The Impact of Timing on Linearizability in Counting Networks*

Marios Mavronicolas [†]
Department of Computer Science
University of Cyprus
Nicosia, CY-1678
Cyprus
mavronic@turing.cs.ucy.ac.cy

Marina Papatriantafilou
Max-Planck-Institut für Informatik
Im Stadtwald
D-66123 Saarbrücken, Germany
& CS Dept., Patras University, Greece
ptrianta@mpi-sb.mpg.de

Philippas Tsigas
Max-Planck-Institut für Informatik
Im Stadtwald
D-66123 Saarbrücken, Germany
tsigas@mpi-sb.mpg.de

Abstract

Counting networks form a new class of distributed, low-contention data structures, made up of *balancers* and *wires,* which are suitable for solving a variety of multiprocessor synchronization problems that can be expressed as counting problems. A *linearizable* counting network guarantees that the order of the values it returns respects the real-time order they were requested. Linearizability significantly raises the capabilities of the network, but at a possible price in network size or synchronization support [13, 18]. In this work, we further pursue the systematic study of the impact of *timing* assumptions on linearizability for counting networks, along the line of research recently initiated by Lynch *et al.* in [18]. We consider two basic *timing* models, the instantaneous balancer model, in which the transition of a token from an input to an output port of a balancer is modeled as an instantaneous event, and the *periodic balancer* model, where balancers send out tokens at a fixed rate. In both models, we assume lower and upper bounds on the delays incurred by wires connecting the balancers. We present necessary and sufficient conditions for linearizability in these models, in the form of precise inequalities that involve not only parameters of the timing models, but also certain structural parameters of the counting network, which may be of more general interest. Our results extend and strengthen previous impossibility and possibility results on linearizability in counting networks [13, 18].

*This work will also appear for publication elsewhere. Contact author: Philippas Tsigas

[†]Partially supported by funds for the promotion of research at University of Cyprus (research project "Parallel, Concurrent and Distributed Computations"). Part of the work of this author was performed while visiting Max-Planck Institut für Informatik, Saarbrücken, Germany.

1 Introduction

In the *counting problem,* a number of concurrent processors repeatedly assign themselves successive values from a given range, such as memory addresses or destinations on an interconnection network. A solution is said to be *linearizable* [14] if the order of the assigned values reflects the real-time order in which they were requested. Linearizable counting provides the ground for a number of concurrent solutions to significant multiprocessor synchronization problems, such as time-stamp generation, multi-version database handling, scheduling of multi-threaded computations, implementation of data structures, dynamic load balancing, and buffer management (see, e.g., [7, 9, 11, 23]).

A *counting network* [3] is a highly concurrent data structure used to implement a counter. Roughly speaking, a counting network is a directed graph whose nodes are simple computing elements called *balancers,* and whose edges are called *wires.* A request for a counter value is represented by a *token,* which enters on one of the network's input wires, propagates through the network asynchronously by traversing a sequence of balancers, and leaves on an output wire. Counting networks are among the very few counting techniques that are known to be scalable, since they minimize contention ("hot-spots") as concurrency increases by distributing memory accesses, thus increasing parallelism and throughput (see, e.g., [3, 6, 12, 22, 21]).

In order to enhance the design of concurrent counting techniques, so that they both are scalable and support effective specification and analysis of MIMD shared memory algorithms —that rely on linearizability for correctness— it would be desirable to construct *linearizable counting networks.* Alternatively, it would be useful to study the possibility of using additional software constructs in order to extend a given counting network to become linearizable [13], or the conditions under which implemented counting networks always exhibit a linearizable behavior [18].

In this work we pursue the latter approach, following a direction pointed by a recent watershed paper [18], which studied linearizability properties of *uniform* counting networks relatively to timing assumptions on traffic speed. We further continue the systematic study of the impact of *timing* assumptions on linearizability for counting networks. More specifically, we study the boundaries between linearizable and non-linearizable behaviors of *any* counting network with respect to speed variations of its tokens and balancers, in a hope to provide practitioners with additional formal tools to support decision making in the phase of design. We provide necessary and sufficient conditions for a counting network to be linearizable, in the form of precise inequalities expressed in terms of specific graph-theoretic parameters and their relation to variations in traffic speed.

In more detail, we consider two basic timing models for balancer implementations in either shared memory or message passing. In both models, we follow Lynch *et al.* [18] and assume that there exist (non-zero) lower and upper bounds c_{min} and c_{max}, respectively, on the time it takes a token to traverse a wire from balancer to balancer. In the *instantaneous balancer* model, introduced and studied by Lynch *et al.* [18], the transition of a token from an input to an output port of a balancer is modeled as an instantaneous event. As mentioned in [18], this can be shown to be equivalent to the c_{min} and c_{max} bounds including the traversal of a node, but the output being determined at the instant of the token arrival. However, in some implementations of balancers there may be restrictions due to bandwidth or clock rates. In shared memory implementations of balancers memory accesses to variables implementing the balancers may require a constant number of steps to be completed due to restrictions in bandwidth, while in message passing implementations, processors that use messages to "simulate" the balancers may have access to clocks running at a bounded or fixed rate (see, e.g., [3, 12, 21, 22]). We model these "non-instantaneous" implementations by introducing a new timing model, called the *periodic balancer* model, assuming a *constant* period at which a balancer forwards tokens to its outputs. This assumption is motivated by periodic constraints commonly used in many real-time problems (especially in scheduling real-time tasks on multiprocessors [15, 17]), and resembles a timing model for periodic processes studied by Rhee and Welch [20]. The periodic balancer model is more realistic in that it models balancer delay to be proportional to the number of tokens concurrenttly traversing a balancer; this modeling is aligned with the concept of *stalls* introduced and used by Dwork *et al.* [6] in their elegant framework for analyzing contention in counting networks.

Let r_{min} and r_{max} denote the minimum and maximum balancer's periods, respectively, over all balancers in the network.

We study, in particular, uniform and non-uniform counting networks; in *uniform* networks [18], each node lies on some path from inputs to outputs, and all paths from inputs to outputs have constant lengths. Our study introduces and identifies a crucial graph-theoretic parameter of counting networks, called *influence radius* and denoted $irad(\widetilde{G})$ for the undirected version $\widetilde{G}$ of G; roughly speaking, the influence radius is defined to be the maximum common length of two maximal paths from an inner node of the network to any two output nodes, and captures the maximum degree of influence two output nodes can receive in common. It turns out that the influence radius determines in a precise, quantitative way whether a uniform counting network is linearizable. We believe that the influence radius plays a similar role for other classes of networks as well. Our specific results and their relation to previous work are as follows.

Necessary conditions

For the instantaneous balancer model, we show that any uniform, counting network of depth d is *not* linearizable if $c_{max}/c_{min} > 1 + d/irad(\widetilde{G})$. This result for the case of diffracting trees [22] where $irad(\widetilde{G}) = d = \log n$ constitutes an alternative proof of an impossibility result of Lynch *et al.* [18, Theorem 4.1].

For the limit case where c_{max} tends to infinity, corresponding to completely asynchronous traffic, our impossibility result shows that infinite depth (hence, infinite size) is a necessary

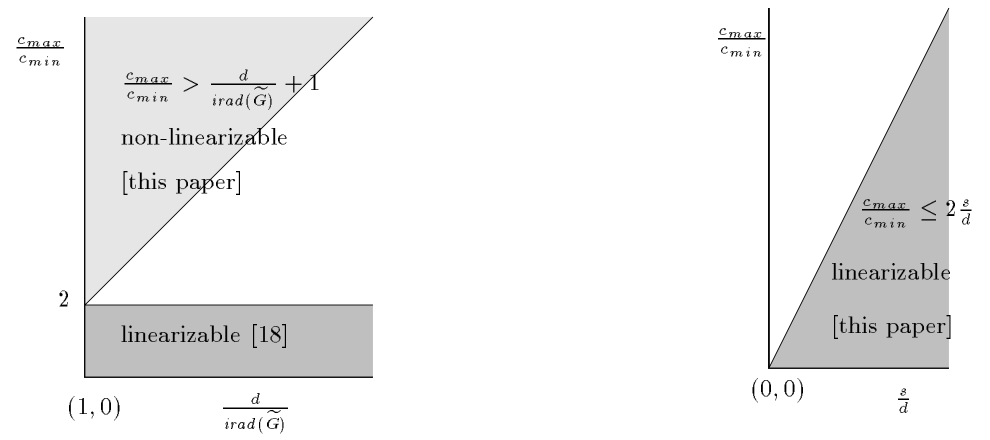

Figure 1: Pictorial summary of the results for the instantaneous balancer mobel

condition for linearizability in uniform, counting networks – this constitutes an alternative proof of an impossibility result of Herlihy *et al.* [13, Theorem 5.1].

Moreover, our previous necessary condition justifies a method proposed in [18, Corollary 3.12] for turning *any* counting network of depth d such that $c_{max}/c_{min} = k$, for any constant $k > 2$, into a linearizable network of depth $O(d)$; the method prepends each of the network's inputs with a simple path consisting of $d \cdot k$ single-input single-output balancers. This can be seen as a mechanism for safeguarding the appropriate ratio $d/rad(\widetilde{G})$.

We next turn to the periodic balancer model, for which we show a necessary condition for linearizability, which depends on the influence radius of the network and the product of fan-outs of balancers along a certain crucial path from inputs to outputs. The proof of this condition follows the same structure as the one for the instantaneous balancer model, but, because of the more delicate timing assumptions in the periodic balancer model, it requires substantially more careful timing arguments.

Sufficient condition

We show that *any* counting network is linearizable if $c_{max}/c_{min} \leq 2s/d$, where s, the *shallowness* of the network, is the length of the *shortest* path from inputs to outputs and d is the depth of the network. This result extends the tight result by Lynch *et al.* which is for uniform networks, to the non-uniform cases. The aforementioned result shows that for uniform counting networks the local condition $c_{max}/c_{min} \leq 2$ is sufficient to guarantee linearizability.

Our results agree well with and provide a complement to known results [13, 18] on linearizability properties of counting networks, in the ways explained in the previous paragraphs. More important, our necessary and sufficient conditions together imply that, in general, linearizability may not be dictated by local conditions [18, Sections 3 and 4], but, rather, by conditions which need involve graph-theoretic parameters describing the structure of the network. We

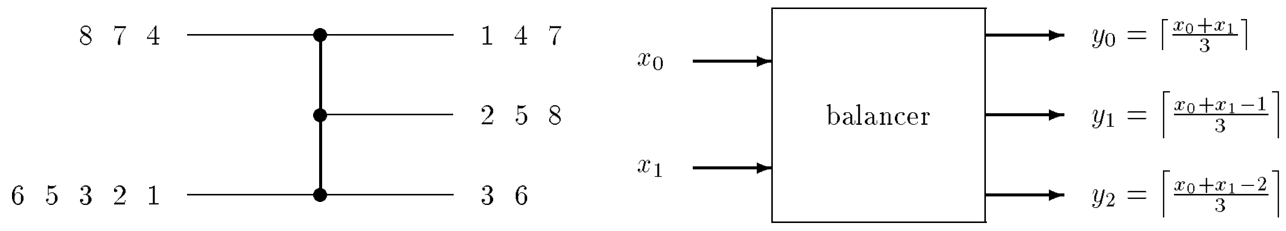
8 7 4
6 5 3 2 1
1 4 7
2 5 8
3 6
x_0
x_1
balancer
$y_0 = \left\lceil \frac{x_0+x_1}{3} \right\rceil$
$y_1 = \left\lceil \frac{x_0+x_1-1}{3} \right\rceil$
$y_2 = \left\lceil \frac{x_0+x_1-2}{3} \right\rceil$

Figure 2: A (2,3)-balancer: symbolic and node representations

remark that our impossibility results are shown using very simple, lock step and round-robin executions, which are expected to be common in practice. Besides, we believe that our proof techniques may be of independent interest.

Our results, which for the instantaneous balancer model are depicted in Figure 1, imply that given a (uniform) counting network, we can determine traffic classes for which the network is linearizable or not by simply computing its depth and influence radius. We proceed by describing counting networks and the model of computation in Section 2. Sections 3 and 4 contain our necessary and sufficient conditions, respectively, for linearizable counting networks. We conclude, in Section 4, with a discussion of our results and some open problems.

2 Preliminaries

Our definitions for balancing networks are standard and follow those in [1, 2, 3, 8, 10, 13, 18]. For self-containment, we define in this section, all terms used throughout the paper.

A (f_{in}, f_{out})-*balancer*, or *balancer* for short, is a computing element receiving tokens on f_{in} *input wires,* and sending out tokens to f_{out} *output wires*; f_{in} and f_{out} are called the *fan-in* and *fan-out* of the balancer, respectively. Tokens arrive on the balancer's input wires at arbitrary times and are output on its output wires. Intuitively, a balancer resembles a toggle mechanism which, given a stream of input tokens, alternately forwards them to its output wires, from top to bottom; thus, a balancer effectively balances the number of tokens that have been output on its output wires. We denote by x_i, $i \in [f_{in}]$, the number of tokens ever received on the i-th input wire of a balancer, and by y_j, $j \in [f_{out}]$, the number of tokens ever sent on its j-th output wire. (Throughout the paper, we will often abuse notation and use x_i (resp., y_i) as both the name of the i-th input wire (resp., output wire) and the number of tokens received (resp., sent) on the wire.) The *state* of a balancer at a given time is the collection of tokens on its input and output wires partitioned per input or output wire. In the initial state all wires contain no tokens. (For clarity we assume that all tokens are distinct.) A state of a $((f_{in}, f_{out})$-balancer is *quiescent* if $\sum_{i=0}^{f_{in}-1} x_i = \sum_{i=0}^{f_{out}-1} y_i$; that is, the number of tokens that entered the balancer is equal to the number of tokens that left it. The following formal properties are required for a (f_{in}, f_{out})-balancer.

1. *Safety property*: In any state, $\sum_{i=0}^{f_{in}-1} x_i \geq \sum_{j=0}^{f_{out}-1} y_j$ (a balancer never creates output tokens).

2. *Liveness property*: Given any finite number of tokens $m = \sum_{i=0}^{f_{in}-1} x_i$ that have been input to a (f_{in}, f_{out})-balancer, the balancer reaches within a finite amount of time a *quiescent state*, i.e. a state such that $\sum_{i=0}^{f_{in}-1} x_i = \sum_{j=0}^{f_{out}-1} y_j = m$ (a balancer never "swallows" input tokens).

3. *Step property*: In any quiescent state, $0 \leq y_i - y_j \leq 1$ for any pair of indices i and j such that $0 \leq i < j \leq f_{out} - 1$; that is, the output has the step property.

A (w_{in}, w_{out})*-balancing network* is a collection of interconnected balancers; such a network is associated with a directed graph G, with three kinds of nodes: w_{in} *source* nodes, $x_0, \ldots x_{w_{in}-1}$, and w_{out} *sink* nodes, $y_0, \ldots x_{w_{out}-1}$ which represent the input and output wires of the network, and a finite number of *inner* nodes, which represent the balancers of the network. The edges of the graph are the wires of the network's balancers, the f_{in} incoming and f_{out} outgoing edges of a node being the corresponding (f_{in}, f_{out})-balancer's input and output wires, respectively; the sink and source nodes have degree 1. We denote by $\widetilde{G}$ the non-directed version of G. Throughout the paper we consider acyclic networks. The *size* of a balancing network is the total number of its balancers. For any wire z in a balancing network, its *depth*, denoted $depth(z)$, is defined to be zero if z is an input wire of the network and $\max_{i \in [f_{in}]} depth(z_i) + 1$ if z is the output wire of a balancer with input wires $z_0, z_1, \ldots, z_{f_{in}-1}$. The *depth* d of a balancing network is the maximal depth over all of its wires. For any balancer b in a balancing network, its *depth*, denoted $depth(b)$, is the maximal wire depth over all of its input wires. Each maximal set of balancers having the same depth l is called the *level* l of the network.

A balancing network is *uniform* [18] if each node of the network lies on some path from inputs to outputs, and all paths from inputs to outputs have the same length, which is equal to the depth of the network. Note that for any balancer b in a uniform balancing network, $depth(b) = dist_G(x, b)$ for any source node x in G. The *configuration* of a balancing network at a given time is defined as the tuple of states of its component balancers at that time. For a configuration σ, denote by $state_\sigma(b)$ the state of balancer b in σ. A configuration is *initial* if all component states are initial. A configuration of a balancing network is *quiescent* if $\sum_{i=0}^{w_{in}-1} x_i = \sum_{j=0}^{w_{out}-1} y_j$; that is, the number of tokens that entered the network is equal to the number of tokens that left it. The safety and liveness properties of a balancing network follow naturally from its definition and the safety and liveness properties of a balancer.

A (w_{in}, w_{out})*-counting network* is a (w_{in}, w_{out})-balancing network for which, in any quiescent state, $0 \leq y_j - y_k \leq 1$, for any pair of indices j and k such that $0 \leq j < k \leq w - 1$; that is, the output of a counting network has the *step property.*

Each one of the w_{out} outputs of a balancing network is connected to an atomic counter (sink node), identified with the name of the respective wire. The tokens exiting the network through wire y_j, are consecutively assigned the integers $j, j+w, j+3w, \ldots$. The integer assigned to a token T by a counter is called the *returned value,* or *value* for short, of the token, and is denoted by $val(T)$.

We briefly describe below our model of multiprocessor computation, following [4, 14, 19]. We model computations of the system as sequences of (atomic) *events*, or *events* for short. Each event is either a *balancer transition event*, denoted by $trans\langle T, b\rangle$, representing transition of token T through balancer b, or a *wire transition event*, denoted by $trans\langle T, b_1, b_2\rangle$ representing transition of token T through a wire connecting an output of b_1 to an input of b_2.

An *execution* $\mathcal{E}$ of a balancing network is a (possibly infinite) sequence of alternating configurations and events $\sigma_0, e_1, \sigma_1, e_2, \sigma_2, \ldots$, where σ_0 is the initial configuration.

Let c_{min} and c_{max}, such that $0 < c_{min} \leq c_{max} < \infty$, be the minimum and maximum time, respectively, that it takes for a token to traverse a link of the network. In the *asynchronous* model, the ratio c_{max}/c_{min} is unbounded, while in the completely synchronous it equals 1. A balancer b passes input tokens to its outputs at a constant rate, denoted by $r(b)$, such that $0 \leq r_b < \infty$. Denote $r_{min} = \min r(b)$ and $r_{max} = \max r(b)$, where the minimum and maximum are taken over all balancers. In the *instantaneous balancer* model, $r_b = 0$ for all balancers b. In the *periodic balancer* model, $r_b > 0$ for all balancers b. In the instantaneous balancer model [18] the upper and lower bounds of traversing a node may be included in the bounds c_{min}, c_{max}, but the output of the node is determined at the instant of the token arrival; in the periodic model there are no bounds on the time that it takes for a token to traverse a node.

A *timed event* is a pair (t, e), where t, the "time", is a nonnegative real number, and e is an event. A *timed sequence* is an infinite sequence of alternating configurations and timed events $\sigma_0, (t_1, e_1), \sigma_1, (t_2, e_2), \sigma_2, \ldots, (t_j, e_j), \sigma_j, \ldots$, where the times are nondecreasing and unbounded. A timed sequence $\mathcal{E}$ is a *timed execution* provided that the following all hold:

1. $\sigma_0, e_1, \sigma_1, \ldots, e_j, \sigma_j, \ldots$ is an execution;
2. (Balancer transition time)
   (a) if the jth timed event is $(t_j, trans\langle T, b_1, b_2\rangle)$ and there are no input tokens in $state_{C_{j-1}}(b_2)$, then there exists a $k > j$ with $t_k = t_j + r(b_2)$ such that the kth timed event is $(t_k, trans\langle T, b_2\rangle)$;
   (b) if the jth timed event is $(t_j, trans\langle T, b\rangle)$ and there exists an input token T' in $state_{\sigma_{j-1}}(b)$, then there exists a $k > j$ with $t_k = t_j + r(b)$ such that the kth timed event is $(t_k, trans\langle T', b\rangle)$;
3. (Lower bound on wire transition time) if the jth timed event is $(t_j, trans\langle T, b_1\rangle)$, then there does not exist a $k > j$ with $t_k < t_j + c_{min}$ such that the kth timed event is $(t_k, trans\langle T, b_1, b_2\rangle)$ for any balancer b_2;

4. (Upper bound on wire transition time) if the jth timed event is $(t_j, \mathit{trans}\langle T, b_1\rangle)$, then there exists a $k > j$ with $t_k \leq t_j + c_{max}$ such that the kth timed event is $(t_k, \mathit{trans}\langle T, b_1, b_2\rangle)$ for some balancer b_2.

The original definition of *linearizability* proposed by Herlihy and Wing [14] is adopted to counting networks in the natural way (cf. [13]). Given a timed execution $\mathcal{E}$, for any token T, define $t_{in}(T, \mathcal{E})$ to be the least t such that $(t, \mathit{trans}\langle T, b_1, b_2\rangle)$ is an event in $\mathcal{E}$, and $t_{out}(T, \mathcal{E})$ to be the greatest t such that $(t, \mathit{trans}\langle T, b\rangle)$ is an event in $\mathcal{E}$. We say that token T_1 *precedes* token T_2 in $\mathcal{E}$ if $t_{in}(T_1, \mathcal{E}) < t_{out}(T_2, \mathcal{E})$; we write $T_1 \xrightarrow{\mathcal{E}} T_2$ to denote this precedence. A timed execution $\mathcal{E}$ of a balancing network is *linearizable* if for every pair of tokens T_1 and T_2 such that $T_1 \xrightarrow{\mathcal{E}} T_2$, it holds that $\mathit{val}(T_1, \mathcal{E}) < \mathit{val}(T_2, \mathcal{E})$ for the values that they get in $\mathcal{E}$. A balancing network is *linearizable* if each of its timed executions is linearizable.

3 Necessary Conditions

Consider an arbitrary uniform counting network G and its undirected version $\widetilde{G}$. For any pair of sink nodes y_j and y_k, where $0 \leq j, k \leq w_{out} - 1$, let $\mathit{dist}_{\widetilde{G}}(y_j, y_k)$ denote the length of a shortest path in $\widetilde{G}$ connecting y_i and y_j (there is at least one simple path connecting them, since $\widetilde{G}$ is connected); each such shortest path is called a *geodesic* between y_j and y_k in $\widetilde{G}$ and denoted by $\gamma_{\widetilde{G}}(y_j, y_k)$. Notice that for uniform networks, the length of any geodesic is odd; hence, there exists on each geodesic $\gamma_{\widetilde{G}}(y_j, y_k)$ a node v_γ such that $\mathit{dist}_{\widetilde{G}}(v_\gamma, y_j) = \mathit{dist}_{\widetilde{G}}(v_\gamma, y_k) = \mathit{dist}_{\widetilde{G}}(y_j, y_k)/2$. Call v_γ a *closest common ancestor* of y_j and y_k in $\widetilde{G}$. The *influence radius* between y_j and y_k in $\widetilde{G}$, denoted $\mathit{irad}_{\widetilde{G}}(y_j, y_k)$, is the distance between y_j (or y_k) and a closest common ancestor of y_j and y_k in $\widetilde{G}$; notice that $\mathit{irad}_{\widetilde{G}}(y_j, y_k) = \mathit{dist}_{\widetilde{G}}(y_j, y_k)/2$. Notice also that if two nodes u and v are both closest common ancestors for the same pair of sink nodes in $\widetilde{G}$, $\mathit{depth}(u) = \mathit{depth}(v)$. The *influence radius* of $\widetilde{G}$, denoted $\mathit{irad}(\widetilde{G})$, is the maximum influence radius among any of its sink nodes.

Sections 3.1 and 3.2 present necessary conditions for linearizability for uniform networks, in the instantaneous and periodic balancer models, respectively. in terms of timing and the depth and influence radius of the network. In all proofs of necessary conditions, we use a timed execution in which tokens propagate through the network in lock step, and each token traverses any balancer and any link after delay exactly b_{min} and c_{min}, respectively. It is called a *fast, synchronous* timed execution.

3.1 Instantaneous Balancer Model

We prove the following theorem:

Theorem 3.1 *In the instantaneous balancer model, a linearizable, uniform counting network does not exist if*

$$\frac{c_{max}}{c_{min}} > \frac{d}{irad(\widetilde{G})} + 1.$$

Proof. Assume, by way of contradiction, that a linearizable, uniform (w_{in}, w_{out})-counting network G exists while $c_{max}/c_{min} > d/irad(\widetilde{G}) + 1$.

The following is an informal outline of the proof. We start with a fast, synchronous timed execution of G in which two distinguished tokens exit G through two antipodal sink nodes*. By "retiming", we slow down the token receiving the least value, while maintaining the propagation of the other token through G, thus, the latter token receives the same value after "retiming". The "retimed" timed execution is further "augmented" to include a sufficient number of tokens fed in the network after the "fast" token exits it, and performing a fast traversal. The assumption on the timing parameters implies that at least one of the additional comments will bypass the "slow" token of the previous timed execution and attain the value it received before retiming. This value is smaller than that of the "fast"token, contradicting linearizability. We now present the details of the formal proof.

Consider an arbitrary pair of antipodal sink nodes y_j and y_k, where $0 \leq j < k \leq w_{out}$; thus, $irad_{\widetilde{G}}(y_j, y_k) = irad(\widetilde{G})$. Let $\mathcal{E}$ be a timed execution in which we feed the network with $k+1$ tokens $T_0, \ldots, T_k$, each T_i, $0 \leq i \leq k$, entering the network through source node $x_{i \bmod w_{in}}$ and executing in lockstep in the maximum speed $(1/c_{min})$. Consider the network after it reaches a quiescent state; the output will have the step property and henceforth y_i equals 1 for $i = 0, \ldots k-1$ and 0 for all other $i < w_{out}$. Let T_κ and $T_\jmath$ be the tokens that exit the network from y_k and y_j, respectively. Tokens $T_\jmath$ and T_κ in $\mathcal{E}$ traverse the network through the paths $\pi(T_\jmath, \mathcal{E}) = b_{\jmath,0} \rightsquigarrow b_{\jmath,1} \rightsquigarrow \ldots \rightsquigarrow b_{\jmath,d-1}$ and $\pi(T_\kappa, \mathcal{E}) = b_{\kappa,0} \rightsquigarrow b_{\kappa,1} \rightsquigarrow \ldots \rightsquigarrow b_{\kappa,d-1}$, respectively.

Now, "perturb" $\mathcal{E}$ to create another timed execution $\mathcal{E}'$ of G in the following way: let the same $k+1$ tokens enter the network from the same input nodes as in $\mathcal{E}$ and keep the same timing as in $\mathcal{E}$ until $T_\jmath$ goes through balancer $b_{\jmath,d-irad(\widetilde{G})}$; then slowdown only $T_\jmath$ to the minimum speed $1/c_{max}$ until it exits the network. Now $T_\jmath$ and T_κ traverse the network through the paths $\pi(T_\jmath, \mathcal{E}') = b'_{\jmath,0} \rightsquigarrow b'_{\jmath,1} \rightsquigarrow \ldots \rightsquigarrow b'_{\jmath,d-1}$ and $\pi(T_\kappa, \mathcal{E}') = b'_{\kappa,0} \rightsquigarrow b'_{\kappa,1} \rightsquigarrow \ldots \rightsquigarrow b'_{\kappa,d-1}$, respectively. Notice that by the construction of $\mathcal{E}'$, $b'_{\jmath,l} = b_{\jmath,l}$ for $0 \leq l \leq d - rad(\widetilde{G})$.

Lemma 3.2 *There is no directed path in* G *from* $b'_{\jmath,d-irad(\widetilde{G})+1}$ *to any node of* $\pi(T_\kappa, \mathcal{E})$.

Proof. Since the traversal of $b_{\jmath,d-irad(\widetilde{G})}$ by $T_\jmath$ is not retimed in $\mathcal{E}'$, it follows that $b'_{\jmath,d-irad(\widetilde{G})+1} = b_{\jmath,d-irad(\widetilde{G})+1}$. Clearly, there is no directed path from $b_{\jmath,d-irad(\widetilde{G})+1}$ to any $b_{\kappa,i}$ with $0 \leq i \leq d - irad(\widetilde{G}) + 1$. The existence of a path from $b_{\jmath,d-irad(\widetilde{G})+1}$ to some $b_{\kappa,i}$ with $i > d - irad(\widetilde{G}) + 1$ would imply that $dist_{\widetilde{G}}(y_k, y_j) < 2irad(\widetilde{G})$, a contradiction, since y_k, y_j are antipodal. $\square$

*sink nodes are antipodal if they realize the influence radius

Lemma 3.3 *In $\mathcal{E}'$ each balancer is visited by the same number of tokens as in $\mathcal{E}$.*

Proof. Since the total number of tokens entering the network in $\mathcal{E}'$ is the same as in $\mathcal{E}$, the lemma holds for each balancer at level 0. By a simple inductive argument using the liveness property of the balancers in the network, it holds for all levels. □

Lemma 3.4 $\pi(T_\kappa, \mathcal{E}') = \pi(T_\kappa, \mathcal{E})$.

Proof. We use induction on the number of levels. It holds that $b_{\kappa,0} = b'_{\kappa,0}$. Assume that for all l, $0 \leq l < d$, it is $b_{\kappa,l} = b'_{\kappa,l}$; from lemma 3.3 it is known that $b'_{\kappa,l}$ in $\mathcal{E}'$ is traversed by the same number of tokens as in $\mathcal{E}$; none of these tokens is $T_\jmath$ (from lemma 3.2). Moreover, from the construction of $\mathcal{E}'$ all the tokens but $T_\jmath$ are not retimed in $\mathcal{E}'$. Hence, T_κ follows the same output port after traversing $b_{\kappa,l}$ as in $\mathcal{E}$. This shows that T_κ will go to the same balancer of level $l+1$ as in $\mathcal{E}$, i.e. $b'_{\kappa,l+1} = b_{\kappa,l+1}$. □

Lemma 3.5 $val(T_\jmath, \mathcal{E}') < k$.

Proof. Since the network is counting, after a quiescence state is reached, the output must have the step property. Since only $k+1$ ($k < w_{out}$) tokens enter the network and since, by lemma 3.4, T_κ exits from the k-th output node, $T_\jmath$ must exit from a node with a lower index. □

We now perturb $\mathcal{E}'$ to obtain a third timed execution $\mathcal{E}''$ of G which is not linearizable. Let $F = \prod_{i=0}^{d-1} f_{out}(b'_{\jmath,i})$, i.e. F is the product of the fan-outs of all the balancers on $\pi(T_\jmath, \mathcal{E}')$. Let a set of tokens Υ, where $|\Upsilon| = F$, enter the network from source node $x_{\jmath \bmod w_{in}}$, in time $dc_{min} + \epsilon$ (recall that dc_{min} is the time that T_κ exits the network), where ϵ is an arbitrarily small constant, and propagate in the the network in lockstep and at maximum speed ($1/c_{min}$). We will prove that there is at least one token $T_v \in \Upsilon$ that will follow the same path $\pi(T_\jmath, \mathcal{E}')$, which $T_\jmath$ followed in $\mathcal{E}'$, and that will bypass $T_\jmath$ before $T\jmath$ exits the network. Let $W(B, \jmath)$ denote the output wire of each balancer B that token $T_\jmath$ has used in $\mathcal{E}'$.

Lemma 3.6 *For each l, $0 \leq l \leq d-1$, at least $\prod_{i=l}^{d-1} f_{out}(b'_{\jmath,i})$ tokens have exited balancer $b'_{\jmath,l}$ by time $dc_{min} + \epsilon + (l+1)c_{min}$ in $\mathcal{E}''$.*

Proof. We show this by induction on the number of levels. All F tokens of Υ go through $b'_{\jmath,0}$. Since they execute in lockstep, by time $dc_{min} + \epsilon + c_{min}$, all (i.e. $\prod_{i=0}^{d-1} f_{out}(b'_{\jmath,i})$) of them have exited balancer $b'_{\jmath,0}$, which proves the base case. Assume that the lemma holds for all l, $0 \leq l < d-1$. Then at least $\prod_{i=l}^{d-1} f_{out}(b'_{\jmath,i})$ tokens are through balancer $b'_{\jmath,l}$ by time $dc_{min} + \epsilon + (l+1)c_{min}$. A fraction $1/f_{out}(b'_{\jmath,l})$ of these tokens are output on the same port that $T_\jmath$ is output in $\mathcal{E}'$ and are, hence, forwarded to $b'_{\jmath,l+1}$. Since all delays in link traversals involving these tokens are equal to c_{min}, it follows that by time $dc_{min} + \epsilon + (l+2)c_{min}$ they will have exited $b'_{\jmath,l+1}$, thus showing the lemma for $l+1$, as well. $\square$

Lemma 3.6 shows that by time $2dc_{min} + \epsilon$ at least one token in Υ will exit the network from the sink node y_r, where $T_\jmath$ exits from in $\mathcal{E}'$. The tokens in Υ enter the network immediately after T_κ exits and, hence T_κ precedes them in $\mathcal{E}''$. At the time that T_κ exits, $T_\jmath$ still has $irad(\widetilde{G})(1 - c_{min}/c_{max})$ links to traverse before exiting the network, for which it will need $irad(\widetilde{G})(c_{max} - c_{min})$ time units. The tokens in Υ need dc_{min} time units to traverse the network. If $dc_{min} < irad(\widetilde{G})(c_{max} - c_{min})$ (which is equivalent to the condition assumed, namely that $c_{max}/c_{min} > d/irad(\widetilde{G}) + 1$) $T_\jmath$ will be bypassed by at least one $T_\upsilon \in \Upsilon$, which will be the first to exit from y_r and will thus get $val(T_\upsilon, \mathcal{E}'') = r$; from lemma 3.5 it is known that $r < k$. Since T_κ has exited the network before the tokens in Υ entered, it follows that it will again, as in $\mathcal{E}'$, exit from wire y_k and get $val(T_\kappa, \mathcal{E}'') = k$. But now $T_\kappa \overset{\mathcal{E}''}{\rightarrow} T_\upsilon$, hence the fact that $r < k$ implies that the linearizability condition is violated. $\square$

3.2 Periodic Balancer Model

We prove the following theorem:

Theorem 3.7 *In the periodic balancer model, a linearizable, uniform counting network does not exist if there exists a path to an output node with fan-out product F_{out} such that*

$$(irad(\widetilde{G}) - 1)r_{max} + irad(\widetilde{G})c_{max} > ((2w_{out} - 1)d + irad(\widetilde{G}) + F_{out} - 1)r_{min} + (d + irad(\widetilde{G}))c_{min} .$$

Proof. Assume, by way of contradiction, that a linearizable, uniform counting network G exists while the condition of the theorem holds, where F_{out} is the maximal product of fan-outs of balancers along a path from a source to a sink node y_j, taken over all such paths. We construct a timed execution of G which is not linearizable. The structure of the proof is similar to the one of theorem 3.1 for the instantaneous balancer model, but, because of the more delicate timing assumptions in the periodic balancer model, it requires substantially more careful timing arguments.

Fix again any pair of antipodal sink nodes y_j and y_k, where $0 \leq j < k \leq w_{out} - 1$; thus, $irad_{\widetilde{G}}(y_j, y_k) = irad(\widetilde{G})$. Let $\mathcal{E}$ be a fast, synchronous timed execution where each token $\widetilde{T}_l$, $0 \leq l \leq k$, enters the network through source node $x_{l \bmod w_{in}}$, each balancer outputs one token per r_{min} time, and all links incur a delay of c_{min}. By the step property for counting networks, when G reaches a quiescent configuration, $y_l = 1$ for $0 \leq l \leq k$ and 0 otherwise. Let $T_\jmath$ and T_κ be the tokens that exit $\mathcal{B}$ through the sink nodes y_j and y_k, respectively, so that $val(T_\jmath, \mathcal{E}) = j$ and $val(T_\kappa, \mathcal{E}) = k$. Assume that in $\mathcal{E}$, $T_\jmath$ and T_κ traverse the network through the paths $\pi(T_\jmath, \mathcal{E}) = b_{\jmath,0} \rightsquigarrow b_{\jmath,1} \rightsquigarrow \ldots \rightsquigarrow b_{\jmath,d-1}$ and $\pi(T_\kappa, \mathcal{E}) = b_{\kappa,0} \rightsquigarrow b_{\kappa,1} \rightsquigarrow \ldots \rightsquigarrow b_{\kappa,d-1}$, respectively.

Notice that T_κ encounters d balancers while traversing the network. Since $\mathcal{E}$ involves $k+1 \leq w_{out}$ tokens, T_κ will wait at most $w_{out} r_{min}$ time to traverse each of the d balancers, and c_{min} time to traverse each of the d links to exit completely the network, so that

Fact 3.1 $t_{out}(T_\kappa, \mathcal{E}) \leq d w_{out} r_{min} + d c_{min}$

We "perturb" $\mathcal{E}$ to obtain another timed execution $\mathcal{E}'$ of G so that each event occurring in $\mathcal{E}$ no later than event $trans\langle T_\jmath, b_{d-irad(\widetilde{G})}\rangle$ is not retimed, while later events are retimed so that each balancer and link encountered by $T_\jmath$ in $\mathcal{E}'$ outputs one token per r_{max} time, and incurs a c_{max} delay, respectively. Assume that in $\mathcal{E}'$, $T_\jmath$ and T_κ traverse the network through the paths $\pi(T_\jmath, \mathcal{E}') = b'_{\jmath,0} \rightsquigarrow b'_{\jmath,1} \rightsquigarrow \ldots \rightsquigarrow b'_{\jmath,d-1}$ and $\pi(T_\kappa, \mathcal{E}') = b'_{\kappa,0} \rightsquigarrow b'_{\kappa,1} \rightsquigarrow \ldots \rightsquigarrow b'_{\kappa,d-1}$, respectively.

Notice that, by construction, $b'_{\jmath,l} = b_{\jmath,l}$ for $0 \leq l \leq d - irad(\widetilde{G})$. Also, since traversal of $b_{\jmath,d-irad(\widetilde{G})}$ by $\jmath$ in $\mathcal{E}$ is not retimed in $\mathcal{E}'$, $\jmath$ follows the same output wire after traversing $b_{\jmath,d-irad(\widetilde{G})}$ in either $\mathcal{E}$ or $\mathcal{E}'$; hence, $b'_{\jmath,d-irad(\widetilde{G})+1} = b_{\jmath,d-irad(\widetilde{G})+1}$. Finally, note that

$$dist_{\widetilde{G}}(b'_{\jmath,d-irad(\widetilde{G})+1}, y_\jmath) = dist_{\widetilde{G}}(b_{\jmath,d-irad(\widetilde{G})+1}, y_\jmath) = irad(\widetilde{G}) - 1\,.$$

Hence, since $y_\jmath$ and y_k are antipodal, it must be that $dist_{\widetilde{G}}(b'_{\jmath,irad(\widetilde{G})+1}, y_k) > irad(\widetilde{G}) - 1$. Hence, it follows that;

Claim 3.8 *There is no directed path in G from $b'_{\jmath,d-irad(\widetilde{G})+1}$ to any node of $\pi(T_\kappa, \mathcal{E})$.*

We continue to show:

Lemma 3.9 $\pi(T_\kappa, \mathcal{E}') = \pi(T_\kappa, \mathcal{E})$

Proof. Since tokens propagate through G in lock step, and no event occurring no later than the traversal of the level $d - irad(\widetilde{G})$ is retimed in $\mathcal{E}'$, it follows that for each l, $0 \leq l \leq d - irad(\widetilde{G})$, $b'_{\kappa,l} = b_{\kappa,l}$. We proceed to show by induction on l, where $d - irad(\widetilde{G}) + 1 \leq l \leq d-1$, that $b_{\kappa,l} = b'_{\kappa,l}$. For the base case where $l = d - irad(\widetilde{G}) + 1$, notice that since traversal of $b_{\kappa,d-irad(\widetilde{G})}$ by T_κ in $\mathcal{E}$ is not retimed in $\mathcal{E}'$, T_κ follows the same output wire after traversing $b_{\kappa,d-irad(\widetilde{G})}$ in either $\mathcal{E}$ or $\mathcal{E}'$; hence, $b'_{\kappa,d-irad(\widetilde{G})+1} = b_{\kappa,d-irad(\widetilde{G})+1}$. Assume inductively that $b'_{\kappa,l} = b_{\kappa,l}$, where $d - irad(\widetilde{G}) + 1 \leq l < d-1$, and consider the balancer $b'_{\kappa,l+1}$. Since the execution is lock step, all not retimed in $\mathcal{E}'$ tokens will reach $b'_{\kappa,l}$, and by Claim 3.8, no retimed token will reach $b_{\kappa,l}$. It follows that each token visiting $b_{\kappa,l}$ in $\mathcal{E}'$ will follow the same link out of $b_{\kappa,l}$ as in $\mathcal{E}$; in particular, this implies that $b'_{\kappa,l+1} = b_{\kappa,l+1}$, as needed. $\square$

Since only $k+1 \leq w_{out}$ tokens are involved in $\mathcal{E}'$ and G is a counting network, the step property immediately implies:

Claim 3.10 $val(T_\jmath, \mathcal{E}') < k$

We now "perturb" $\mathcal{E}'$ to obtain a third timed execution $\mathcal{E}''$ of G which is not linearizable. Let $F_{out} = \prod_{r=1}^{d} f_{out}(b'_{\jmath,r})$; that is, F_{out} is the product of fan-outs of balancers in $\pi(T_\jmath, \mathcal{E}')$. In $\mathcal{E}''$, new tokens $\widetilde{T}_1, \widetilde{T}_2, \ldots, \widetilde{T}_{F_{out}}$ enter the network $\mathcal{B}$ through source node $x_{\jmath \bmod w_{in}}$. For any arbitrarily small constant $\epsilon > 0$, for each l, $1 \leq l \leq F_{out}$,

$$t_{in}(\widetilde{T}_l, \mathcal{E}'') \ = \ t_{out}(T_\kappa, \mathcal{E}') + \epsilon \ \leq \ w_{out} r_{min} + d c_{min} + \epsilon$$

by Fact 3.1. That is, each of the additional tokens enters the network strictly after T_κ exits it in execution $\mathcal{E}'$. All additional tokens traverse the network in round robin order, each balancer they encounter outputs one token per r_{min} time, and all link traversals involving them are equal to c_{min}. We prove:

Lemma 3.11 *For each l, $0 \leq l \leq d-1$, at least $\prod_{r=l+1}^{d-1} f_{out}(b'_{\jmath,r})$ tokens have exited balancer $b'_{\jmath,l}$ in $\mathcal{E}''$ toward balancer $b'_{\jmath,l+1}$ (or an atomic counter if $l = d-1$) by time*

$$d w_{out} r_{min} + d c_{min} + \epsilon + (l+1)\left(\prod_{r=0}^{d-1} f_{out}(b'_{\jmath,r}) + w_{out}\right) r_{min} + l c_{min}$$

Proof. By induction on l. For the base case, where $l = 0$, notice that by construction of $\mathcal{E}''$ $\prod_{r=0}^{d-1} w_{out}(b'_{\jmath,r})$ tokens enter balancer $b'_{\jmath,0}$ by time $w_{out} r_{min} + d c_{min} + \epsilon$. Since

$$\prod_{r=0}^{d-1} f_{out}(b'_{\jmath,r}) \ = \ f_{out}(b_{\jmath,0}) \prod_{r=1}^{d-1} f_{out}(b'_{\jmath,r}) \, ,$$

it follows that at least $\prod_{r=1}^{d-1} w_{out}(b'_{\jmath,r})$ tokens will exit balancer $b'_{\jmath,0}$ toward balancer $b'_{\jmath,1}$. Since $\mathcal{E}''$ involves $\prod_{r=0}^{d-1} f_{out}(b'_{\jmath,r}) + k + 1 \leq \prod_{r=0}^{d-1} f_{out}(b'_{\jmath,r}) + w_{out}$ tokens in total, and $b'_{\jmath,0}$ outputs one token per time r_{min}, it will take at most $(\prod_{r=0}^{d-1} f_{out}(b_{\jmath,r}) + w_{out}) r_{min}$ time for the last token to exit $b'_{\jmath,0}$ toward $b'_{\jmath,1}$, which proves the base case.

Assume inductively that the claim holds for any integer l, where $0 \leq l < d$; that is, at least $\prod_{r=l+1}^{d-1} f_{out}(b'_{\jmath,r})$ tokens have exited balancer $b'_{\jmath,l}$ toward balancer $b'_{\jmath,l+1}$ by time

$$d w_{out} r_{min} + d c_{min} + \epsilon + (l+1)\left(\prod_{r=0}^{d-1} f_{out}(b'_{\jmath,r}) + w_{out}\right) r_{min} + l c_{min}$$

in $\mathcal{E}''$. Since links incur a delay of c_{min} for each of the additional tokens, it follows that at least $\prod_{r=l+1}^{d-1} f_{out}(b'_{\jmath,r})$ tokens enter balancer $b'_{\jmath,l+1}$ by time

$$d w_{out} r_{min} + d c_{min} + \epsilon + (l+1)\left(\prod_{r=0}^{d-1} f_{out}(b'_{\jmath,r}) + w_{out}\right) r_{min} + l c_{min} + c_{min}$$

Since

$$\prod_{r=l+1}^{d-1} f_{out}(b'_{\jmath,r}) \ = \ f_{out}(b'_{\jmath,l}) \prod_{r=l+2}^{d-1} f_{out}(b'_{\jmath,r}) \, ,$$

it follows that at least $\prod_{r=l+2}^{d-1} f_{out}(b'_{j,r})$ will exit balancer $b'_{j,l+1}$ toward balancer $b'_{j,l+2}$. Since $\mathcal{E}''$ involves $\prod_{r=0}^{d-1} f_{out}(b'_{j,r}) + k + 1 \leq \prod_{r=0}^{d-1} f_{out}(b'_{j,r}) + w_{out}$ tokens in total, and $b'_{j,l+1}$ outputs one token per time r_{min}, it will take at most $(\prod_{r=0}^{d-1} f_{out}(b'_{j,r}) + w_{out})r_{min}$ time for the last token to exit $b'_{j,l+1}$ toward $b'_{j,l+2}$, which proves that the lemma holds for $l+1$, as well. $\square$

By Lemma 3.11, at least $f_{out}(b'_{j,d})$ tokens have exited balancer $b'_{j,d-1}$ by time

$$(2dw_{out} + \prod_{r=0}^{d-1} f_{out}(b'_{j,r}))r_{min} + (2d-1)c_{min} + \epsilon\,.$$

Let $\widetilde{T}$ be the first of these tokens to exit; thus,

$$t_{out}(\widetilde{T}, \mathcal{E}'') \;\leq\; (2dw_{out} + \prod_{r=0}^{d-1} f_{out}(b'_{j,r}))r_{min} + 2dc_{min} + \epsilon\,.$$

By construction of $\mathcal{E}''$, T_j is "fast" in $\mathcal{E}''$ till it passes $b'_{j,d-irad(\widetilde{G})}$, and "slow" afterwards. Hence, even if T_j never waits at a balancer *en route* due to other tokens concurrently traversing the same balancer, it holds that

$$\begin{aligned} t_{out}(T_{j,\mathcal{E}''}) \;\;\geq\;\; & (d - irad(\widetilde{G}) + 1)r_{min} + (d - irad(\widetilde{G}))c_{min} + \\ & (irad(\widetilde{G}) - 1)r_{max} + irad(\widetilde{G})c_{max} \end{aligned}$$

Hence, the difference $t_{out}(T_j, \mathcal{E}'') - t_{out}(\widetilde{T}, \mathcal{E}'')$ is greater than

$$(irad(\widetilde{G})-1)r_{max}+irad(\widetilde{G})c_{max}-((2w_{out}-1)d+\prod_{r=0}^{d-1} f_{out}(b'_{j,r})+irad(\widetilde{G})-1)r_{min}-(d-irad(\widetilde{G}))c_{min}$$

which is greater than 0 by assumption. Hence, $\widetilde{T}$ reaches $b'_{j,d}$ before T_j in $\mathcal{E}''$ and receives $val(\widetilde{T}, \mathcal{E}'') = r$. Since $val(T_\kappa) = k > r$, linearizability implies that $\widetilde{T} \overset{\mathcal{E}''}{\to} T_\kappa$, a contradiction. $\square$

4 Sufficient Conditions

In this section, we present our sufficient condition for linearizability in counting networks, which generalizes the respective result in [18], for the case of non-uniform networks, too.

Consider any arbitrary (w_{in}, w_{out})-counting network G (uniform or not), and let its *shallowness*, $s = \min_{i,j} dist_G(x_i, y_j)$; that is, s is the length of the shortest directed path in G. We prove:

Theorem 4.1 *In the instantaneous balancer model, a counting network is linearizable if*

$$\frac{c_{max}}{c_{min}} \leq 2\frac{s}{d}.$$

Proof. Assume, by way of contradiction, that $c_{max}/c_{min} \leq 2s/d$, while there exists a uniform counting network $\mathcal{B}$ which is not linearizable. By definition of linearizability, there exists a timed execution $\mathcal{E}$ of $\mathcal{B}$ such that for a pair of tokens T_κ and T_υ $T_\kappa \xrightarrow{\mathcal{E}} T_\upsilon$, while $val(T_\kappa, \mathcal{E}) > val(T_\upsilon, \mathcal{E})$.

We start with some auxiliary definitions. Following [18], we associate with each balancer b and token T in a timed execution, auxiliary *history variables* $\mathcal{H}_b(t)$ and $\mathcal{H}_T(t)$, respectively, which formalize the "knowledge" b and T have, respectively, at time t about other tokens in the network. Formally, at time 0, $\mathcal{H}_b(0) = \emptyset$ and $\mathcal{H}_T(0) = T$. Each time a token traverses a balancer, the knowledge of the two is combined; formally, if a token T traverses a balancer b at time t, then $\mathcal{H}_b(t) = \mathcal{H}_T(t) = \mathcal{H}_b(t^-) \cup \mathcal{H}_T(t^-)$, where t^- is the time of occurrence of the timed event immediately preceding this in the timed execution. For a token T traversing $\mathcal{B}$ through the path $b_0 \rightsquigarrow b_2 \rightsquigarrow \ldots \rightsquigarrow b_{d-1}$, crossing each b_i at time t_i, define the *history sequence numbers* $S_T^0, \ldots, S_T^{d-1}$, where S_T^i is the number of tokens that have been through b_i by time t_i in the timed execution.

Let y_k be the output node through which T_κ exits $\mathcal{B}$ in $\mathcal{E}$. The proof of [18, Lemma 3.1] does not rely on uniformity; hence, it applies to non-uniform networks as well to yield:

Claim 4.2 *Assume T_κ is the a^{th} token to exit $\mathcal{B}$ through y_k. Then, $|\mathcal{H}_{T_\kappa}(t_{out}(T_\kappa, \mathcal{E}))| \geq w(a-1) + k + 1$.*

We now "perturb" $\mathcal{E}$ to obtain a timed execution $\mathcal{E}'$ which contains only the tokens in $\mathcal{H}_{T_\kappa}(t_{out}(T_\kappa, \mathcal{E}))$ following the same timing in traversing $\mathcal{B}$. Since no events were retimed and we only removed tokens about which T_κ did not "know" in $\mathcal{E}$, token T_κ still exits $\mathcal{B}$ through y_k at time $t_{out}(T_\kappa, \mathcal{E}') = t_{out}(T_\kappa, \mathcal{E})$ in $\mathcal{E}'$. Note also that since in $\mathcal{E}$ T_υ enters $\mathcal{B}$ after T_κ has exited, T_υ does not belong to $\mathcal{H}_{T_\kappa}(t_{out}(T_\kappa, \mathcal{E}))$; hence, T_υ is not participating in $\mathcal{E}'$. Now consider the token T_μ for which $val(T_\mu, \mathcal{E}') = val(T_\upsilon, \mathcal{E})$. By construction, $t_{in}(T_\mu, \mathcal{E}') = t_{in}(T_\mu, \mathcal{E})$. Since T_κ "knows" about T_μ at the time it is exiting $\mathcal{B}$, and this information must have been propagated through $\mathcal{B}$ from some input wire to y_k, it must have traversed at least s wires. Hence, T_μ must have entered $\mathcal{B}$ by time sc_{min} before in the latest. Hence, it follows;

Claim 4.3 $t_{in}(T_\mu, \mathcal{E}) \; = \; t_{in}(T_\mu, \mathcal{E}') \; \leq \; t_{out}(T_\kappa, \mathcal{E}) - sc_{min} \; = \; t_{out}(T_\kappa, \mathcal{E}') - sc_{min}$

The rest of the proof shows that T_μ must have been bypassed in $\mathcal{E}$ by some faster token, which, in turn, has similarly been bypassed by some other faster token; repeating over this argument yields that T_μ was bypassed by T_υ; note that T_υ returns the same value that T_μ returns in $\mathcal{E}'$. Next, we use the assumption $c_{max}/c_{min} \leq 2s/d$ to show that for this to be possible T_υ must have entered $\mathcal{B}$ before T_κ exited it in $\mathcal{E}$; this contradicts the assumption that $T_\kappa \xrightarrow{\mathcal{E}} T_\upsilon$ and completes the proof.

Let $\mathcal{V}$ be the set of balancers visited by tokens during $\mathcal{E}'$. Then,

Lemma 4.4 *In $\mathcal{E}$ each $b \in \mathcal{V}$ processes at least the same number of tokens that it processes in $\mathcal{E}'$.*

Proof. Since the total number of tokens entering the network in $\mathcal{E}'$ is less than in $\mathcal{E}$, the lemma holds for each balancer at level 0. By a simple inductive argument using the safety property which holds for all the balancers in the network, it holds for all levels. $\Box$

Token T_μ during $\mathcal{E}'$ is traversing the network through a sequence of balancers $b_0, \ldots, b_{l-1}$ along path $\pi(T_\mu, \mathcal{E}')$, by going through b_i at time instant t_i^μ. We say that a token T_{μ_j} in $\mathcal{E}$ *simulates steps* of T_μ of $\mathcal{E}'$ on balancers $b_{j_1}, \ldots b_{j_x}$ of $\pi(T_\mu, \mathcal{E}')$, if T_{μ_j} goes through these balancers in $\mathcal{E}$ and its history sequence numbers corresponding to them equal the respective history sequence numbers of T_μ in $\mathcal{E}'$ (naturally, T_μ in $\mathcal{E}$ may simulate itself). From the previous lemma we conclude that there exist tokens $T_{\mu_1}, \ldots, T_{\mu_\varepsilon}$, which in $\mathcal{E}$ simulate consecutive steps of T_μ of $\mathcal{E}'$. Note that $T_{\mu_\varepsilon} = T_\upsilon$. The following lemma is essential for the completeness of the proof of our theorem.

Lemma 4.5 *For any T_{μ_j}, which simulates in $\mathcal{E}$ steps of T_μ of $\mathcal{E}'$ on balancers $b_{i_j}, \ldots b_{i_x}$ of $\pi(T_\mu, \mathcal{E}')$, the time $t_i^{\mu_j}$ that it goes through balancer $b_i \in \{b_{i_j}, \ldots, b_{i_x}\}$ is such that $t_i^{\mu_j} \leq t_{in}(T_\mu, \mathcal{E}) + depth(b_{i_j})c_{max}$.*

Proof. By induction on the length of the path $\pi(T_\mu, \mathcal{E}')$. At the first wire T_μ simulates itself. Tokens can not traverse links slower than $1/c_{max}$ and a token that is simulating T_μ has to bypass a token that is already simulating T_μ ... that has simulated T_μ because it has bypassed it on a balancer of $\pi(T_\mu, \mathcal{E}')$. $\Box$

From the last lemma we have that:

$$\begin{aligned} t_{in}(T_\upsilon, \mathcal{E}) + sc_{min} &\leq t_{out}((T_\upsilon, \mathcal{E}) \leq t_{in}(T_\mu, \mathcal{E}) + length(\pi(T_\mu, \mathcal{E}')) \\ &\leq t_{out}(T_\mu, \mathcal{E}) \leq t_{in}(T_\mu, \mathcal{E}) + dc_{max} \end{aligned}$$

Combining the above with the inequality of claim 4.3 we have that: $t_{in}(T_\upsilon, \mathcal{E}) \leq t_{out}(T_\kappa, \mathcal{E}) - 2sc_{min} + dc_{max}$. For positive values of $2sc_{min} - dc_{max}$, which holds from our assumption, it follows that $t_{in}(T_\upsilon, \mathcal{E}) < t_{out}(T_\kappa, \mathcal{E})$, contradicting the assumption that $T_\kappa \xrightarrow{\mathcal{E}} T_\upsilon$. $\Box$

Theorem 4.1 essentially says that the less "equilateral" the network is, the smaller are the variations in token speeds under which it can retain linearizability; specialized in the case of uniform counting networks, where $s = d$, it yields that $c_{max}/c_{min} \leq 2$ is sufficient for linearizability – a result shown in [18].

Discussion

We presented necessary and sufficient conditions for linearizability in counting networks, under different timing assumptions on balancers and wires. Although we do not yet have a complete characterization of linearizability for the specific timing models we consider, our results demonstrate how the possibility of achieving linearizability depends on both timing parameters of the model and structural parameters of the network.

We remark that the proofs of our necessary conditions can be extended to apply to other classes of balancing networks too, suggesting that it is not the requirement for the step property that has been the main obstacle to implementing linearizable counting networks, but, rather, the requirement for linearizability.

Our work leaves open several interesting problems. Can our necessary conditions be extended to general (non-uniform) counting networks? An obvious open problem is to prove a sufficient condition for linearizability in the periodic balancer model. It would also be interesting to understand how much non-linearizable a counting network may be in case linearizability is impossible; Lynch *et al.* [18, Theorem 4.4] take the first step in this direction by providing a lower bound on the *non-linearizability fraction* for the special case of the bitonic counting network. Our necessary conditions should yield similar results for *any* uniform network in the models we studied. Does our sufficient condition for linearizability hold also for counting networks required to handle both tokens and *antitokens* [21]?

References

[1] E. Aharonson and H. Attiya, "Counting Networks with Arbitrary Fan-Out," *Distributed Computing,* Vol. 8, pp. 163–169, 1995. Preliminary version: *Proceedings of the 3rd Annual ACM–SIAM Symposium on Discrete Algorithms,* pp. 104–113, January 1992.

[2] W. Aiello, R. Venkatesan and M. Yung, "Coins, Weights and Contention in Balancing Networks," *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing,* pp. 193–205, August 1994.

[3] J. Aspnes, M. Herlihy and N. Shavit, "Counting Networks," *Journal of the ACM,* Vol. 41, No. 5, pp. 1020–1048, September 1994. Preliminary version: "Counting Networks and Multi-Processor Coordination," *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing,* pp. 348–358, 1991.

[4] H. Attiya and M. Mavronicolas, "Efficiency of Semi-Synchronous versus Asynchronous Networks," *Mathematical Systems Theory,* Vol. 27, No. 6, pp. 547–571, November/December 1994.

[5] C. Busch and M. Mavronicolas, "A Combinatorial Treatment of Balancing Networks," *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing,* pp. 206–215, August 1994.

[6] C. Dwork, M. Herlihy and O. Waarts, "Contention in Shared Memory Algorithms," *Proceedings of the 25th Annual ACM Symposium on Theory of Computing,* pp. 174–183, 1993.

[7] C. S. Ellis and T. J. Olson, "Algorithms for Parallel Memory Allocation," *Journal of Parallel Programming,* Vol. 17, No. 4, pp. 303–345, August 1988.

[8] E. W. Felten, A. LaMarca and R. Ladner, "Building Counting Networks from Larger Balancers," Technical Report 93-04-09, Department of Computer Science and Engineering, University of Washington, April 1993.

[9] A. Gottlieb, B. D. Lubachevsky and L. Rudolph, "Basic Techniques for the Efficient Coordination of Very Large Numbers of Cooperating Sequential Processors," *ACM Transactions on Programming Languages and Systems,* Vol. 5, No. 2, pp. 164–189, April 1983.

[10] N. Hardavellas, D. Karakos and M. Mavronicolas, "Notes on Sorting and Counting Networks," *Proceedings of the 7th International Workshop on Distributed Algorithms (WDAG-93),* Lecture Notes in Computer Science, Vol. # 725 (A. Schiper, ed.), Springer-Verlag, pp. 234–248, Lausanne, Switzerland, September 1993.

[11] M. Herlihy, "A Methodology for Implementing Highly Concurrent Data Structures," *Proceedings of the 2nd Annual ACM Symposium on Principles and Practice of Parallel Programming,* pp. 197–206, March 1990.

[12] M. Herlihy, B.-C. Lim and N. Shavit, "Low Contention Load Balancing on Large-Scale Multiprocessors," *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures,* pp. 219–227, July 1992.

[13] M. Herlihy, N. Shavit and O. Waarts, "Linearizable Counting Networks," *Distributed Computing,* to appear, 1996. Preliminary version: "Low Contention Linearizable Counting Networks," *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science,* pp. 526–535, October 1991.

[14] M. Herlihy and J. Wing, "Linearizability: A Correctness Condition for Concurrent Objects," *ACM Transactions on Programming Languages and Systems,* Vol. 12, No. 3, pp. 463–492, July 1990.

[15] K. Jeffay, D. F. Stanat and C. U. Martel, "On Optimal, Non-Preemptive Scheduling of Periodic and Sporadic Tasks," *Proceedings of the 12th IEEE Real-Time Systems Symposium,* pp. 129–139, December 1991.

[16] M. Klugerman and C. Plaxton, "Small-Depth Counting Networks," *Proceedings of the 24th Annual ACM Symposium on Theory of Computing,* pp. 417–428, May 1992.

[17] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM,* Vol. 20, No. 1, pp. 46–61, January 1973.

[18] N. Lynch, N. Shavit, A. Shvartsman and D. Touitou, "Counting Networks are Practically Linearizable," *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing,* May 1996, to appear.

[19] N. Lynch and M. Tuttle, "An Introduction to Input/Output Automata," *CWI Quarterly,* Vol. 2, No. 3, pp. 219–246, September 1989.

[20] I. Rhee and J. L. Welch, "The Impact of Time on the Session Problem," *Proceedings of the 11th Annual ACM Symposium on Principles of Distributed Computing,* pp. 191–202, August 1992.

[21] N. Shavit and D. Touitou, "Elimination Trees and the Construction of Pools and Stacks," Preliminary version: *Proceedings of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures,* pp. 54–63, July 1995.

[22] N. Shavit and A. Zemach, "Diffracting Trees," *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures,* pp. 167–176, June 1994.

[23] H. S. Stone, "Database Applications of the Fetch-and-Add Instruction," *IEEE Transactions on Computers,* Vol. C-33, No. 7, pp. 604–612, July 1984.