Threshold Counters with Increments and Decrements*

COSTAS BUSCH
Brown University, USA

NEOPHYTOS DEMETRIOU
University of Cyprus, Cyprus

MAURICE HERLIHY
Brown University, USA

MARIOS MAVRONICOLAS
University of Connecticut, USA

Abstract

A *threshold counter* is a shared data structure that assumes integer values. It provides two operations: Increment changes the current counter value from v to $v+1$, while Read returns the value $\lfloor v/w \rfloor$, where v is the current counter value and w is a fixed constant. Thus, the Read operation returns the "approximate" value of the counter to within the constant w. Threshold counters have many potential uses, including software barrier synchronization.

Threshold networks are a class of distributed data structures that can be used to construct highly-concurrent, low-contention implementations of shared threshold counters. In this paper, we give the first proof that *any* threshold network construction of a threshold counter can be extended to support a Decrement operation that changes the counter value from v to $v-1$.

Keywords
Distributed Computing, Threshold Counters, Threshold and Weak Threshold Networks, Increments, Decrements

1 Introduction

A *threshold counter* is a shared data structure that assumes integer values. It provides two operations: Increment changes the current counter value from v to

*This work has been supported by NSF grant DMS-9505949.

$v+1$, but does not return any information, while Read returns the value $\lfloor v/w \rfloor$, where v is the current counter value and w is a fixed constant. Thus, the Read operation returns the "approximate" value of the counter to within the constant w. Threshold counters have a variety of potential uses, most obviously software barrier synchronization (see, for example, [11, Section 4.2.5], or [6, 7]). Threshold counters are interesting because they can sometimes be implemented more efficiently than exact counters.

The most obvious way to implement a shared counter, whether threshold or exact, is to use a single shared variable protected by a lock. However, such centralized data structures may become "hot-spots" for shared memory communication, or a "sequential bottleneck" with respect to concurrency. Aspnes *et al.* [3] devised a class of distributed data structures, called *balancing networks*, that provides a decentralized way to solve a variety of counter-based synchronization problems.

Balancing networks are made up of balancers. Informally, a *balancer* [3] is a switching element with input wires and output wires. *Tokens* arrive asynchronously on input wires, and are routed to successive output wires in "round-robin" fashion. A *balancing network* is an acyclic network of balancers. A balancing network's *depth* is the length of its longest path.

Balancing networks can be used to construct *counting networks* [3], which are useful for constructing shared exact counters, and *smoothing networks* [3], which are useful for load balancing. Balancing networks can also be used to construct *threshold networks* [3] and *weak threshold networks* [4], which provide highly-concurrent, low-contention implementations of threshold counters. Each of these classes of networks supports some form of Increment operation, implemented by passing a token through the network.

Threshold networks are interesting because there are constructions of them with substantially lower depth than the best known, practical construction of counting networks. While the most practical construction of a counting network known to date is the *bitonic counting network* [3, Section 3] of depth approximately $\log^2 w$, there exists, in contrast, a threshold network construction of depth $\log w$ [3, Section 5.3].

Supporting decrements in threshold and weak threshold networks would allow them to implement decrementable threshold counters, which have many potential practical uses. For example, one might use a decrementable threshold counter to control memory allocation policies on a multiprocessor. A thread might increment the counter when it allocates a block of memory, and decrement the counter when it frees that block. The operating system might monitor the counter, requesting additional resources if the counter's approximate value exceeds a certain threshold. In this work, we address the question of supporting decrements in threshold and weak threshold networks.

The principal contribution of this work is the first proof that *any* threshold network implementation of a threshold counter can be extended to support a Decrement operation that changes the counter value from v to $v-1$. We also show that

the same is true of weak threshold network implementations under the assumption that the weak threshold network is made up of balancers, called *regular,* that have the same number of input and output wires.

The extension uses a new construct called an *antitoken*, which was recently introduced by Shavit and Touitou [10]. While each token that arrives at a balancer advances the toggle and exits on the next successive output wire, an antitoken, by contrast, sets the toggle back, and exits on the preceding wire. Informally, an antitoken "cancels" the effect of the most recent token, and vice versa.

Shavit and Touitou [10] proved that antitokens implement a Decrement operation for a restricted class of balancing networks called a *counting tree.* Subsequently, Aiello *et al.* [2] proved that antitokens are more powerful: they can be used to extend counting networks and smoothing networks to support decrements. More generally, they identified a broad class of properties, called *boundedness properties*, that are preserved by the introduction of antitokens; thus, if a balancing network satisfies any arbitrary boundedness property when traversed by tokens alone, then it continues to satisfy that same property when traversed by tokens and antitokens. Being a threshold counter, however, is not a boundedness property, so different arguments are needed to reason about the behavior of threshold networks.

The proof techniques employed by Aiello *et al.* [2] were purely combinatorial, centered around the concept of a *fooling pair* of inputs [2, Section 3]. In this work, we adapt and extend these techniques to encompass both threshold networks and weak threshold networks (under the regularity assumption) within the structural class of balancing networks whose properties are preserved by the introduction of antitokens and decrement operations.

The rest of this paper is organized as follows. Section 2 provides a framework for our discussion. Section 3 introduces the threshold property and the weak threshold property, and establishes some simple properties. The paper's principal contribution, our results for threshold and weak threshold networks, appears in Sections 4 and 5, respectively. We conclude, in Section 6, with a discussion of our results and some open problems.

2 Framework

2.1 Notation

For any integer $g \geq 2$, $\mathbf{x}^{(g)}$ denotes the vector $\langle x_0, x_1, \ldots, x_{g-1} \rangle^{\mathrm{T}}$. For any vector $\mathbf{x}^{(g)}$, denotes $||\mathbf{x}^{(g)}||_1 = \sum_{i=0}^{g-1} x_i$. We use $\mathbf{0}^{(g)}$ to denote $\langle 0, 0, \ldots, 0 \rangle^{\mathrm{T}}$, a vector with g zero entries; similarly, we use $\mathbf{1}^{(g)}$ to denote $\langle 1, 1, \ldots, 1 \rangle^{\mathrm{T}}$, a vector with g unit entries. A *constant vector* is any vector of the form $c\,\mathbf{1}^{(g)}$, for any constant c.

For any integer x and positive integer δ, denote x div δ and x mod δ the integer quotient and remainder, respectively, of the division of x by δ; note that

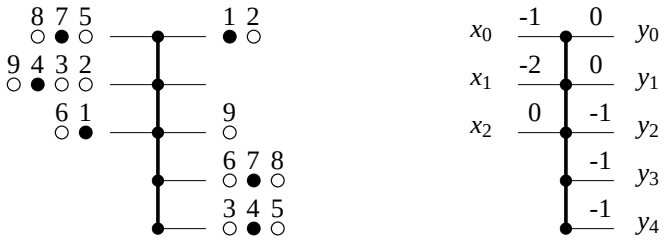

Figure 1: A balancer

$0 \leq x \bmod \delta \leq \delta - 1$, while $x = (x \operatorname{div} \delta)\,\delta + x \bmod \delta$. Clearly, δ divides x if $x \bmod \delta = 0$. Say that δ *divides* $\mathbf{x}^{(g)}$ if δ divides each entry of $\mathbf{x}^{(g)}$.

2.2 Balancers and Balancing Networks

This section is adapted from [2, Sections 2.2 and 2.3]. Balancing networks are constructed from acyclically wired elements, called balancers, that route *tokens* and *antitokens* through the network, and *wires*. Balancers can have multiple input and output wires, in the style of Aharonson and Attiya [1], Felten *et al.* [5], and Hardavellas *et al.* [8]. Following Shavit and Touitou [10] and Busch *et al.* [2], balancers handle both tokens and antitokens. We think of a token and an antitoken as the basic "positive" and "negative" unit, respectively, that are routed through the balancer.

For any pair of positive integers f_{in} and f_{out}, an $(f_{\text{in}}, f_{\text{out}})$*-balancer,* or *balancer* for short, is a routing element receiving tokens and antitokens on f_{in} input wires, numbered $0, 1, \ldots, f_{\text{in}} - 1$, and sending out tokens and antitokens to f_{out} output wires, numbered $0, 1, \ldots, f_{\text{out}} - 1$; f_{in} and f_{out} are called the balancer's *fan-in* and *fan-out,* respectively. A *regular* balancer is an $(f_{\text{in}}, f_{\text{out}})$-balancer such that $f_{\text{in}} = f_{\text{out}}$; that is, fan-in equals fan-out for a regular balancer.

Tokens and antitokens arrive on the balancer's input wires at arbitrary times, and they are output on its output wires. Roughly speaking, a balancer acts like a "generalized" *toggle,* which, on a stream of input tokens and antitokens, alternately forwards them to its output wires, going either down or up on each input token and antitoken, respectively. For clarity, we assume that all tokens and antitokens are distinct.

Figure 1 depicts a balancer with three input wires and five output wires, stretched horizontally; the balancer is stretched vertically. In the left part, tokens and antitokens are denoted with full and empty circles, respectively; the numbering reflects the real-time order of tokens and antitokens in an execution where they traverse the balancer one by one (such an execution is called a *sequential* execution).

For each input index i, $0 \leq i \leq f_{\text{in}} - 1$, we denote by x_i the *balancer input state variable* that stands for the algebraic sum of the numbers of tokens and antitokens that have entered on input wire i; that is, x_i is the number of tokens that have entered on input wire i *minus* the number of antitokens that have entered on input wire i. Denote $\mathbf{x}^{(f_{\text{in}})} = \langle x_0, x_1, \ldots, x_{f_{\text{in}}-1} \rangle^{\text{T}}$; call $\mathbf{x}^{(f_{\text{in}})}$ an *input vector*. For each output index j, $0 \leq j \leq f_{\text{out}} - 1$, we denote by y_j the *balancer output state variable* that stands for the algebraic sum of the numbers of tokens and antitokens that have exited on output wire j; that is, y_j is the number of tokens that have exited on output wire j *minus* the number of antitokens that have exited on output wire j. Denote $\mathbf{y}^{(f_{\text{out}})} = \langle y_0, y_1, \ldots, y_{f_{\text{out}}-1} \rangle^{\text{T}}$; call $\mathbf{y}^{(f_{\text{out}})}$ an *output vector*.

The *configuration* of a balancer at any given time is the tuple $\langle \mathbf{x}^{(f_{\text{in}})}, \mathbf{y}^{(f_{\text{out}})} \rangle$; roughly speaking, the configuration is the collection of its input and output state variables. In the *initial configuration*, all input and output wires are empty; that is, in the initial configuration, $\mathbf{x}^{(f_{\text{in}})} = \mathbf{0}^{(f_{\text{in}})}$, and $\mathbf{y}^{(f_{\text{out}})} = \mathbf{0}^{(f_{\text{out}})}$.

A configuration of a balancer is *quiescent* if there are no tokens or antitokens in the balancer. Note that the initial configuration is a quiescent one. The following formal properties are required for an $(f_{\text{in}}, f_{\text{out}})$-balancer.

1. *Safety property:* in any configuration, a balancer never creates either tokens or antitokens spontaneously.

2. *Liveness property:* for any finite numbers t of tokens and a of antitokens that enter the balancer, the balancer reaches within a finite amount of time a quiescent configuration where $t - e$ tokens and $a - e$ antitokens have exited the network, where e, $0 \leq e \leq \min\{t, a\}$, is the number of tokens and antitokens that are "eliminated" in the balancer.

3. *Step property:* in any quiescent configuration, for any pair of output indices j and k such that $0 \leq j < k \leq f_{\text{out}} - 1$, $0 \leq y_j - y_k \leq 1$.

From the safety and liveness properties, it follows that for any quiescent configuration $\langle \mathbf{x}^{(f_{\text{in}})}, \mathbf{y}^{(f_{\text{out}})} \rangle$ of a balancer, $||\mathbf{x}^{(f_{\text{in}})}||_1 = ||\mathbf{y}^{(f_{\text{out}})}||_1$; that is, in a quiescent configuration, the algebraic sum of tokens and antitokens that exited the balancer is equal to the algebraic sum of tokens and antitokens that entered it. Note that the equality holds even for the case where some of the tokens and antitokens are "eliminated" in the balancer.

We are mostly interested in quiescent configurations of a balancer. For any input vector $\mathbf{x}^{(f_{\text{in}})}$ to balancer b, denote $\mathbf{y}^{(f_{\text{out}})} = b(\mathbf{x}^{(f_{\text{in}})})$ the output vector in the quiescent configuration that b will reach after all tokens and antitokens that entered b have exited; write also $b : \mathbf{x}^{(f_{\text{in}})} \to \mathbf{y}^{(f_{\text{out}})}$ to denote the balancer b.

For any quiescent configuration $\langle \mathbf{x}^{(f_{\text{in}})}, \mathbf{y}^{(f_{\text{out}})} \rangle$ of a balancer $b : \mathbf{x}^{(f_{\text{in}})} \to \mathbf{y}^{(f_{\text{out}})}$, the *state* of the balancer b, denoted $\text{state}_b(\langle \mathbf{x}^{(f_{\text{in}})}, \mathbf{y}^{(f_{\text{out}})} \rangle)$, is defined to be $\text{state}_b(\langle \mathbf{x}^{(f_{\text{in}})}, \mathbf{y}^{(f_{\text{out}})} \rangle) = ||\mathbf{y}^{(f_{\text{out}})}||_1 \bmod f_{\text{out}}$; by definition of quiescent configuration, it follows that $\text{state}_b(\langle \mathbf{x}^{(f_{\text{in}})}, \mathbf{y}^{(f_{\text{out}})} \rangle) = ||\mathbf{x}^{(f_{\text{in}})}||_1 \bmod f_{\text{out}}$. Thus, for the sake of simplicity, we will denote $\text{state}_b(\mathbf{x}^{(f_{\text{in}})}) = \text{state}_b(\langle \mathbf{x}^{(f_{\text{in}})}, \mathbf{y}^{(f_{\text{out}})} \rangle)$.

We remark that the state of an $(f_{\text{in}}, f_{\text{out}})$-balancer is some integer in the set $\{0, 1, \ldots, f_{\text{out}} - 1\}$, which captures the "position" to which it is set as a toggle mechanism. This integer is determined by either the balancer input state variables or the balancer output state variables in the quiescent configuration. Note that the state of the balancer in the initial configuration is 0.

A $(w_{\text{in}}, w_{\text{out}})$-*balancing network* B is a collection of interwired balancers, where output wires are connected to input wires, having w_{in} designated *input wires*, numbered $0, 1, \ldots, w_{\text{in}} - 1$, which are not connected to output wires of balancers, having w_{out} designated *output wires*, numbered $0, 1, \ldots, w_{\text{out}} - 1$, similarly not connected to input wires of balancers, and containing no cycles. A balancing network is *regular* if each of its interwired balancers is regular.

Tokens and antitokens arrive on the network's input wires at arbitrary times, and they traverse a sequence of balancers in the network in a completely asynchronous way till they exit on the output wires of the network.

For each input index i, $0 \leq i \leq w_{\text{in}} - 1$, we denote by x_i the *network input state variable* that stands for the algebraic sum of the numbers of tokens and antitokens that have entered on input wire i; that is, x_i is the difference of the number of tokens that have entered on input wire i *minus* the number of antitokens that have entered on input wire i. Denote $\mathbf{x}^{(w_{\text{in}})} = \langle x_0, x_1, \ldots, x_{w_{\text{in}}-1} \rangle^{\text{T}}$; call $\mathbf{x}^{(w_{\text{in}})}$ an *input vector*. For each output index j, $0 \leq j \leq f_{\text{out}} - 1$, we denote by y_j the *network output state variable* that stands for the algebraic sum of the numbers of tokens and antitokens that have exited on output wire j; that is, y_j is the number of tokens that have exited on output wire j *minus* the number of antitokens that have exited on output wire j. Denote $\mathbf{y}^{(w_{\text{out}})} = \langle y_0, y_1, \ldots, y_{w_{\text{out}}-1} \rangle^{\text{T}}$; call $\mathbf{y}^{(w_{\text{out}})}$ an *output vector*.

The *configuration* of a network at any given time is the tuple of configurations of its individual balancers. In the *initial configuration*, all input and output wires of balancers are empty. The safety and liveness property for a balancing network follow naturally from those of its balancers. Thus, a balancing network eventually reaches a *quiescent configuration* in which all tokens and antitokens that entered the network have either exited the network or pairwise "eliminated" themselves. In any quiescent configuration of B we have $\|\mathbf{x}^{(w_{\text{in}})}\|_1 = \|\mathbf{y}^{(w_{\text{out}})}\|_1$; that is, in a quiescent configuration, the algebraic sum of tokens and antitokens that exited the network is equal to the algebraic sum of tokens and antitokens that entered it.

Naturally, we are interested in quiescent configurations of a network. For any quiescent configuration of a network B with corresponding input and output vectors $\mathbf{x}^{(w_{\text{in}})}$ and $\mathbf{y}^{(w_{\text{out}})}$, respectively, the *state* of B, denoted $\text{state}_B(\mathbf{x}^{(w_{\text{in}})})$, is defined to be the collection of the states of its individual balancers. We remark that we have specified $\mathbf{x}^{(w_{\text{in}})}$ as the single argument of state_B, since $\mathbf{x}^{(w_{\text{in}})}$ uniquely determines all input and output vectors of balancers of B, which are used for defining the states of the individual balancers. Note that the state of the network in its initial configuration is a collection of 0's. For any input vector $\mathbf{x}^{(w_{\text{in}})}$, denote $\mathbf{y}^{(w_{\text{out}})} = B(\mathbf{x}^{(w_{\text{in}})})$ the output vector in the quiescent configuration that B

will reach after all tokens and antitokens that entered B have exited; write also $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ to denote the network B. Clearly, $B(\mathbf{0}^{(w_{\text{in}})}) = \mathbf{0}^{(w_{\text{out}})}$.

2.3 Boundedness Properties

Boundedness properties were introduced by Aiello *et al.* [2]. Our presentation summarizes [2, Section 2.4]. Fix throughout any integer $g \geq 2$.

For any integer $K \geq 1$, the *K-smoothing property* [1, 3] is defined to be the set of all vectors $\mathbf{y}^{(g)}$ such that for any entries y_j and y_k of $\mathbf{y}^{(g)}$, where $0 \leq j,k \leq g-1$, $|y_j - y_k| \leq K$; any vector $\mathbf{y}^{(g)}$ in the K-smoothing property is a *K-smooth vector*. A *smoothing property* is a K-smoothing property, for some integer $K \geq 1$.

A *boundedness property* [2, Section 2.4] is any subset of some smoothing property, that is closed under addition with a constant vector. Thus, a boundedness property is a strict generalization of the smoothing property, since any smoothing property is trivially a boundedness property. Since there are infinitely many smoothing properties, there are infinitely many boundedness properties as well.

The *step property* [3] is defined to be the set of all vectors $\mathbf{y}^{(g)}$ such that for any entries y_j and y_k of $\mathbf{y}^{(g)}$, where $0 \leq j < k \leq g-1$, $0 \leq y_j - y_k \leq 1$; any vector $\mathbf{y}^{(g)}$ in the step property is a *step vector*. An equivalent definition of a step vector $\mathbf{y}^{(g)}$ given in [3] requires that for each index j, $0 \leq j \leq g-1$, $y_j = \lceil (\|\mathbf{y}^{(g)}\|_1 - j)/g \rceil$. Note that any step vector is 1-smooth (but not vice versa); hence. the step property is a (proper) subset of the 1-smoothing property, which is trivially closed under addition with a step vector. It follows that the step property is a boundedness property.

Say that *a vector $\mathbf{y}^{(g)}$ has the (boundedness) property* Π if $\mathbf{y}^{(g)} \in \Pi$. Say that *a balancing network $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ has the (boundedness) property* Π if for every input vector $\mathbf{x}^{(w_{\text{in}})}$, $B(\mathbf{x}^{(w_{\text{in}})}) \in \Pi$. A *counting network* [3] is a balancing network that has the step property. Similarly, a *K-smoothing network* [1, 3] is a balancing network that has the K-smoothing property. The main result of Aiello *et al.* [2] establishes that allowing negative inputs does not spoil the boundedness property of a balancing network.

Theorem 1 (Aiello et al. [2]) *Fix any boundedness property* Π *and a balancing network* $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ *such that* $\mathbf{y}^{(w_{\text{out}})}$ *has the boundedness property* Π *whenever* $\mathbf{x}^{(w_{\text{in}})}$ *is a non-negative vector. Then,* B *has the boundedness property* Π.

2.4 Fooling Pairs

Our presentation follows [2, Section 3]. Say that input vectors $\mathbf{x}_1^{(f_{\text{in}})}$ and $\mathbf{x}_2^{(f_{\text{in}})}$ are a *fooling pair to balancer $b : \mathbf{x}^{(f_{\text{in}})} \to \mathbf{y}^{(f_{\text{out}})}$* [2, Section 3] if $\text{state}_b(\mathbf{x}_1^{(f_{\text{in}})}) =$

$\text{state}_b(\mathbf{x}_2^{(f_{\text{in}})})$; roughly speaking, a fooling pair "drives" the balancer to identical states in the two corresponding quiescent configurations. The concept of a fooling pair can be extended from a single balancer to a network in the natural way. Say that input vectors $\mathbf{x}_1^{(w_{\text{in}})}$ and $\mathbf{x}_2^{(w_{\text{in}})}$ are *a fooling pair to network* $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ if for each balancer b of B, the input vectors of b in quiescent configurations corresponding to $\mathbf{x}_1^{(w_{\text{in}})}$ and $\mathbf{x}_2^{(w_{\text{in}})}$, respectively, are a fooling pair to b; roughly speaking, a fooling pair "drives" all balancers of the network to identical states in the two corresponding quiescent configurations.

The next result relates the output vectors of any balancing network on certain combinations of a fooling pair of input vectors.

Lemma 1 (Aiello et al. [2]) *Consider a balancing network* $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$. *Take any input vectors* $\mathbf{x}_1^{(w_{\text{in}})}$ *and* $\mathbf{x}_2^{(w_{\text{in}})}$ *that are a fooling pair to network* B. *Then, for any input vector* $\mathbf{x}^{(w_{\text{in}})}$, $B(\mathbf{x}_1^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}) - B(\mathbf{x}_1^{(w_{\text{in}})}) = B(\mathbf{x}_2^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}) - B(\mathbf{x}_2^{(w_{\text{in}})})$.

We continue to survey some further combinatorial properties of fooling pairs that we will use in our later proofs. Say that $\mathbf{x}^{(w_{\text{in}})}$ is a *null vector to network* $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ [2, Section 3] if the vectors $\mathbf{x}^{(w_{\text{in}})}$ and $\mathbf{0}^{(w_{\text{in}})}$ are a fooling pair to B. Intuitively, a null vector "hides" itself from the network B in the sense that it does not alter the state of B while traversing it. The next claim determines the output of a balancing network on any non-negative multiple of a null vector.

Lemma 2 (Aiello et al. [2]) *Consider a balancing network* $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$. *Take any vector* $\mathbf{x}^{(w_{\text{in}})}$ *that is null to* B. *Then, for any integer* $k \geq 0$, $B(k\mathbf{x}^{(w_{\text{in}})}) = kB(\mathbf{x}^{(w_{\text{in}})})$.

For any balancing network B, denote $W_{\text{out}}(B)$, the product of the fan-outs of balancers of B. The next claim establishes a sufficient condition involving $W_{\text{out}}(B)$ for a vector to be null to B.

Lemma 3 (Aiello et al. [2]) *Consider a balancing network* $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$. *Assume that* $W_{\text{out}}(B)$ *divides* $\mathbf{x}^{(w_{\text{in}})}$. *Then,* $\mathbf{x}^{(w_{\text{in}})}$ *is a null vector to* B.

3 The Threshold Property and the Weak Threshold Property

In this section, we introduce the threshold property and the weak threshold property; we prove several simple properties of them. Fix throughout any integer $w_{\text{out}} \geq 2$.

Say that a vector $\mathbf{y}^{(w_{\text{out}})}$ is a *threshold vector* [3] if $y_{w_{\text{out}}-1} = \lfloor ||\mathbf{y}^{(w_{\text{out}})}||_1 / w_{\text{out}} \rfloor$. The *threshold property* is the set of all threshold vectors $\mathbf{y}^{(w_{\text{out}})}$. It is straightforward to see that adding a constant vector to a threshold vector yields another

threshold vector; thus, the threshold property is closed under addition with a constant vector. Moreover, take any step vector $\mathbf{y}^{(w_{\text{out}})}$; thus, by equivalent definition of step vector, $y_{w_{\text{out}}-1} = \lceil ||(\mathbf{y}^{(w_{\text{out}})} - (w_{\text{out}} - 1))||_1 / w_{\text{out}} \rceil$. A straightforward calculation reveals that $y_{w_{\text{out}}-1} = \lfloor ||\mathbf{y}^{(w_{\text{out}})}||_1 / w_{\text{out}} \rfloor$. Hence, $\mathbf{y}^{(w_{\text{out}})}$ is a threshold vector. It follows that the step property is a subset of the threshold property.

Say that a vector $\mathbf{y}^{(w_{\text{out}})}$ is a *weak threshold vector* [4] if there is some output index j, possibly $j \neq w_{\text{out}} - 1$, such that $y_j = \lfloor ||\mathbf{y}^{(w_{\text{out}})}||_1 / w_{\text{out}} \rfloor$. The *weak threshold property* is the set of all weak threshold vectors $\mathbf{y}^{(w_{\text{out}})}$. As for the case of threshold vectors, it is straightforward to see that adding a constant vector to a weak threshold vector yields another weak threshold vector; thus, the threshold property is closed under addition with a constant vector. Moreover, the threshold property is a (proper) subset of the weak threshold property.

We start by showing that the threshold property is not a boundedness property in all non-trivial cases.

Proposition 1 *The threshold property is not a boundedness property if and only if* $w_{\text{out}} > 2$.

Proof. Suppose first that $w_{\text{out}} = 2$. We will show that the threshold property is identical to the step property in this case, which is a boundedness property.

Since the step property is a subset of the threshold property, it remains to show that the threshold property is a subset of the step property. Take any threshold vector $\mathbf{y}^{(2)}$; so, $y_1 = \lfloor (y_0 + y_1)/2 \rfloor$. There are two cases to consider. If $y_0 + y_1$ is even, then $y_1 = (y_0 + y_1)/2$, or $y_0 - y_1 = 0$. If $y_0 + y_1$ is odd, then $y_1 = (y_0 + y_1 - 1)/2$, or $y_0 - y_1 = 1$. It follows that in all cases $0 \leq y_0 - y_1 \leq 1$; hence, $\mathbf{y}^{(2)}$ is a step vector, so that the threshold property is a subset of the step property. It follows that the threshold property is identical to the step property for $w_{\text{out}} = 2$. Since the step property is a boundedness property, it follows that the threshold property is a boundedness property for $w_{\text{out}} = 2$, as needed.

Suppose now that $w_{\text{out}} > 2$. Assume, by way of contradiction, that the threshold property is a boundedness property. By definition of boundedness property, this implies that the threshold property is a subset of the K-smoothing property for some integer $K \geq 1$. Consider the threshold vector $\mathbf{y}^{(w_{\text{out}})}$ with $y_{w_{\text{out}}-1} = K+1$, $y_{w_{\text{out}}-2} = (K+1)(w_{\text{out}} - 1)$, and $y_l = 0$ for $0 \leq l < w_{\text{out}} - 2$. Since the threshold property is a subset of the K-smoothing property, it follows that $\mathbf{y}^{(w_{\text{out}})}$ is K-smooth. However, $|y_{w_{\text{out}}-2} - y_{w_{\text{out}}-1}| = |(K+1)(w_{\text{out}} - 2)| = (K+1)(w_{\text{out}} - 2) \geq K+1$, since $w_{\text{out}} > 2$. A contradiction. □

Likewise, we can prove an identical fact for the weak threshold property.

Proposition 2 *The weak threshold property is not a boundedness property if and only if* $w_{\text{out}} > 2$.

Propositions 1 and 2 imply that Theorem 1 does not apply a fortiori to either

threshold networks or weak threshold networks. Hence, in order to show that allowing negative inputs does not spoil either threshold networks or weak threshold networks, different arguments are needed. In the rest of this section, we prepare such arguments.

Say that a vector $\mathbf{y}^{(w_{\text{out}})}$ is a *saturated vector* if $y_{w_{\text{out}}-1} = ||\mathbf{y}^{(w_{\text{out}})}||_1/w_{\text{out}}$. Clearly, any saturated vector is a threshold vector, but not vice versa. We can easily show the following property of saturated vectors.

Proposition 3 *Consider a saturated vector* $\mathbf{y}^{(w_{\text{out}})}$. *Then,* $\tilde{\mathbf{y}}^{(w_{\text{out}})} = -\mathbf{y}^{(w_{\text{out}})}$ *is a saturated vector.*

We continue to show another closure property of the threshold property; more specifically, we prove that the threshold property is closed under addition with a saturated vector.

Proposition 4 *Consider a threshold vector* $\mathbf{y}^{(w_{\text{out}})}$ *and a saturated vector* $\tilde{\mathbf{y}}^{(w_{\text{out}})}$. *Then,* $\mathbf{y}^{(w_{\text{out}})} + \tilde{\mathbf{y}}^{(w_{\text{out}})}$ *is a threshold vector.*

Proof. Clearly,

$$\begin{aligned} y_{w_{\text{out}}-1} + \widetilde{y}_{w_{\text{out}}-1} &= \left\lfloor \frac{||\mathbf{y}^{(w_{\text{out}})}||_1}{w_{\text{out}}} \right\rfloor + \frac{||\tilde{\mathbf{y}}^{(w_{\text{out}})}||_1}{w_{\text{out}}} \\ &\quad \text{(since } \mathbf{y}^{(w_{\text{out}})} \text{ is threshold and } \tilde{\mathbf{y}}^{(w_{\text{out}})} \text{ is saturated)} \\ &= \left\lfloor \frac{||\mathbf{y}^{(w_{\text{out}})}||_1}{w_{\text{out}}} + \frac{||\tilde{\mathbf{y}}^{(w_{\text{out}})}||_1}{w_{\text{out}}} \right\rfloor \\ &= \left\lfloor \frac{||\mathbf{y}^{(w_{\text{out}})} + \tilde{\mathbf{y}}^{(w_{\text{out}})}||_1}{w_{\text{out}}} \right\rfloor, \end{aligned}$$

so that $\mathbf{y}^{(w_{\text{out}})} + \tilde{\mathbf{y}}^{(w_{\text{out}})}$ is a threshold vector, as needed. □

By Proposition 3, the following is an immediate consequence of Proposition 4.

Corollary 1 *Consider a threshold vector* $\mathbf{y}^{(w_{\text{out}})}$, *and a saturated vector* $\tilde{\mathbf{y}}^{(w_{\text{out}})}$. *Then,* $\mathbf{y}^{(w_{\text{out}})} - \tilde{\mathbf{y}}^{(w_{\text{out}})}$ *is a threshold vector.*

Say that a vector $\mathbf{y}^{(w_{\text{out}})}$ is a *weak saturated vector* if there is some output index j, possibly $j \neq w_{\text{out}} - 1$, such that $y_j = ||\mathbf{y}^{(w_{\text{out}})}||_1/w_{\text{out}}$. Clearly, any saturated vector is a weak saturated vector, but not vice versa.

The threshold property and the weak threshold property give rise to corresponding networks in the natural way. A *threshold network* [3] is a balancing network $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ that has the threshold property. Roughly speaking, a threshold network detects input "chunks" of size w_{out} on the output wire $w_{\text{out}} - 1$, called the *threshold wire*.

A *weak threshold network* [4] is a balancing network $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ that has the weak threshold property. Thus, like threshold networks, weak threshold networks detect, on each input vector $\mathbf{x}^{(w_{\text{in}})}$, input "chunks" of size w_{out} on some output wire $j = j(\mathbf{x}^{(w_{\text{in}})})$, $0 \leq j \leq w_{\text{in}} - 1$, called the *threshold wire for input* $\mathbf{x}^{(w_{\text{in}})}$. However, unlike threshold networks, it is possible that threshold wires for different input vectors be different.

4 Threshold Networks

In this section, we establish that the threshold property is preserved by the introduction of antitokens. We start by proving a technical claim.

Proposition 5 *Take a threshold network $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$. Assume that $W_{\text{out}}(B)$ divides $\mathbf{x}^{(w_{\text{in}})}$. Then, $\mathbf{y}^{(w_{\text{out}})}$ is a saturated vector.*

Proof. Since $W_{\text{out}}(B)$ divides $\mathbf{x}^{(w_{\text{in}})}$, Lemma 3 implies that $\mathbf{x}^{(w_{\text{in}})}$ is a null vector to network B. Thus, by Lemma 2, $B(w_{\text{out}}\,\mathbf{x}^{(w_{\text{in}})}) = w_{\text{out}}\,B(\mathbf{x}^{(w_{\text{in}})}) = w_{\text{out}}\,\mathbf{y}^{(w_{\text{out}})}$. Since B is a threshold network, it follows that $w_{\text{out}}\,\mathbf{y}^{(w_{\text{out}})}$ is a threshold vector. By definition of threshold vector, this implies that $w_{\text{out}}\,y_{w_{\text{out}}-1} = \lfloor w_{\text{out}} ||\mathbf{y}^{(w_{\text{out}})}||_1 / w_{\text{out}} \rfloor = ||\mathbf{y}^{(w_{\text{out}})}||_1$; hence, $y_{w_{\text{out}}-1} = ||\mathbf{y}^{(w_{\text{out}})}||_1 / w_{\text{out}}$. By definition of saturated vector, this implies that $\mathbf{y}^{(w_{\text{out}})}$ is a saturated vector, as needed. □

Proposition 5 provides a sufficient condition on the input vector of a threshold network, which involves structural parameters of the network itself, for the corresponding output vector to be a saturated vector. Thus, Proposition 5 is reminiscent, in both its statement and proof, to [2, Proposition 4.1], which provides a corresponding sufficient condition for the output vector of a balancing network that has any boundedness property to be a constant vector. Hence, Proposition 5 establishes an analogy between constant vectors with respect to a network that has any boundedness property, and saturated vectors with respect to a threshold network. We continue with using Proposition 5 to show our equivalence result for threshold networks.

Theorem 2 (Threshold networks support decrements) *Take a balancing network $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ such that $\mathbf{y}^{(w_{\text{out}})}$ is a threshold vector whenever $\mathbf{x}^{(w_{\text{in}})}$ is a non-negative vector. Then, B is a threshold network.*

Proof. Consider any arbitrary input vector $\mathbf{x}^{(w_{\text{in}})}$. We will show that $B(\mathbf{x}^{(w_{\text{in}})})$ is a threshold vector.

Construct from $\mathbf{x}^{(w_{\text{in}})}$ an input vector $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ such that for each index i, $0 \leq i \leq w_{\text{in}} - 1$, $\widetilde{x_i}$ is the least multiple of $W_{\text{out}}(B)$ such that $\widetilde{x_i} + x_i \geq 0$. Clearly, $W_{\text{out}}(B)$ divides $\tilde{\mathbf{x}}^{(w_{\text{in}})}$. By Proposition 5, $B(\tilde{\mathbf{x}}^{(w_{\text{in}})})$ is a saturated vector, while by Lemma 3,

$\tilde{\mathbf{x}}^{(w_{\text{in}})}$ is a null vector to network B. We apply Lemma 1 with $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ for $\mathbf{x}_1^{(w_{\text{in}})}$, $\mathbf{0}^{(w_{\text{in}})}$ for $\mathbf{x}_2^{(w_{\text{in}})}$, and $\mathbf{x}^{(w_{\text{in}})}$ for $\mathbf{x}^{(w_{\text{in}})}$; we obtain that

$$\begin{aligned} B(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}) &= B(\mathbf{x}^{(w_{\text{in}})}) + B(\tilde{\mathbf{x}}^{(w_{\text{in}})}) - B(\mathbf{0}^{(w_{\text{in}})}) \\ &= B(\mathbf{x}^{(w_{\text{in}})}) + B(\tilde{\mathbf{x}}^{(w_{\text{in}})}) , \end{aligned}$$

so that $B(\mathbf{x}^{(w_{\text{in}})}) = B(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}) - B(\tilde{\mathbf{x}}^{(w_{\text{in}})})$.

Since $\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}$ is a non-negative input vector, it follows, by assumption on B, that $B(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})})$ is a threshold vector. Since $B(\tilde{\mathbf{x}}^{(w_{\text{in}})})$ is a saturated vector, Corollary 1 implies that $B(\mathbf{x}^{(w_{\text{in}})})$ is a threshold vector, as needed. □

The proof of Theorem 2 used Lemmas 1 and 3, which, however, hold for *any* balancing network; it used Corollary 1, which determines a special class of vectors, namely, the saturated vectors, to provide closure under subtraction to the threshold property; finally, it used Proposition 5, which provides a sufficient condition for the output of a threshold network to be a saturated vector. We remark that the general structure of the proof of Theorem 2 closely follows the one of [2, Theorem 4.2] (quoted as Theorem 1 in this paper).

5 Weak Threshold Networks

In this section, we establish that the weak threshold property is preserved by the introduction of antitokens, under some technical assumption on the structure of weak threshold networks.

There is a difficulty when we try to extend the proof Theorem 2 to weak threshold networks. Specifically, the analog of Corollary 1 for weak threshold vectors and (weak) saturated vectors does not hold in general. For example, for the weak threshold vector $\mathbf{y}^{(3)} = \langle 9,4,1\rangle^{(\mathrm{T})}$ and the (weak) saturated vector $\tilde{\mathbf{y}}^{(3)} = \langle 2,3,1\rangle^{(\mathrm{T})}$ we have that $\mathbf{y}^{(3)} - \tilde{\mathbf{y}}^{(3)}$ is not a weak threshold vector. Therefore, some additional care is needed in extending the proof of Theorem 2 (which relies on Corollary 1) to weak threshold networks.

We have only been able to extend Theorem 2 to the case of *regular* weak threshold networks, namely weak threshold networks such that each of their balancers has the same fan-in and fan-out. We will need a simple technical claim which has been shown by Herlihy *et al* [9, Lemma 4.1] for networks consisting of balancers with fan-in and fan-out equal to two, and which, apparently, holds for any regular balancing network.

Lemma 4 (Herlihy et al. [9]) *Take a regular balancing network* $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$. *Then, for any integer* $c > 0$, $B(c\,\mathbf{1}^{(w_{\text{in}})}) = c\,\mathbf{1}^{(w_{\text{out}})}$.

Roughly speaking, Lemma 4 asserts that if exactly c tokens enter on each input wire, then exactly c tokens will exit from each output wire. We are now ready to show that regular weak threshold networks support decrements.

Theorem 3 (Regular weak threshold networks support decrements) *Take a regular balancing network $B : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ such that $\mathbf{y}^{(w_{\text{out}})}$ is a weak threshold vector whenever $\mathbf{x}^{(w_{\text{in}})}$ is a non-negative vector. Then, B is a weak threshold network.*

Proof. Consider any arbitrary input vector $\mathbf{x}^{(w_{\text{in}})}$. We will show that $B(\mathbf{x}^{(w_{\text{in}})})$ is a weak threshold vector.

Construct from $\mathbf{x}^{(w_{\text{in}})}$ a constant input vector $\tilde{\mathbf{x}}^{(w_{\text{in}})} = c\,\mathbf{1}^{(w_{\text{in}})}$, where c is the least multiple of $W_{\text{out}}(B)$ such that for each index i, $0 \le i \le w_{\text{in}} - 1$, $c + x_i \ge 0$. (Alternatively, c is the *maximum* $\tilde{x}_i$, $0 \le i \le w_{\text{in}} - 1$, where $\tilde{x}_i$ is the least multiple of $W_{\text{out}}(B)$ such that $\tilde{x}_i + x_i \ge 0$.) Clearly, $W_{\text{out}}(B)$ divides $\tilde{\mathbf{x}}^{(w_{\text{in}})}$. By Lemma 3, $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ is a null vector to network B. We apply Lemma 1 with $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ for $\mathbf{x}_1^{(w_{\text{in}})}$, $\mathbf{0}^{(w_{\text{in}})}$ for $\mathbf{x}_2^{(w_{\text{in}})}$, and $\mathbf{x}^{(w_{\text{in}})}$ for $\mathbf{x}^{(w_{\text{in}})}$; we obtain that

$$\begin{aligned} B(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}) &= B(\mathbf{x}^{(w_{\text{in}})}) + B(\tilde{\mathbf{x}}^{(w_{\text{in}})}) - B(\mathbf{0}^{(w_{\text{in}})}) \\ &= B(\mathbf{x}^{(w_{\text{in}})}) + B(\tilde{\mathbf{x}}^{(w_{\text{in}})})\,, \end{aligned}$$

so that $B(\mathbf{x}^{(w_{\text{in}})}) = B(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}) - B(\tilde{\mathbf{x}}^{(w_{\text{in}})})$.

Since $\tilde{\mathbf{x}}^{(w_{\text{in}})} = c\,\mathbf{1}^{(w_{\text{in}})}$, it follows by Lemma 4 that $B(\tilde{\mathbf{x}}^{(w_{\text{in}})}) = c\,\mathbf{1}^{(w_{\text{out}})}$. Since $\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}$ is a non-negative input vector, it follows, by assumption on B, that $B(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})})$ is a weak threshold vector. Let j be the threshold wire for $B(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})})$. Since $B(\tilde{\mathbf{x}}^{(w_{\text{in}})}) = c\,\mathbf{1}^{(w_{\text{out}})}$, $B(\tilde{\mathbf{x}}^{(w_{\text{in}})})_j = c$ so that

$$\begin{aligned} B(\mathbf{x}^{(w_{\text{in}})})_j &= B(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})})_j - B(\tilde{\mathbf{x}}^{(w_{\text{in}})})_j \\ &= B(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})})_j - c \\ &= \left\lfloor \frac{||\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}||_1}{w_{\text{out}}} \right\rfloor - c \\ &\quad \text{(since } B(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}) \text{ is weak threshold)} \\ &= \left\lfloor \frac{||\tilde{\mathbf{x}}^{(w_{\text{in}})}||_1 + ||\mathbf{x}^{(w_{\text{in}})}||_1}{w_{\text{out}}} \right\rfloor - c \\ &= \left\lfloor \frac{c\,w_{\text{out}} + ||\mathbf{x}^{(w_{\text{in}})}||_1}{w_{\text{out}}} \right\rfloor - c \\ &\quad \text{(by definition of } \tilde{\mathbf{x}}^{(w_{\text{in}})}) \\ &= c + \left\lfloor \frac{||\mathbf{x}^{(w_{\text{in}})}||_1}{w_{\text{out}}} \right\rfloor - c \\ &= \left\lfloor \frac{||\mathbf{x}^{(w_{\text{in}})}||_1}{w_{\text{out}}} \right\rfloor . \end{aligned}$$

It follows that $B(\mathbf{x}^{(w_{\text{in}})})$ is a weak threshold vector, as needed. □

6 Conclusion

We have shown that any balancing network that satisfies the threshold property on all non-negative input vectors, it will also satisfy it for any arbitrary input vector. We have also shown a corresponding fact for the weak threshold property, assuming that the network is regular. It would be interesting to see whether or not the regularity assumption can be dropped for weak threshold networks. Our results imply that, in designing and verifying threshold and (regular) weak threshold networks, it is possible to restrict attention to non-negative input vectors, which conveniences design and simplifies proofs.

Our proofs have built on the combinatorial techniques introduced in [2]. It would still be interesting to find further applications of these techniques to other classes of balancing networks. The major problem still left open by our work is to formally characterize all properties of balancing networks that are preserved under the introduction of decrement operations via antitokens.

References

[1] E. Aharonson and H. Attiya, "Counting Networks with Arbitrary Fan-Out," *Distributed Computing*, Vol. 8, pp. 163–169, 1995.

[2] W. Aiello, C. Busch, M. Herlihy, M. Mavronicolas, N. Shavit, and D. Touitou, "Supporting Increment and Decrement Operations in Balancing Networks," *Proceedings of the 16th International Symposium on Theoretical Aspects of Computer Science*, G. Meinel and S. Tison eds., pp. 377–386, Vol. 1563, Lecture Notes in Computer Science, Springer-Verlag, Trier, Germany, March 1999.

[3] J. Aspnes, M. Herlihy and N. Shavit, "Counting Networks," *Journal of the ACM*, Vol. 41, No. 5, pp. 1020–1048, September 1994.

[4] C. Busch and M. Mavronicolas, "Impossibility Results for Weak Threshold Networks," *Information Processing Letters*, Vol. 63, No. 2, pp. 85–90, July 1997.

[5] 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.

[6] D. Grunwald and S. Vajracharya, "Efficient Barriers for Distributed Shared Memory Computers," *Proceedings of the 8th International Parallel Processing Symposium*, IEEE Computer Society Press, April 1994.

[7] R. Gupta and C. R. Hill, "A Scalable Implementation of Barrier Synchronization Using an Adaptive Tree," *International Journal of Parallel Programming*, Vol. 18, No. 3, pp. 161–180, June 1989.

[8] 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.

[9] M. Herlihy, N. Shavit, and O. Waarts, "Linearizable Counting Networks," *Distributed Computing*, Vol. 9, pp. 193–203, 1996.

[10] N. Shavit and D. Touitou, "Elimination Trees and the Construction of Pools and Stacks," *Theory of Computing Systems*, Vol. 30, No. 6, pp. 545–570, November/December 1997.

[11] G. V. Wilson, *Practical Parallel Programming*, The MIT Press, 1995.

Costas Busch is a graduate student in the Department of Computer Science, Brown University, Providence, RI, USA. Email: cb@cs.brown.edu

Neophytos Demetriou is an undergraduate student in the Department of Computer Science, University of Cyprus, Nicosia, Cyprus. Email: k2pts@spidernet.com.cy

Maurice Herlihy is affiliated with the Department of Computer Science, Brown University, Providence, RI, USA. Email: mph@cs.brown.edu

Marios Mavronicolas is affiliated with the Department of Computer Science and Engineering, University of Connecticut, Storrs, CT, USA. Part of his work was performed while at Department of Computer Science, University of Cyprus, Nicosia, Cyprus. Email: mavronic@engr.uconn.edu