes the following theorems.

Theorem 3.1. *Consider the message passing model. Every randomized TD protocol that fails with probability less than a constant incurs $\Omega(\log k)$ average message complexity on expectation.*

Theorem 3.2. *Consider the transmissions model. Every randomized TD protocol that fails with probability less than a constant incurs $\Omega(\log k)$ maximum transmission complexity on expectation.*

4 A level-h separator decomposition

We now turn to describe our constructions. Before dwelling into their details, let us consider first the transmissions model. As mentioned earlier, an $\Omega(\min\{k,n\})$ factor is inherent to the transmission complexity of the protocols in [1, 5, 10]. In particular, this happens in [1] because the behavior of a node does not depend on the number of events it 'saw' but rather on the depth of the node in T. In fact, whenever an event occurs at a node of odd depth, the node sends a message to its parent. Thus, if all k events occur at a single node of odd depth, this node will transmit k times. A similar phenomena occurs also in the protocol of [10], where each event causes $\Omega(1)$ transmissions, even if all events occur at a single node.

The protocol of [5] is first designed assuming the tree is a path, and builds an implicit binary tree over the path. This implicit binary tree is used to aggregate information to the root in a gradual manner. Then, [5] shows how to reduce the problem from a tree to a path. In the transmissions model, their protocol on the path can be modified to have low transmission complexity, however, given a tree, when simulating the protocol on the path using the reduction from tree to path, the protocol may incur linear number of transmissions at a node. For this reason, we avoid the tree-to-path reduction described in [5], and instead construct the following separator decomposition, over which we shall aggregate the information.

A *separator* of a tree T is a node whose removal (together with its incident edges) breaks T into disconnected subtrees, each of size at most $|T|/2$. It is well known that each tree has a separator. Let T be a tree of size n, and let h be a positive integer. Below we define a recursive decomposition of the tree T called a *level-h separator decomposition*.

We describe a level of the recursive procedure operating on a tree T (that may be a subtree of the original tree input to the top recursive level). The procedure on T is composed of at most two stages. At the first stage a separator s of T is selected to be the *representative* of T. The removal of s breaks T into disconnected subtrees $T_1, \ldots, T_l$, each of size at most $|T|/2$. Assume without loss of generality that $|T_1| \geq \cdots \geq |T_l|$. If $|T_l| > n/2^{h-1}$, then we continue recursively to the next level by decomposing $T_1, \ldots, T_{l-1}$ and $T'_l = T - (T_1 \cup \cdots \cup T_{l-1})$, that is, the original subtrees $T_1, \ldots, T_{l-1}$ and the subtree T'_l obtained from T_l by augmenting s to it. Otherwise, we continue to the second stage, described below.

Let $l' \leq l$ be the smallest index such that $|T_{l'}| \leq n/2^{h-1}$. (This is well defined as $|T_l| \leq n/2^{h-1}$.) We group the subtrees $T_{l'}, T_{l'+1}, \ldots, T_l$ in forests $F_1, \ldots, F_t$, where for each $j < t$, $|F_j| \in [n/2^{h-1}, n/2^h)$, and $|F_t| \leq n/2^{h-1}$. (This grouping is possible since for each i such that $l' \leq i \leq l$, we have $|T_i| \leq n/2^{h-1}$.) In addition, we let s be the representative of each of the forests $F_1, \ldots, F_t$. Next, we continue recursively to the next level by decomposing $T_1, T_2, \ldots, T_{l'-1}$ and $T'_{l'}$, where $T'_{l'}$ is obtained by attaching s to $T_{l'}$.

The above completes the description of one level of the recursion operating on the subtree T. Recall that the representative of T is its separator s. Note that if the second stage is not applied, then T is partitioned into the subtrees $T_1, \ldots, T_{l-1}$ and T'_l, and each of these components is assigned a representative in the next level of the recursion. If, on the other hand, the second stage is applied, then T is partitioned into the components: $T_1, \ldots, T_{l'-1}, T'_{l'}$ and $F_1, \ldots, F_t$. The separator s is assigned as the representative of each of the forests $F_1, \ldots, F_t$, whereas the subtrees $T_1, \ldots, T_{l'-1}$ and $T'_{l'}$ are assigned a representative in the next level of the recursion. Note that the recursion must stop after at most $h-1$ levels.

The first stage of our protocol (for both models of communication) is the construction of the above level-h separator decomposition over T, for a value h defined later. We refer to this stage as the *preprocessing* stage. Note that if we do not restrict the message size, then such a decomposition can be constructed in a total of $O(n)$ messages (respectively, transmissions), i.e., $O(1)$ average message (resp., transmission) complexity, by aggregating the structure of T to the root, calculating the decomposition at the root, and broadcasting the decomposition to all nodes. Moreover, one can easily see that even if the sizes of messages are restricted to $O(\log n)$ bits, such a decomposition can be constructed with $O(h)$ average message (resp., transmission) complexity. As shown later, given the level-h separator decomposition, the expected average message complexity of our randomized protocol in the message passing model is $O(h^2 \log k)$. The same bound holds also for the transmission complexity of our deterministic protocol in the transmissions model (though for a different value of h). Thus, the construction of the separator decomposition in the preprocessing stage does not influence the asymptotic complexities of our protocols.

The hierarchy of components. A level-h separator decomposition induces an implicit *component tree* denoted $\mathcal{CT}$, representing the hierarchy between the decomposed components of T. To avoid confusion, we refer to the basic elements of $\mathcal{CT}$ as *vertices*, rather than *nodes* which correspond to the basic elements of T. We build $\mathcal{CT}$ by following the recursive decomposition and letting each vertex in $\mathcal{CT}$ correspond to a decomposed component of T as follows. The root of $\mathcal{CT}$ corresponds to the whole tree T. Given some vertex x in $\mathcal{CT}$ that corresponds to a subtree T' which is partitioned by the recursive procedure into components $C_1, \ldots, C_r$, we let $y_1, \ldots, y_r$ be the children of x in $\mathcal{CT}$, and for each $1 \leq j \leq r$, we let y_j correspond to C_j. Note that if T' is partitioned during the first stage only, then the components are all subtrees, and otherwise, the components include both subtrees and forests.

The component of T corresponding to a vertex $x \in \mathcal{CT}$ is denoted by $C(x)$, and we say that the nodes in $C(x)$ are *assigned* to x. Thus, in particular, if x is a child of y in $\mathcal{CT}$, then $C(x) \subseteq C(y)$. Let $Rep(x)$ denote the representative of $C(x)$. Note that if x is a leaf vertex then $Rep(x)$ does not necessarily belong to $C(x)$. Note also that the components that correspond to leaves of $\mathcal{CT}$ induce a partition of T. For every node $u \in T$, let $\ell(u)$ denote the leaf that u is assigned to.

Define the *height* $h(v)$ of a vertex $v \in \mathcal{CT}$ recursively as follows. The height of the root vertex is h-1, and the height of a non-root vertex v with parent u is $h(v) = h(u) - 1$. Observe that the height of a leaf is at least zero.

Observation 4.1. *The level-h separator decomposition satisfies:*
(1) For every vertex $x \in \mathcal{CT}$ of height i, $|C(x)| = O(n/2^{h-i})$; if x is a leaf then $|C(x)| = O(n/2^h)$ (even if the height of x is larger than 0).
(2) For every vertex x in $\mathcal{CT}$ with parent y, the distance in T between $Rep(x)$ and $Rep(y)$ is at most $|C(x)|$;

the distance in T between any node u and $Rep(\ell(u))$ is at most $|C(\ell(u))| = O(n/2^h)$.
(3) At most 2^{h-1} leaves ℓ of $\mathcal{CT}$ satisfy $|C(\ell)| < n/2^{h-1}$. (This holds because every internal vertex of $\mathcal{CT}$ has at most one such leaf child.)
(4) $\mathcal{CT}$ has at most 2^h leaves.

For simplicity of presentation, in the following description of the protocol, some actions are described as actions taken by the vertices of $\mathcal{CT}$. These actions are actually simulated by the corresponding representatives. E,g., when we say that a vertex $x \in \mathcal{CT}$ sends a message to its parent y in $\mathcal{CT}$, we actually mean that $Rep(x)$ delivers a message to $Rep(y)$ along the unique path from $Rep(x)$ to $Rep(y)$.[3]

5 A randomized TD protocol in the message passing model

In this section we consider the message passing model of communication. Given a tree network T of size n, a parameter k, and a sufficiently small failure probability q, namely $0 < q < 1/\log k$, we construct a randomized TD protocol for T that admits $O(\log k \log^2 \log(1/q))$ average message complexity on expectation. We recall that our randomized protocol operates against an adaptive adversary, in the sense that the adversary's decisions may depend on previous coin tosses made by the protocol.

For simplicity of presentation, we avoid concurrency issues and describe our protocol assuming that the events occur "one-by-one", namely, that an event occurs only after the protocol has finished its operations in response to previous events. The protocol adapts to the general case (in which events may occur very fast or even simultaneously at different nodes) in a rather standard fashion, using *locks* and *handshake* procedures. However, to describe this adaptation, several technicalities need to be addressed, which make the description cumbersome and may distract the reader from the main combinatorial ideas. The more detailed description is thus deferred to the full version of this paper. Let us begin with an overview.

We assume a given level-h separator decomposition, where h is proportional to $\log\log(\log k/q)$. Let $\mathcal{CT}$ be the resulting implicit component tree. Every vertex x at height i in $\mathcal{CT}$ tries to maintain a one-sided $(1+O(1/h))^{i+1}$-estimation of the number of events that occurred in $C(x)$. Unfortunately, in the randomized setting, these estimations are not necessarily as claimed at all times. To overcome this obstacle, our analysis relies on employing Chernoff's bound on the sum of a carefully chosen set of random variables, for making sure that the vertices at the higher heights of $\mathcal{CT}$ maintain such an estimation with high probability, at least as long as the number of events that occurred is sufficiently large. In particular, if the number of events that occurred throughout the tree is close to k, then the root z of $\mathcal{CT}$ maintains a constant estimation of this number with high probability. As explained later in Section 5.2, the specific choice of the (independent) random variables is one of the most technically challenging issues handles with in the current paper.

To obtain the desired estimations, each node keeps track of the number of events that occurred at itself. Consider some leaf ℓ in $\mathcal{CT}$. In order to maintain the above mentioned estimations, a new event that occurs at a node in the component $C(\ell)$ tosses a biased coin and with probability proportional to $2^h/k$, counts the number of events that occurred in $C(\ell)$. If this number indicates that the estimation criterion at the leaf is violated, then this triggers a procedure that propagates information up the component tree $\mathcal{CT}$ until the first ancestor x of ℓ in $\mathcal{CT}$ is found, whose estimation of the number of events in $C(x)$ does not violate the estimation criterion. Subsequently, the estimations of all descendants of x in $\mathcal{CT}$ are updated appropriately.

The above procedure is the heart of the protocol. Once this is established, we can employ a rather standard

[3] The construction of the separator decomposition, which is done in the preprocessing stage, can easily guarantee (cf. [7]) also that the nodes will have the proper knowledge so that such a message can be delivered in T along the unique path from $Rep(x)$ to $Rep(y)$.

procedure that operates in iterations. In each iteration $i \geq 1$, we employ the above procedure with some parameter k_i (for the first iteration, we have $k_1 = k$). During iteration i, when the root z finds out that sufficiently many events have occurred (a constant fraction of k), it signals all nodes to start the next iteration with a new parameter $k_{i+1} \leftarrow k_i - k'$, where k' is the number of events that actually occurred in iteration i. Now, if k events have occurred in the network, then after $O(\log k)$ such iterations the root z of $\mathcal{CT}$ will know this fact with high probability, and hence can "safely" signal termination. We now turn to describe the protocol in detail.

5.1 The randomized protocol

Black and white events. For simplicity of the presentation, we assume that n and k are both powers of 2 (this assumption can be easily removed using standard techniques). Fix $h = \lceil \log\log(\log k/q) + \log(1740) - 3 - \log\log e \rceil$. Each node v keeps track of the number of events that occurred at itself. The protocol *colors* each event in either *black* or *white*. Informally, black events can be thought of as events which were already "counted" by the protocol while white events are still "unknown". When a new event occurs it is first colored white until, after a while, the protocol colors it black. A black event remains black through the end of the execution.

Consider some vertex x at height i. Let $\sharp_{\mathrm{B}}(x)$ denote the current number of black events that occurred in the nodes of $C(x)$, and let

$$\widehat{\sharp}_{\mathrm{B}}(x) = \max\left\{\sharp_{\mathrm{B}}(x), \frac{k}{2^{h-i}}\right\} .$$

The vertex x stores[4] a variable $\eta(x)$ that, as shown later, serves as a one-sided $(1+\lambda/h)^{i+1}$-estimation of $\widehat{\sharp}_{\mathrm{B}}(x)$, where λ is a constant to be determined later on. That is, the variable $\eta(x)$ satisfies the inequalities $\frac{\widehat{\sharp}_{\mathrm{B}}(x)}{(1+\lambda/h)^{i+1}} \leq \eta(x) \leq \widehat{\sharp}_{\mathrm{B}}(x)$. Initially, the variable $\eta(x)$ is set to $\frac{k}{2^{h-i}}$. (Note that if x is the root then $\eta(x)$ is initially set to $k/2$.)

In addition, each internal vertex x with children $w_1, w_2, \cdots, w_t$ keeps a copy of the variable $\eta(w_i)$, for each $1 \leq i \leq t$, and sets $\tilde{\eta}(x) = \sum_{i=1}^{t} \eta(w_i)$. (Informally, $\tilde{\eta}(x)$ is the sum of estimates of x's children, and thus may be a slightly more updated estimate of $\widehat{\sharp}_{\mathrm{B}}(x)$ than $\eta(x)$; a large ratio between $\tilde{\eta}(x)$ and $\eta(x)$ indicates that many events occurred in $C(x)$ since the last time $\eta(x)$ was updated, and thus a new update procedure is required.) The variable $\tilde{\eta}(\ell)$ for a leaf ℓ is initially set to $\tilde{\eta}(\ell) = \eta(\ell)$.

The `Count&Color` subroutine. A basic ingredient of the protocol is a simple subroutine called `Count&Color`. When initiated at some vertex x, the subroutine invokes broadcast and upcast operations in $C(x)$ (cf. [14]); the broadcast operation colors black all the events it encounters (that way, upon the completion of the broadcast, it is guaranteed that all the events in $C(x)$ that occurred prior to the initiation of the procedure are colored black) and the upcast operation counts the number of black events it 'sees' and informs this number to x. Thus, upon the termination of `Count&Color`, vertex x knows the precise number of events that occurred in $C(x)$; since at that time all these events are colored black, this number is precisely $\sharp_{\mathrm{B}}(x)$.

The `Ignition` procedure. Consider some leaf ℓ in $\mathcal{CT}$. The nodes assigned to ℓ (recall that every node is assigned to some leaf) are involved in the following probabilistic process. Each new event occurring at a node u in $C(\ell)$ (which is colored white) invokes procedure `Ignition` that works as follows. Node u tosses a biased coin and with probability $\pi = \ln(30)2^{h+7}/k$ sends an `Ignition` signal to ℓ (again, the signal

[4]Recall that, in fact, it is the node $Rep(x)$ in T that actually stores the variable $\eta(x)$.

is actually sent to $Rep(\ell)$) instructing it to invoke `Count&Color`. Subsequently, when `Count&Color` is completed, ℓ sets $\tilde{\eta}(\ell) = \widehat{\sharp}_{\mathrm{B}}(\ell)$. This is referred to as an *ignition* of that event at leaf ℓ.

The `Find_Pivot` procedure. For every internal vertex x, fix $\tau(x) = \eta(x) \cdot (1 + \lambda/h)$. `Find_Pivot` is invoked at x whenever $\tilde{\eta}(x) > \tau(x)$. Informally, the role of the procedure is to correct the estimation of $\widehat{\sharp}_{\mathrm{B}}(x)$ stored in the variable $\eta(x)$ of x, if this estimation violates the proper estimation criterion.

Consider an invocation of `Find_Pivot` at vertex x of height i. The procedure first invokes Procedure `Update_Estimations` (to be described soon) at x. In particular, `Update_Estimations` updates $\eta(x)$ to be precisely $\widehat{\sharp}_{\mathrm{B}}(x)$. If x is not the root, then `Find_Pivot` continues by informing x's parent y about the new estimation $\eta(x)$. (Recall that to implement the signal delivery from x to y, a signal in T is delivered from $Rep(x)$ to $Rep(y)$. If x is of height i, then the distance between x and y is at most $|C(x)| = O(n/2^{h-i})$.) In turn, when the parent y receives this signal it updates its own copy of that $\eta(x)$ variable. This update incurs also an update to $\tilde{\eta}(y)$ which may trigger the invocation of `Find_Pivot` at y.

Now suppose that x is the root of $\mathcal{CT}$. As before, `Update_Estimations` is invoked at x, but this time if $\eta(x) > k/2$ (which means that at least $k/2$ events occurred), then a new iteration of the protocol starts by invoking `Restart` (to be described soon) at x.

The `Update_Estimations` procedure. When invoked at vertex x, Procedure `Update_Estimations` first invokes `Count&Color` and then, as its name implies, updates the estimations of x and all its descendants. More precisely, if w is either x or one of its descendants in $\mathcal{CT}$, then the variable $\eta(w)$ is updated so that $\eta(w) = \widehat{\sharp}_{\mathrm{B}}(w)$. These updates can be implemented using a broadcast and upcast operations in which the number of events in each component $C(w)$ is collected at w (note that at this time, all these events are black).

The `Restart` procedure. Recall that `Restart` is invoked at the root z only when $\eta(z) = \widehat{\sharp}_{\mathrm{B}}(x)$ is calculated and satisfies $\widehat{\sharp}_{\mathrm{B}}(x) > k/2$. First, if $\widehat{\sharp}_{\mathrm{B}}(x)$ actually satisfies $\widehat{\sharp}_{\mathrm{B}}(x) \geq k$, then a *termination* signal is broadcast throughout T. Otherwise, we set $k \leftarrow k - \widehat{\sharp}_{\mathrm{B}}(x)$ and restart the protocol with the new parameter k. (For that, we do not need to build a new component tree, however we do need to inform all nodes of the new parameter k, and instruct them to initialize their estimations and ignore previous events.)

5.2 Success probability

Note that by the description of the protocol, termination is not signaled unless the number of events κ exceeds k. Our goal now is to prove that with probability at least $1 - q$, if $\kappa \geq k$, then within finite time, a termination signal is broadcast. A time t is called *quiet* if all actions of the protocol in response to previous events have been completed. (Recall that in this extended abstract we assume that a quiet time must exist between the occurrences of any two consecutive events.)

We now show that at any quiet time, $\eta(x)$ serves as a one-sided $(1 + \lambda/h)^{i+1}$-estimation of $\widehat{\sharp}_{\mathrm{B}}(x)$ for every vertex x of height i in $\mathcal{CT}$. Initially, before any event occurs, we have $\eta(x) = \widehat{\sharp}_{\mathrm{B}}(x) = \frac{k}{2^{h-i}}$. Note that a variable $\eta(x)$ may change only when `Update_Estimations` is invoked at x or at one of its ancestors in $\mathcal{CT}$. When such a procedure is completed, we have $\eta(x) = \widehat{\sharp}_{\mathrm{B}}(x)$, and therefore $\eta(x) \leq \widehat{\sharp}_{\mathrm{B}}(x)$ at all times. On the other hand, observe that at any quiet time, we have $\widehat{\sharp}_{\mathrm{B}}(\ell) = \tilde{\eta}(\ell) \leq \tau(\ell) = \eta(\ell)(1 + \lambda/h)$, for any leaf ℓ. Assume now, by induction on i, that for every internal vertex x at height i with children $w_1, \ldots, w_t$, we have at any quiet time, for every $1 \leq j \leq t$, $\widehat{\sharp}_{\mathrm{B}}(w_j)/(1 + \lambda/h)^i \leq \eta(w_j)$. Therefore at any quiet time,

$$\widehat{\sharp}_{\mathrm{B}}(x)/(1 + \lambda/h)^i = \sum_{j=1}^{t} \widehat{\sharp}_{\mathrm{B}}(w_j)/(1 + \lambda/h)^i \leq \sum_{j=1}^{t} \eta(w_j) = \tilde{\eta}(x) \leq \tau(x) = \eta(x)(1 + \lambda/h) \; .$$

It follows that at any quiet time, $\eta(x)$ serves as a one-sided $(1 + \lambda/h)^{i+1}$-estimation of $\widehat{\sharp}_{\mathrm{B}}(x)$ for every vertex

x of height i in $\mathcal{CT}$. Taking $\lambda \leq \frac{\ln(3/2)}{1+1/h}$ establishes the following.

Observation 5.1. *At any quiet time, the root r admits a one-sided $(3/2)$-estimation of $\widehat{\sharp}_{\mathrm{B}}(r)$. Thus, if $\sharp_{\mathrm{B}}(r) \geq k/2$, then the root admits a one-sided $(3/2)$-estimation of $\sharp_{\mathrm{B}}(r)$ (which is the number of black events in T).*

By the definition of `Find_Pivot`, procedure `Restart` is not invoked unless the number of (black) events that occurred in the tree is at least $k/2$. Fix $\xi = k/2^{h+3}$. For any choice of $r \in \mathbb{Z}_{\geq 0}$, we would like to bound the probability that the protocol did not invoke `Restart` after any of the first $k + r\xi$ new-coming events. (Taking $r = 0$ would suffice for the purposes of proving an upper bound on the failure probability of the protocol; however, we aim towards a more general claim which will be used to bound the expected average message complexity.) Observation 5.1 guarantees that as long as the root did not invoke `Restart`, the number of black events that occurred in T is (strictly) smaller than $\frac{3}{2} \cdot \frac{k}{2} = 3k/4$. Therefore for the sake of analysis, we shall ignore the invocations of `Restart` on behalf of the protocol and bound the probability that $k + r\xi$ events occurred throughout T and yet, less than $3k/4$ of them are black.

To avoid dealing with the exact manner in which the protocol colors the events, we consider a slightly more general framework. An adversary inserts $k + r\xi$ *balls* (corresponding to new-coming white events) into ν *bins* (corresponding to the $\nu \leq 2^h$ leaves of the component tree — see Observation 4.1) denoted by $\ell_1, \ldots, \ell_\nu$. From time to time, the content of one of the bins is *emptied* into a *pool* (this corresponds to coloring the nodes black). The act of emptying a bin into the pool is triggered in one of two ways: (i) a new ball inserted into bin ℓ *ignites* with probability $\pi = \ln(30)2^{h+7}/k$, in that case it causes ℓ to be emptied ; or (ii) the adversary may empty a bin, whenever it wishes to do so. Our goal is to bound the probability that less than $3k/4$ balls end up in the pool. Recall, the adversary is adaptive in the sense that its next decisions may depend on previous coin tosses (that determined which of the previous balls ignited).

Fix some bin ℓ and consider a sequence of $\xi = k/2^{h+3}$ balls inserted into ℓ one after the other. Let A be the event that at least one of the last $\xi/16$ balls ignited (once again, $\xi/16$ is a power of 2). Clearly, if A occurs, then more than $15\xi/16$ balls of the sequence end up in the pool. Let m_j be the total number of balls that were inserted into bin ℓ_j throughout the adversarial scenario. Assume first that the adversary is obliged to ensure that $m_j = 0 \mod \xi$ for every $1 \leq j \leq 2^h$. Under this assumption, we shall partition the balls inserted into bin ℓ_j to ξ-*sequences* and consider each ξ-sequence separately. If event A does not occur for some ξ-sequence in bin ℓ, then we will not count its balls in the pool even if some ball in a later sequence ignites and they do end up in the pool. Moreover, when calculating the number of balls that end up in the pool, we do not count any of the last $\xi/16$ balls in a sequence. Obviously, our account for the number of balls in the pool, referred to as the *conservative account*, is dominated by the actual number of balls in the pool.

Let B_s be the event that subject to the conservative account, less than s balls end up in the pool. Our goal now is to prove that $\mathbb{P}[B_{3k/4+2^h\xi}] < (q/\log k) \cdot \exp(-\Omega(r))$, under the assumption that $m_j = 0 \mod \xi$ for every $1 \leq j \leq \nu$.

Let S and S' be two different ξ-sequences (not necessarily in the same bin). We say that S *precedes* S' if the adversary inserted the first ball of S into its bin before it did so with the first ball of S'. By defining the precedence operator, we impose a total order on the $\rho = \frac{k+r\xi}{\xi} = 2^{h+3} + r$ ξ-sequences in the scenario. Let $S_1, \ldots, S_\rho$ be the ξ-sequences ordered in accordance to the precedence operator. For $i = 1, \ldots, \rho$, define the indicator random variable X_i so that $X_i = 1$ if event A occurs for sequence S_i; $X_i = 0$ otherwise. (Recall that A is the event that at least one of the last $\xi/16$ balls of the ξ-sequence ignited.) Let $X = \sum_{i=1}^{\rho} X_i$.

A key observation in our analysis is that the random variables $X_1, \ldots, X_\rho$ are independent[5] and identically distributed with $\mathbb{P}[X_i = 1] = 1 - (1 - \pi)^{\xi/16} = 1 - (1 - \ln(30)2^{h+7}/k)^{k/2^{h+7}} \geq 29/30$. Each ξ-

[5] It is interesting to point out that if the precedence operator is not defined with respect to the first ball in the ξ-sequence, but rather, say, with respect to the last ball, then we cannot guarantee that the random variables $X_1, \ldots, X_\rho$ are independent. The same can be

sequence for which event A occurs contributes exactly $15\xi/16$ balls to the conservative account. Therefore if less than $3k/4 + 2^h\xi$ balls end up in the pool, then $X < \frac{3k/4+2^h\xi}{15\xi/16} = 14 \cdot 2^{h+3}/15$. On the other hand, $\mu = \mathbb{E}[X] \geq 29 \cdot (2^{h+3} + r)/30$.

Recall that Chernoff's bound states that $\mathbb{P}[X < (1-\delta)\mu] < \exp(-\mu\delta^2/2)$ for every $0 < \delta < 1$. Since the event that less than $3k/4 + 2^h\xi$ balls end up in the pool implies that X is (strictly) smaller than $(1 - 1/29)\mu$, we may fix $\delta = 1/29$ and deduce that $\mathbb{P}[B_{3k/4+2^h\xi}] < \exp\left[-(2^{h+3} + r)/1740\right]$. Since $h \geq \log\log(\log k/q) + \log(1740) - 3 - \log\log e$, we obtain $\mathbb{P}[B_{3k/4+2^h\xi}] < (q/\log k) \cdot \exp(-\Omega(r))$.

Now, consider the *real* adversarial scenario in which the $m_j = 0 \mod \xi$ assumption does not necessarily hold. For the sake of the analysis, upon completion of the real scenario, we force the adversary to insert additional d_j *imaginary* balls into bin ℓ_j, where $d_j = \min\{d \geq 0 \mid m_j + d = 0 \mod \xi\}$, for every $1 \leq j \leq \nu$. Clearly, the assumption that $m_j = 0 \mod \xi$ for every $1 \leq j \leq \nu$ holds in the resulting *imaginary* scenario. However, some imaginary balls may ignite and end up in the pool together with some real balls that were not suppose to end up in the pool according to the real scenario. To tackle this obstacle, note that every bin contributes at most ξ more balls (imaginary and real) to the conservative account, thus if $B_{3n/4}$ occurs in the real scenario, then $B_{3n/4+2^h\xi}$ occurs in the imaginary scenario (recall that $\nu \leq 2^h$). Therefore $\mathbb{P}[B_{3k/4}] < (q/\log k) \cdot \exp(-\Omega(r))$ in any adversarial scenario.

Recall that `Restart` is not invoked unless at least $k/2$ occurred. The above inequality asserts that if the number of events exceeds $k/2$ then the probability that `Restart` is not invoked after finite time is at most $(q/\log k) \cdot \exp(-\Omega(r))$. Note that when `Restart` is invoked the number of events M is calculated at the root z. If $M \geq k$, then termination is signaled and otherwise, the protocol sets $k' \leftarrow k - M$ and a new iteration of the protocol starts with the new parameter k'. It follows using union bound that if the number of events exceeds k then the probability that termination is not signaled after at most $\log k$ applications of `Restart` is at most $q \cdot \exp(-\Omega(r))$. We thus obtain the following lemma.

Lemma 5.2. *(1) If the protocol signals termination, then the number of events that occurred in T is at least k; and (2) for every $r \in \mathbb{Z}_{\geq 0}$, the probability that more than $k + r\xi$ events occurred and yet the protocol did not signal termination after finite time is at most $q \cdot \exp(-\Omega(r))$.*

5.3 The average message complexity

We now analyze the expected average message complexity of our randomized protocol. Specifically, we prove that the expected number of messages sent (by all nodes) during the whole execution is $O(n \log k \log^2 \log(1/q))$, i.e., the expected average message complexity is $O(\log k \log^2 \log(1/q))$.

We divide the execution of the protocol to iterations as follows. The first iteration is defined as the time period from the first time an event occurs until the first application of `Restart` is completed, and for $i > 1$, the ith iteration is defined as the time period between the completions times of the ith and $i+1$st iterations. Since there are at most $\log k$ iterations, it remains to show that each iteration of the protocol admits total message complexity $O(n \log^2 \log(1/q))$ on expectation.

Consider now an iteration of the protocol. Note that an application of `Restart` consists of a simple broadcast operation on T and thus incurs $O(n)$ messages. Apart from that, the messages involved are divided to two types: (a) messages sent during the execution of `Ignition`; and (b) messages sent during the execution of `Find_Pivot`.

---

said if the $m_j = 0 \mod \xi$ assumption is violated and residual balls are ignored. In fact, in both "alternative" analysis approaches, a clever adversary may abuse the fact that some of the coin tosses are revealed before the corresponding balls are assigned to a specific ξ-sequence S_i (and hence to a specific random variable X_i).

Let us first bound the expected number of messages of the first type, namely the messages resulted from the different applications of `Ignition`. Each such application is invoked when a new (white) event is presented at some node u. Let ℓ denote the leaf such that u is assigned to, i.e., such that $u \in C(\ell)$. In each such application, a biased coin is tossed and with probability $\sim 2^h/k$, a signal is delivered from u to $Rep(\ell)$, and subsequently, `Count&Color` is invoked at $Rep(\ell)$. By Observation 4.1, these operations consist of sending $O(n/2^h)$ messages. Therefore, given that tk events occurred before termination was signaled, we know that the expected number of messages triggered by all applications of `Ignition` is $O(tn)$. Let Y be a random variable denoting the total number of messages triggered by all applications of `Ignition` throughout the execution. Let Z be a random variable denoting the number of events that occurred before termination was signaled. We have

$$\begin{aligned}\mathbb{E}[Y] \;&=\; \sum_{j=1}^{\infty}\mathbb{E}[\,Y \mid Z=j]\cdot\mathbb{P}[Z=j] \;\le\; \sum_{t=1}^{\infty}\mathbb{E}[\,Y \mid Z=tk]\cdot\mathbb{P}[(t-1)k<Z\le tk] \\ &\le \sum_{t=1}^{\infty}O(tn)\cdot\mathbb{P}[Z>(t-1)k] \;=\; O(n)+\sum_{t=2}^{\infty}O(tn)\cdot\mathbb{P}[Z>tk] \;=\; O(n)\,,\end{aligned}$$

where the last equation follows from Lemma 5.2.

We now turn to bound the number of messages of type (b). Note that each application of `Find_Pivot` at vertex x of height i consists of invoking `Update_Estimations` at x and then sending a signal from x to its parent y in $\mathcal{CT}$. An application of `Update_Estimations` at x consists of two broadcast and upcast operations (one for implementing `Count&Color` and one for updating the estimates in $C(x)$). Both operations are initiated at $Rep(x)$. By observation 4.1, we get that each such application can be implemented using $O(|C(x)|) = O(n/2^{h-i})$ messages. In addition, by Observation 4.1, the signal delivery from x to its parent in $\mathcal{CT}$ also consists of sending $O(n/2^{h-i})$ messages. If follows that the number of messages sent by an application of `Find_Pivot` at vertex x of height i is $O(n/2^{h-i})$.

Recall that `Find_Pivot` is invoked at vertex x of height i only when $\tilde{\eta}(x) > \tau(x) = \eta(x)(1+\lambda/h)$. Since $\eta(x) \ge k/2^{h-i}$, we obtain the following.

Observation 5.3. *Consider an invocation of `Find_Pivot` at some vertex x of height i. The number of new events which occurred at $C(x)$ since the last invocation of `Find_Pivot` was completed at x or at one of x's ancestors in $\mathcal{CT}$ (or since the beginning of the execution if we consider the first invocation of `Find_Pivot` at x) is $\Omega\left(\frac{k}{h2^{h-i}}\right)$. (Note that all these events are now colored black.)*

Since the number of black events is always at most $3k/4$, Observation 5.3 implies that the number of invocations of `Find_Pivot` at vertices of height i is $O(h2^{h-i})$ for each i. Each such invocation incurs $O(n/2^{h-i})$ messages. Therefore, the average number of messages (per node) resulted from the invocations of `Find_Pivot` at height i vertices is $O(h)$ for each i. Summing over all heights, we conclude that the average number of messages resulted from all invocations of `Find_Pivot` is $O(h^2)$. By the choice of $h = O(\log\log(\log k/q))$, we conclude that during each iteration, our protocol admits expected average message complexity $O(\log^2\log(\log k/q))$. Since $q < 1/\log k$, we obtain the following.

Theorem 5.4. *Given the parameter k and failure probability $0 < q < 1/\log k$, there exists a randomized TD protocol on an n-node tree with expected average message complexity $O(\log k \log^2\log(1/q))$.*

By setting $q = 2^{-\log^c n}$ for any constant $c > 1$, we get the following.

Corollary 5.5. *Given the parameter k, there exists a randomized TD protocol on an n-node tree that fails with negligible probability (i.e., asymptotically smaller than the reciprocal of any polynomial in n) and incurs $O(\log k \log^2\log n)$ expected average message complexity.*

6 A deterministic TD protocol in the transmissions model

In this section we show how to translate our randomized protocol from the previous section to obtain a deterministic TD protocol that operates in the transmissions model of communication and admits $O(\log^2 n \log^2 k)$ maximum transmission complexity.

Consider the randomized protocol given by Theorem 5.4. Observe that by choosing $\pi = 1$ we get that every new-coming event ignites (with probability 1) and the protocol becomes deterministic. Also note that by choosing $h = \log n + 1$ we get that $|C(\ell)| = 1$ for every leaf $\ell \in \mathcal{CT}$. Therefore `Ignition` does not incur any communication even though each event ignites. Moreover, we get that for every vertex x, $Rep(x) \in C(x)$.

We now give a simple method for transforming this deterministic protocol to operate under the transmissions model. Note that the communication between nodes is done by three basic sub-protocols, namely, *broadcast* and *upcast* sub-protocols performed on some component, and a *signal delivery* sub-protocol that delivers a signal between two designated nodes (along the shortest path). Observe that for every vertex x, a broadcast operation invoked at x is actually initiated at the representative $Rep(x) \in C(x)$ of $C(x)$. To implement this broadcast in the transmissions model, $Rep(x)$ simply attaches the identity of $C(x)$ to the broadcast message, and then only nodes that belong to $C(x)$ and hear this message for the first time continue their corresponding actions in this broadcast operation. Similarly, note that in the message passing model, the upcast operation on $C(x)$ consists of sending at most one message $M(u)$ from each node u in $C(x)$ to one of its neighbors v. To translate this sub-protocol to the transmissions model, node u simply attaches the identity of v to $M(u)$ and transmits this new combined message; when v hears this message it continues accordingly, while every other neighbor of u ignores it. This method also explains how to adapt the signal delivery sub-protocol to the transmissions model.

We now turn to analyze the maximum transmission complexity of our new deterministic protocol. Similarly to the analysis in Section 5.3, we divide the execution of the protocol to at most $\log k$ iterations. We show that each iteration of the protocol admits $O(\log^2 n \log k)$ maximum transmission complexity. Consider now an iteration of the protocol. Note that an application of `Restart` consists of a simple broadcast operation on T and thus incurs at most one transmission per node. As mentioned, the applications of `Ignition` do not contribute anything to the maximum transmission complexity. It is therefore left to bound the maximum transmission complexity resulted from applying `Find_Pivot`.

The number of times a node u transmits as a result of applying `Find_Pivot` is asymptotically bounded from above by the number of times `Find_Pivot` is applied at a vertex x such that $u \in C(x)$. Fix height i and consider some vertex x at height i such that $u \in C(x)$. Let $t < t'$ be the two times when two consecutive applications of `Find_Pivot` at x were completed. Similarly to the proof of Observation 5.3, it follows from the description of `Find_Pivot` that the number of (black) events in $C(x)$ increased from time t to time t' by a factor of at least $1 + \lambda/h$. Since the total number of (black) events in the iteration never exceeds $3k/4$, we get that the number of times `Find_Pivot` is invoked at x is $O(\log_{1+\lambda/h} k) = O(h \log k)$. Summing over all heights i, we get that the total number of times `Find_Pivot` is invoked at a vertex x such that $u \in C(x)$ is $O(h^2 \log k) = O(\log^2 n \log k)$. The maximum transmission complexity in one iteration is thus $O(\log^2 n \log k)$. The following theorem follows, as there are at most $\log k$ iterations.

Theorem 6.1. *Consider the transmissions model of communication. Given the parameter k, there exists a deterministic TD protocol with maximum transmission complexity $O(\log^2 n \log^2 k)$.*

References

[1] Y. Afek, B. Awerbuch, S.A. Plotkin and M. Saks. Local management of a global resource in a communication network. *J. ACM*, **43**, 1–19, 1996.

[2] Y. Afek and M.E. Saks. Detecting global termination conditions in the face of uncertainty. In *Proc. 7th ACM Symp. on Principles of Distributed Computing*, pages 109–124, 1987.

[3] R. Bar-Yehuda and S. Kutten. Fault tolerant distributed majority commitment. *J. Algorithms*, **9(4)**, 568–582, 1988.

[4] G. Bracha and S. Toueg. A distributed algorithm for generalized deadlock detection In *3rd ACM Sym. on Principles of Distributed Computing*, 285–301, 1984.

[5] Y. Emek and A. Korman. New Bounds for the Controller Problem. In *Proc. 23rd Int. Symp. on Distributed Computing*, 2009.

[6] E. W. Dijkstra and C. S. Scholten. Termination detection for diffusing computations. *Information Processing Letters* **11(1)**, 1–4, 1980.

[7] P. Fraigniaud and C. Gavoille. Routing in trees. In *Proc. 28th Int. Colloq. on Automata, Languages & Prog. (ICALP)*, LNCS 2076, pages 757–772, Springer, 2001.

[8] O. Goldreich and L. Shrira. The effects of link failures on computations in asynchronous rings. In *Proc. 5th Annual ACM Symp. on Principles of Distributed Computing*, 174–185, 1986.

[9] O. Goldreich and L. Shrira. Electing a leader in a ring with link failures In *Acta Informatica* **24(1)**, 79–91, 1987.

[10] A. Korman and S. Kutten. Controller and estimator for dynamic networks. In *Proc. 26th ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, pages 175–184, 2007.

[11] A. Korman and D. Peleg. Labeling schemes for weighted dynamic trees. *J. Information and Computation*, **205(12)**, 1721–1740, 2007.

[12] A. Korman and D. Peleg. Dynamic routing schemes for graphs with low local density. *ACM Trans. on Algorithms*, **4(4)**, 2008.

[13] S. Kutten. Optimal fault-tolerant distributed construction of a spanning forest. *Inf. Process. Lett.*, **27(6)**, 299–307, 1988.

[14] Peleg, D.: *Distributed Computing: A Locality-Sensitive Approach.* SIAM, Philadelphia, 2000.