Small-Depth Counting Networks

Michael Klugerman[*]
Mathematics Department
Massachusetts Institute of Technology
Cambridge, MA 02139
klugermn@theory.lcs.mit.edu

C. Greg Plaxton[†]
Department of Computer Science
University of Texas at Austin
Austin, TX 78712
plaxton@cs.utexas.edu

Abstract

Generalizing the notion of a sorting network, Aspnes, Herlihy, and Shavit recently introduced a class of so-called "counting" networks, and established an $O(\lg^2 n)$ upper bound on the depth complexity of such networks. Their work was motivated by a number of practical applications arising in the domain of asynchronous shared memory machines. This paper continues the analysis of counting networks, providing a number of new upper bounds. In particular, we present an explicit construction of an $O(c^{\lg^* n} \lg n)$-depth counting network, a randomized construction of an $O(\lg n)$-depth network (that works with extremely high probability), and using the random construction we present an existential proof of a deterministic $O(\lg n)$-depth network. The latter result matches the trivial $\Omega(\lg n)$-depth lower bound to within a constant factor.

1 Introduction

The notion of a "counting network" was recently introduced by Aspnes, Herlihy, and Shavit [4], where it was shown that such networks can be simulated efficiently on an asynchronous shared memory machine to implement counters [9, 10, 11, 16], producer/consumer buffers, and synchronization barriers. The counting network provides a means for the processors of a parallel machine to obtain successive values from a counter. These values can then be used to obtain unique keys to various resources shared by the processors, to allocate tasks evenly among the processors, or to synchronize processors when necessary.

One solution to the counting problem is to use a single shared Fetch-and-Increment variable that is incremented each time a processor makes a request. This can lead to high memory contention when a large number of processors are making requests simultaneously. Counting networks provide a means by which this contention can be significantly reduced and thus allow for a much higher degree of concurrency. More specifically, a number of shared variables are used to implement a single counter in such a way that contention is reduced and a processor incrementing the counter need only access a small number of memory locations, thus providing fast response time and high throughput.

Counting networks are modelled after sorting networks [2, 5, 7, 13] in that they are composed of 2-input 2-output components called balancers (rather than comparators). A balancer takes in tokens along both input wires and, acting like a toggle, outputs these tokens alternately along the top and bottom output wires (see Figure 1). As in the case of a comparator in a comparator network, balancers are used to construct a balancing network with an arbitrary number of input wires (and an equal number of output wires) called a balancing network. A counting network is a balancing network which is able to "count" any number of tokens that enter the network (see Section 2 for a more complete definition). The counting network can then be implemented in software on a shared memory machine by associating a memory location with each balancer [4].

An important measure of the efficiency of a counting network is its depth. This is because the depth of the network is equal to the number of memory locations that a processor must access before its increment request has been fulfilled. In this paper, we present a number of constructions for counting networks of small depth.

Aspnes, Herlihy, and Shavit [4] provide two $O(\lg^2 n)$-depth families of n-input counting networks by proving that the balancing network isomorphic to Batcher's bitonic sorting network [5, 7, 13] and isomorphic to the balanced periodic sorting network of Dowd, Perl, Rudolph, and Saks [8] are counting networks. Later, Klugerman [12] gave an $O(\lg n \lg\lg n)$-depth construction.

*Supported by the Defense Advanced Research Projects Agency under Contracts N00014-87-K-825 and N00014-89-J-1988, the Air Force under Contract AFOSR-89-0271, and the Army under Contract DAAL-03-86-K-0171

†Supported by NSF Research Initiation Award CCR-9111591, and Texas Advanced Research Program (TARP) Award #003658480.

Our main result is a proof of the existence of an $O(\lg n)$-depth counting network where n is the number of input wires. This result answers the question posed in [4], which asks whether such an optimal-depth counting network exists. The technique used to obtain this result involves constructing a set of networks $\mathcal{N}^*$ such that for any fixed input sequence I, if a network $\mathcal{N}$ is chosen uniformly at random from $\mathcal{N}^*$, then $\mathcal{N}$ will count I with extremely high probability. "Good" networks are then chosen non-uniformly from $\mathcal{N}^*$ and are used to construct a deterministic counting network with logarithmic depth. A similar technique has recently been used by Ajtai, Komlós and Szemerédi [6] to improve the constant factor in their $O(\lg n)$ depth sorting network, and by Plaxton [15] to obtain a $2^{O(\sqrt{\lg\lg n})}\lg n$-depth sorting network from an $O(\lg n)$-depth random sorting network [14] that sorts with extremely high probability. The other result in this paper is an explicit construction of a counting network of depth $O(c^{\lg^* n}\lg n)$ (for some positive constant c), which represents an improvement over previously known constructions.

The remainder of the paper is organized as follows. In Section 2, we define counting networks and other necessary terminology. In Section 3, we provide lemmas about balancing and counting networks that will be of use in later sections. In Section 4, we present constructions of balancing networks that will be used as components in the constructions of our counting networks. In Section 5, we present the $O(\lg n)$-depth random counting network. In Section 6, we use the random network to construct a non-uniform deterministic counting network. In Section 7, we describe the explicit construction of the $O(c^{\lg^* n}\lg n)$-depth counting network. Finally, in Section 8 we offer some concluding remarks.

2 Definitions

A counting network is a balancing network with additional properties in the same way that a sorting network [2, 5, 7, 13] is a comparator network with additional properties. The elementary building block of a balancing network is a balancer, rather than a comparator. A **balancer** is a 2-input, 2-output device (see Figure 1).

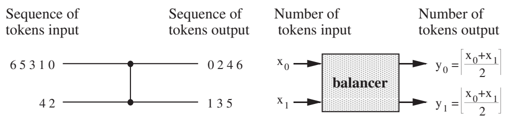

Figure 1: A balancer.

A balancing network is an acyclic circuit made up of balancers, just as a comparator network is an acyclic circuit made up of comparators. Whereas a comparator receives a number along each input wire and outputs them in sorted order, a balancer receives multiple anonymous tokens along both input wires in an asynchronous fashion and sends the tokens alternately to the upper and lower output wires. Thus, if a total of m tokens are received by the two input wires of a particular balancer, that balancer emits $\lceil m/2 \rceil$ tokens via the top output wire and $\lfloor m/2 \rfloor$ tokens via the bottom output wire (see Figure 1).

An n-input **balancing network** has n input wires and n output wires. In the remainder of this paper we will refer to the input (resp., output) wires as $X_0, \ldots, X_{n-1}$ (resp., $Y_0, \ldots, Y_{n-1}$). We denote the number of tokens received by wire X_i in a particular execution as x_i, while we denote the number of tokens output from wire Y_i as y_i.

Each token enters the network through some input wire and then passes through one or more balancers before arriving at an output wire. Figure 2 provides an example of a balancing network.

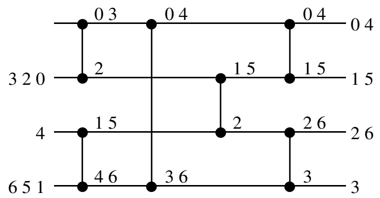

Figure 2: A sequential execution of a counting (balancing) network. Tokens are labelled with distinct numbers.

Two useful measures of the complexity of a balancing network are its size and depth. The **size** of a balancing network is the number of balancers in the network. In order to define the depth of a balancing network, we must first define the depth of each balancer and wire in the network. The input wires of a balancer network have depth 0. Given this, and because a balancing network is required to be acyclic, the following pair of rules can be used to determine the depth of all balancers and all remaining wires in the network: (i) the depth of a balancer is 1 greater than the maximum depth of its two input wires, and (ii) the depth of an output wire of a given balancer is equal to the depth of that balancer. The **depth** of a balancing network may then be defined as the maximum depth of any output wire in the network.

We now define two special types of balancing networks that guarantee certain properties concerning the nature of the output.

An n-input **smoothing network** is a balancing network such that for any input sequence, $|y_i - y_j| \le 1$, $0 \le i < j < n$. The output of such a network is said to have a **smoothed shape**.

An n-input **counting network** is a balancing network such that for any input sequence, $1 \geq y_i - y_j \geq 0$, $0 \leq i < j < n$. Note that a counting network is a restricted form of smoothing network. The output of such a network is said to have a **counted shape**.

One can easily see from the definition of a counting network that assigning the value $in + j$ to the ith token output from wire j of a counting network gives the tokens distinct integer values ranging from 0 to $p - 1$ where p is the total number of tokens input to the network. Thus, the tokens are said to be "counted". Figure 2 shows a sequence of tokens passing through a counting network. Note that the output has a counted shape.

In our constructions we will find it useful to consider balancing networks that count certain restricted classes of input sequences. An input sequence $x_0, \ldots x_{n-1}$ is k**-smoothed** if $|x_i - x_j| \leq k$, $0 \leq i < j < n$. A k**-counter** is a balancing network that counts any k-smoothed input sequence.

We will sometimes refer to the binary representation of the address of a wire. The expression $\mathrm{bin}(i, k)$ will be used to denote the k-bit binary represention of the integer i, for $0 \leq i < 2^k$.

3 Basic lemmas

In this section we present some elementary lemmas about balancing networks that will be useful in later proofs.

The first lemma shows that given a specific balancing network and an input sequence $x_0, \ldots, x_{n-1}$ to this network, the output sequence $y_0, \ldots, y_{n-1}$ is well-defined. This is a simple extension of the serialization lemma given in [4].

Lemma 3.1 The order in which tokens pass through the network does not affect the number of tokens output on each wire.

Proof: We prove the claim by induction on the depth of the network. If the depth is 0, then each output wire corresponds to a single input wire and the result is immediate. Now assume that the claim holds for balancing networks of depth k, $k \geq 0$, and consider any maximum-depth balancer x in a network of depth $k + 1$. By the induction hypothesis, the number of tokens arriving along each input wire of x is well-defined. Applying the definition of a balancer, we see that the number of tokens received by each of the two outputs of x is also well-defined. ■

Since we are only concerned about the number of tokens output per wire from the network and not the specific ordering of the tokens, as a result of Lemma 3.1, we can choose the order in which we wish tokens to traverse a network when we analyze the properties of a specific network.

The following lemma is stated in [4] and is very useful in our constructions. Given the $O(\lg n)$-depth AKS sorting network result [2], this lemma shows that the problem of constructing a small-depth counting network can be reduced to that of constructing a small-depth smoothing network.

Lemma 3.2 A sorting network (with comparators replaced by balancers) when applied to the output of a smoothing network, produces a counting network. ■

In this paper only counting networks having a number of input wires input wires equal to some power of 2 are constructed. However, Klugerman [12] and Aharonson and Attiya [1] have independently shown that only networks of this type exist:

Lemma 3.3 Every counting network must have 2^k input wires for some integer $k \geq 0$. ■

4 Building blocks

In this section, we describe three balancing networks that will provide us with useful "subroutines" for the counting network constructions of Sections 5 through 7.

4.1 The butterfly balancing network

Definition 4.1 A 2^k*-input butterfly balancing network*, where k is a nonnegative integer, may be defined recursively as follows. If $k = 0$, then it is a single wire. If $k > 0$, then it is constructed from two 2^{k-1}-input butterfly balancing networks and an additional level of 2^{k-1} balancers as indicated in Figure 3.

Note that a 2^k-input butterfly balancing network has depth k.

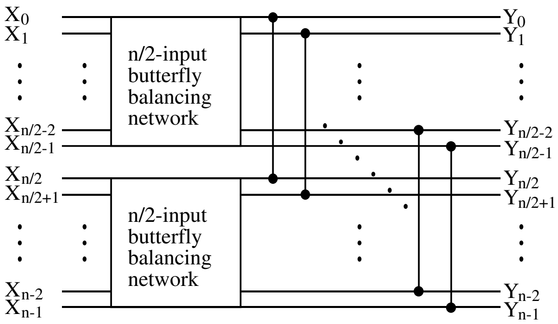

Figure 3: An n-input butterfly balancing network where $n = 2^k$.

Lemma 4.1 Consider a 2^k-input butterfly balancing network. Let i and j denote two k-bit integers that differ in a single bit position, with $i < j$. Then for any input sequence, either $y_i = y_j$ or $y_i = y_j + 1$.

Proof: We prove the claim by induction on k. If $k = 0$, there is nothing to prove. Now assume that the claim holds for $k < d$, $d \geq 1$, and consider the case $k = d$. If i and j differ in bit position d, then the result is immediate since outputs i and j are connected to the same depth d balancer. Thus, we may assume that i and j differ in some bit position a, $0 \leq a < d$. Let i' (resp., j') denote the integer having the same binary representation as i (resp., j), except in bit position d. Note that outputs Y_i and $Y_{i'}$ (resp., Y_j and $Y_{j'}$) represent the two outputs of some balancer b_0 (resp., b_1) at depth d. Let a total of r_0 (resp., r_1) tokens be received by balancer b_0 (resp., b_1). By the induction hypothesis, $r_1 \leq r_0 \leq r_1 + 2$. If $i < i'$, then $y_i = \lceil r_0/2 \rceil$ and $y_j = \lceil r_1/2 \rceil$, so that either $y_i = y_j$ or $y_i = y_j + 1$. The case $i > i'$ is similar. ∎

Lemma 4.1 shows that the output sequence from a butterfly balancing network has a hypercube-like structure. When Y_i is mapped to node $\mathrm{bin}(i, k)$ of a 2^k-node hypercube, the output wire containing the most tokens corresponds to 0^k (i.e., Y_0) while the output wire with the fewest tokens corresponds to 1^k (i.e., Y_{2^k-1}). Furthermore, if one considers a sequence of outputs $\{y_{j_0=2^k-1}, y_{j_1}, \ldots, y_{j_{k-1}}, y_{j_k=0}\}$ corresponding to a chain in the hypercube (when viewed as a Boolean lattice) beginning with the node 1^k and ending with the node 0^k, then $0 \leq y_{j_{i+1}} - y_{j_i} \leq 1$, $0 \leq i < k$.

Corollary 4.1.1 For any input sequence to the 2^k-input butterfly balancing network, and any pair of output wires i and j, we have $|y_i - y_j| \leq k$.

Proof: An immediate consequence of Lemma 4.1, along with the fact that k bits are used to describe the address of each wire. ∎

In our "randomized" construction of Section 5, we will prove that after the tokens pass through a particular stage of the network containing a butterfly balancing network, with high probability all of the output wires corresponding to nodes in the "middle" levels of the hypercube (i.e., those with a large number of both 0's and 1's in the binary representation of their addresses), will contain very close to μ tokens where μ is the average number of tokens per input wire. Let G denote the set of output wires of a 2^k-input butterfly balancing network whose addresses have binary representations containing at least ϵk 0's and ϵk 1's, $0 \leq \epsilon \leq 1$. We say that G is the set of **ϵ-good** wires while those not in G are **ϵ-bad**. The ϵ-good wires are considered the "middle" levels of the hypercube.

4.2 Pairing networks

Definition 4.2 An (m, n, k)-*pairing network*, for nonnegative integers m, n, and k satisfying $n \geq m2^k$, is an n-input, depth-k balancing network constructed as follows. Upon entry to the network, m wires have been designated as *bad* inputs while the remaining $n - m$ wires have been designated as *good inputs*. Similarly, the outputs of the network will be partitioned into a set of $m2^k$ *bad outputs*, and $n - m2^k$ *good outputs*. If $k = 0$, the network consists of n wires, and the bad (resp., good) outputs simply correspond to the bad (resp., good) inputs. If $k > 0$, the desired network $\mathcal{N}$ consists of an $(m, n, k-1)$-pairing network $\mathcal{N}'$ (note that $n \geq m2^k$ implies $n \geq m2^{k-1}$) followed by an additional level of $m2^{k-1}$ balancers pairing (in an arbitrary fashion) each of the bad outputs of $\mathcal{N}'$ with a good output of $\mathcal{N}'$. The bad outputs of $\mathcal{N}$ are exactly the $m2^k$ outputs of this set of $m2^{k-1}$ balancers (see also Figure 4).

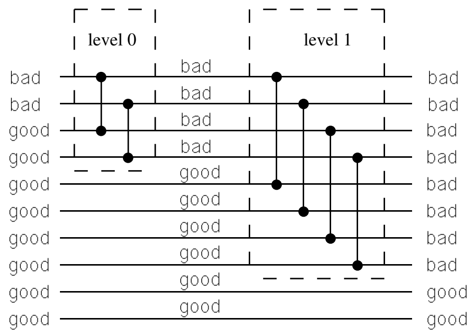

Figure 4: A $(2, 10, 2)$-pairing network.

The following lemma shows that a pairing network can be used to smooth the bad inputs using the good inputs:

Lemma 4.2 If the input sequence to an (m, n, k)-pairing network $\mathcal{N}$ is such that for some pair of integers $a \leq b$, every good (resp., bad) input receives between a and b (resp., $a - 2^k$ and $b + 2^k$) tokens, then every good (resp., bad) output receives between a and b (resp., $a - 1$ and $b + 1$) tokens.

Proof: The good output wires are not connected to balancers so they will clearly output between a and b tokens. With respect to the bad output wires, we will now argue that no bad output receives more than $b+1$ tokens; a symmetric argument may be used show that no bad output receives fewer than $a - 1$ tokens. In fact, we will prove the following stronger claim: No wire at depth i receives more than $b + 2^{k-i}$ tokens, $0 \leq i \leq k$. This claim may be proven by induction on i. The base case, $i = 0$, is immediate. For the induction step, note that every wire at depth $i+1$, $0 \leq i < k$, is an output of a balancer that received at most $b + 2^{k-i}$ tokens (by the induction hypothesis) on one input and at most b tokens on the other input (since one of the inputs must be good). Each of the outputs of such a balancer will receive at most $\lceil (2b + 2^{k-i})/2 \rceil = b + 2^{k-i-1}$ tokens, as required. ∎

4.3 A deterministic 2-counter

In this section, we present an explicit construction of a 2-counter. Our construction makes use of two other primitives. For any positive integer n, a $2n$-input *ladder network* is a depth 1 balancing network constructed by connecting X_i to X_{2n-1-i} with a balancer, $0 \le i < n$ (as indicated in Figure 5). We will also make use of *sorting networks* as "subroutines" in our counting network constructions. Any sorting network may be used at these locations by replacing the comparators with balancers. To obtain small depth we will use the $O(\lg n)$-depth AKS sorting network [2].

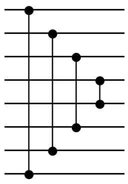

Figure 5: An 8-input ladder.

Lemma 4.3 If a counted shape is input to the top n wires of a $2n$-input ladder, and another counted shape is input to the bottom n wires, then the output of the ladder is smoothed.

Proof: There exist integers n_0 and k_0 such that n_0 of the top n inputs receive k_0 tokens, and the remaining $n - n_0$ top inputs receive $k_0 + 1$ tokens. Let n_1 and k_1 be defined similarly for the bottom inputs. If $n_0 \ge n - n_1$, then every output of the ladder will receive at least $\lfloor \frac{k_0+k_1}{2} \rfloor$ and at most $\lceil \frac{k_0+k_1+1}{2} \rceil = \lfloor \frac{k_0+k_1}{2} \rfloor + 1$ tokens. If $n_0 \le n - n_1$, then every output of the ladder will receive at least $\lfloor \frac{k_0+k_1+1}{2} \rfloor$ and at most $\lceil \frac{k_0+k_1+2}{2} \rceil = \lfloor \frac{k_0+k_1+1}{2} \rfloor + 1$ tokens. In either case, the output is smoothed. ∎

We now prove that the $O(\lg n)$-depth network given by the following four-phase construction (see also Figure 6) is in fact a $2n$-input 2-counter network.

Phase 1: Apply a $2n$-input sorting network to the $2n$ input wires.

Phase 2: Apply a n-input sorting network to the top n wires and another n-input sorting network to the bottom n wires.

Phase 3: Apply a ladder to all $2n$ wires.

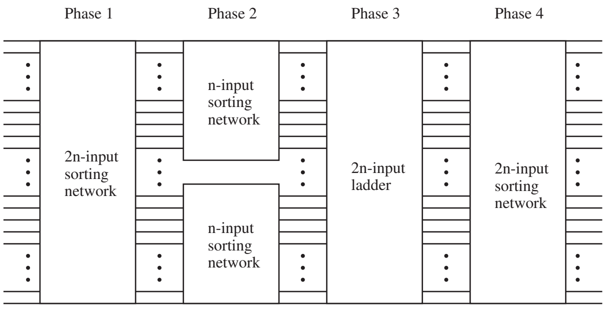

Figure 6: A $2n$-input 2-counter.

Phase 4: Apply a sorting network to all $2n$ wires.

Lemma 4.4 A network that counts any input sequence with no more than k tokens per wire is a k-counter.

Proof: Suppose that a given n-input network $\mathcal{N}$ counts every input sequence with no more than k tokens per wire, and let a k-smoothed input sequence be applied to $\mathcal{N}$. There exists some integer a such that $a \le x_i \le a + k$, for $0 \le i < n$. Using Lemma 3.1, we begin by passing all but a tokens from each input wire through the network. By our assumption, a counted shape will be produced at the output. Next, we pass the remaining a tokens from each input wire through the network. By a simple induction on the depth of network $\mathcal{N}$, we find that each output wire will receive an additional a tokens as a result of this pass. Hence, the final shape will be counted. ∎

Given the preceding lemma, it is sufficient to prove that our network counts when each x_i is drawn from $\{0, 1, 2\}$, for $0 \le i < n$. Fixing a particular input sequence of this type, let n_0, n_1, and n_2 denote the number of wires receiving 0, 1, and 2 tokens, respectively.

Lemma 4.5 After applying the first phase of the construction (i.e., a $2n$-input sorting network), all wires containing 2 tokens are located in the top n_2 wires. Similarly, all wires containing 0 tokens are located in the bottom n_0 wires.

Proof: We will compare the output of a sorting network (made up of comparator gates) to that of an isomorphic balancing network under a fixed input sequence consisting of 0's, 1's, and 2's. Note that a balancer receiving only 0, 1, or 2 tokens on each input wire mimics the behavior of a comparator unless one of the inputs is 0 and the other input is 2. In such a case, the balancer outputs 1 along each wire, while the comparator outputs 2 along the top wire and 0 along the bottom wire. For the purposes of analysis, whenever such an event occurs, we label the 1 output along the top wire of the balancer 1_2 and the 1 output along the bottom wire 1_0. In addition, we "enforce" the hierarchy $0 < 1_0 < 1 < 1_2 < 2$ in the balancing network (i.e., if 1_0 and 1_2 enter a balancer then 1_2 is output along the top wire and 1_0 is output along the bottom wire). An easy induction (on depth) proves that whenever a 2 (resp., 0) is output by a wire of the comparator-based sorting network, a 2 or 1_2 (resp., 0 or 1_0) is output on the corresponding wire of the balancer network. This proves the claim. ∎

We now complete the proof that the network described in this section is a 2-counter by considering two cases.

Case (i): $n_0 \leq n$ and $n_2 \leq n$.
As a result of Lemma 4.5, the inputs to each of the two n-input sorting networks in Phase 2 will be smoothed, so the outputs from each of these networks will have a counted shape (Lemma 3.2). The two counted shapes are then passed through a ladder, which produces a smoothed output (Lemma 4.3). The final sorting network then counts this smoothed shape (Lemma 3.2).

Case (ii): $n_0 > n$ or $n_2 > n$.
Without loss of generality it may be assumed that $n_2 = n + k$ for some $k > 0$. Arguing as in the proof of Lemma 4.5, the output of the first $2n$-input sorting network consists of $n_0 - m$ 0's, m 1_0's, n_1 1's, m 1_2's, and $n_2 - m$ 2's for some nonnegative integer m. Furthermore, the top n wires receive only 2's and 1_2's, so the output of the top n wires is smoothed and at least $n - m$ of these wires receives a 2. At the same time, at most $n - k - m < n - m$ of the bottom n wires receive a 0. Thus, after applying the two n-input sorting networks, the output of the top n wires will be counted and each of the top $n-m$ wires will receive a 2 (Lemma 3.2). Furthermore, every 0 must appear on one of the bottom $n-m$ wires (Lemma 4.5). Hence, when the ladder is applied, all existing 0's will be paired with 2's, leaving a smoothed output consisting of 1's and 2's. The final sorting network then produces a counted shape (Lemma 3.2). ∎

The preceding discussion has provided an explicit construction for logarithmic depth 2-counter networks having any *even* number of inputs. It is straightforward to extend this result to the case of an odd number of inputs, yielding the theorem stated below.

Theorem 4.1 There exists an explicitly constructible family of $O(\lg n)$-depth, n-input 2-counter networks. ∎

5 A random construction

We now present the first major construction leading to our main result. We make use of the networks defined in the previous section to construct a "random" network.

Definition 5.1 Given a set of balancing networks and an associated probability distribution, the random network obtained by sampling from the set is referred to as a *random balancing network*.

We will be particularly interested in random balancing networks that count any input sequence with high probability. Such a network will be referred to as a random counting network, and a lower bound on the associated probability of success (the minimum over all input sequences of the probability that the random network counts the sequence) will be explicitly stated.

In this section, we present an $O(d)$-depth family of 2^d-input random counting networks that count with probability at least $1 - 2^{-2^{\epsilon d}}$, where ϵ is any constant such that $0 < \epsilon < \frac{1}{2}$. Since the depth of the network is independent of ϵ, we will choose ϵ close to $\frac{1}{2}$. Throughout this section, fix a choice of $\epsilon < \frac{1}{2}$, let $\epsilon_0 = (\epsilon + \frac{1}{2})/2$, let d denote an arbitrary nonnegative integer, let $d' = \left\lceil \sqrt{d} \right\rceil$, and let $\mathcal{N}^*$ denote the 2^d-input random counting network to be constructed.

In order to define $\mathcal{N}^*$, we need to provide a set of 2^d-input balancing networks and an associated probability distribution. This set of networks will consist of $(2^d)!$ networks that are identical in every respect except for a permutation of the wires that is applied at one point in the construction. The probability distribution will be uniform over this set of networks. Thus, letting S_d denote the set of $(2^d)!$ permutations on 2^d objects, the networks of the set $\mathcal{N}^*$ are in one-to-one correspondence with the elements of S_d. In particular, for each permutation π in S_d, we will construct the balancing network $\mathcal{N}_\pi$ of $\mathcal{N}^*$ by applying the following procedure (see also Figure 7).

Phase 1: Apply a butterfly balancing network to all 2^d input wires.

Phase 2: Let A_i denote the set of $2^{d'}$ wires $X_{\pi(j)}$, $i2^{d'} \leq j < (i+1)2^{d'}$, $0 \leq i < 2^{d-d'}$. Apply a $2^{d'}$-input bitonic counting network [4] to each A_i.

Phase 3: Apply a butterfly balancing network to all 2^d wires.

Phase 4: Let B denote the set of ϵ_0-bad outputs of Phase 3. Apply a $(|B|, 2^d, \lg d)$-pairing network, mapping the wires of B to the bad inputs of the pairing network.

Phase 5: Apply a butterfly balancing network to all 2^d wires.

Phase 6: For the remainder of the construction, we refer to B (resp., G) as the set of ϵ_0-bad (resp., ϵ_0-good) outputs of Phase 5. Apply a 2-counter to G.

Phase 7: Partition the outputs of the 2-counter of Phase 6 into two equal-sized sets G_0 and G_1, placing the top $|G|/2$ outputs in G_0 and the bottom $|G|/2$ outputs in G_1. Apply a $(|B|, |B| + |G_0|, 2)$-pairing network to $B \cup G_0$, mapping the wires of B to the bad inputs of the pairing network.

Phase 8: Let B' denote the set of $4|B|$ bad outputs of the pairing network of Phase 7. Apply a $(|B'|, |B'| + |G_1|, 2)$-pairing network to $B' \cup G_1$, mapping the wires of B' to the bad inputs of the pairing network.

Phase 9: Apply a 2-counter to all 2^d wires.

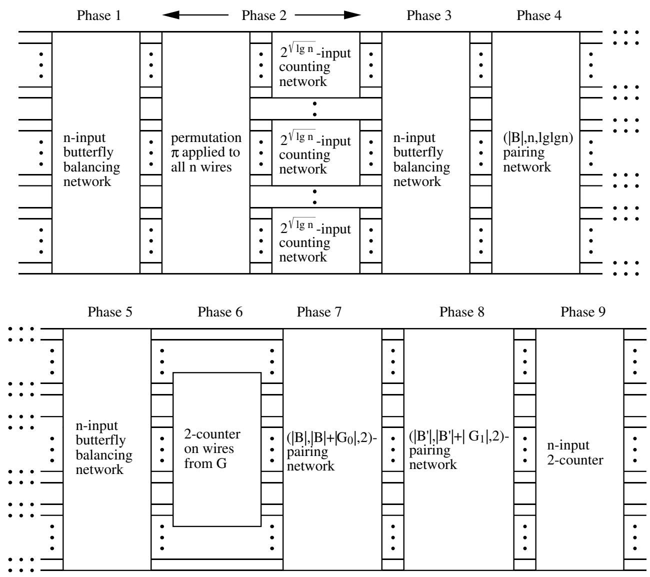

Figure 7: The balancing network $\mathcal{N}_\pi$.

Lemma 5.1 The output of Phase 1 is d-smoothed.

Proof: Immediate from Corollary 4.1.1. ∎

We are left to prove that the remainder of the network is a d-counter with high probability. By Lemma 4.4, we can assume that the number of tokens per wire is between 0 and d after Phase 1.

Definition 5.2 Let I denote an input sequence containing a total of $\mu 2^d$ tokens for some real value μ, let $\mu' = \lfloor \mu + \frac{1}{2} \rfloor$, and focus on the 2^d outputs of Phase 2 when input sequence I is applied to network $\mathcal{N}_\pi$. Let a (resp., b) denote the number of these wires receiving strictly more (resp., fewer) than $\mu' + 1$ (resp., $\mu' - 1$) tokens. The pair $(I, \mathcal{N}_\pi)$ is defined to be ϵ_0-**nice** if and only if $\max\{a, b\} \le 2^{\epsilon_0 d}$.

In the next portion of our proof we obtain a Chernoff-like bound for sampling *without* replacement. We use martingales for this purpose.

Definition 5.3 A martingale is a sequence of random variables $X_0, \ldots, X_m$ such that

$$E[X_{i+1} \mid X_i] = X_i,$$

$0 \le i < m$.

We will make use of the following variant of Azuma's Inequality [3] in the analysis of Phase 2:

Theorem 5.1 Let $X_0 = c$ and let $X_0, \ldots, X_m$ be a martingale with

$$|X_{i+1} - X_i| \le U,$$

$0 \le i < m$. Then

$$\Pr(X_m - c > \lambda U \sqrt{m}) < e^{-\lambda^2/2}$$

for all $\lambda > 0$. ∎

The following lemma represents a straightforward application of Azuma's Inequality.

Lemma 5.2 Let Q be an arbitrary set of N numbers with mean μ drawn from the real interval $[0, U]$. Let the random variable S denote the sum of m elements $\{q_1, \ldots, q_m\}$ chosen uniformly without replacement from Q. Then for all $\lambda > 0$,

$$\Pr(S > \lambda U \sqrt{m} + m\mu) < e^{-\lambda^2/2}.$$

Proof: Let $X_0 = E[S]$ and for $1 \le i \le m$ let

$$X_i = E[S \mid q_1, \ldots, q_i].$$

The X_i's form a martingale. Furthermore,

$$\begin{aligned}
|X_{i+1} - X_i| & \\
&= |E[X \mid q_1, \ldots, q_{i+1}] - E[X \mid q_1, \ldots, q_i]| \\
&= \left| \frac{N-m}{N-i-1} \left(q_{i+1} - \frac{N\mu - \sum_{1 \le k \le i} q_k}{N-i} \right) \right| \\
&\le U
\end{aligned}$$

since $0 \le (N-m)/(N-i-1) \le 1$, $0 \le q_{i+1} \le U$, and $0 \le (N\mu - \sum_{1 \le k \le i} q_k)/(N-i) \le U$. In addition, $X_0 = m\mu$ and $S = X_m$. By Theorem 5.1,

$$\Pr(S - m\mu > \lambda U \sqrt{m}) < e^{-\lambda^2/2} \qquad (1)$$

for all $\lambda > 0$. ∎

The following lemma represents the crux of the random construction. It shows that a great deal of "global" smoothing can be accomplished by smoothing small sets of wires in parallel.

Lemma 5.3 Let I be any input sequence to $\mathcal{N}_\pi$ where π is chosen uniformly at random from S_d. Then with probability at least $1-2^{-2^{\epsilon d}}$, the pair $(I, \mathcal{N}_\pi)$ is ϵ_0-nice.

Proof: Suppose that input sequence I contains $\mu 2^d$ tokens and let $\mu' = \lfloor \mu + \frac{1}{2} \rfloor$. The set A_i (refer to Phase 2) is defined to be *high-bad* (resp., *low-bad*) if the total number of tokens received by A_i exceeds $2^{d'}(\mu'+1)$ (resp., is less than $2^{d'}(\mu'-1)$), for $0 \leq i < 2^{d-d'}$. We will show that with probability at least $1-2^{-2^{\epsilon d}}$, at most $2^{\epsilon d}$ of the A_i's are high-bad. A symmetric result holds for the number of low-bad A_i's. Thus, at the end of Phase 2, $2^{\epsilon d} 2^{\lceil d' \rceil} = o(n^{\epsilon_0})$ wires will contain more (resp., fewer) than $\mu'+1$ (resp., $\mu'-1$) tokens, proving the lemma. Let X denote the event that a particular set of $r \stackrel{\text{def}}{=} \lfloor 2^{\epsilon d} \rfloor$ of the A_i's are all high-bad. We have

$$\begin{aligned} \Pr(\text{at least } r \text{ of the } A_i\text{'s are high-bad}) & \\ \leq & \binom{2^{d-d'}}{r} \Pr(X) \\ \leq & \; 2^{dr} \Pr(X). \end{aligned}$$

Next we show that $\Pr(X) = 2^{-\omega(d)r}$, yielding the result. Let S be the total number of tokens on a particular set of $r2^{d'}$ wires. Then $\Pr(S \geq r2^{d'}(\mu'+1)) \geq \Pr(X)$. S is the sum of values sampled without replacement, so Lemma 5.2 can be applied. In this case, $\lambda = \frac{\sqrt{m}}{2U}$, $U = d$, $m = r2^{d'}$, and $N = 2^d$. So the probability bound in Lemma 5.2 becomes

$$\begin{aligned} \Pr(S \geq r2^{d'}(\mu'+1)) & \leq \Pr(S \geq r2^{d'}(\mu + \frac{1}{2})) \\ & < e^{-\frac{m}{8U^2})} \\ & = e^{-\frac{r2^{d'}}{8d^2}} \\ & = 2^{-\omega(d)r}, \end{aligned}$$

as desired. ■

For the remainder of this section, we assume that I is a particular input sequence with $\mu 2^d$ tokens and that $\mathcal{N}_\pi$ is a particular network in $\mathcal{N}^*$ such that the pair $(I, \mathcal{N}_\pi)$ is ϵ_0-nice. We also set $\mu' = \lfloor \mu + \frac{1}{2} \rfloor$.

Lemma 5.4 If I is input to $\mathcal{N}_\pi$ then at any level after Phase 3, at most $d2^{\epsilon d + d' + 1}$ wires will receive more than $\mu'+1$ (resp., fewer than $\mu'-1$) tokens.

Proof: Before Phase 3, we know that at most $2^{\epsilon d + d' + 1}$ wires received more than $\mu'+1$ tokens. Since each of these wires receives at most d tokens, they can contribute tokens to at most $d2^{\epsilon d + d' + 1}$ wires at any given level of the network. Thus, at any level after Phase 3 there will be at most $d2^{\epsilon d + d' + 1}$ wires with more than $\mu'+1$ tokens. An analogous argument yields the upper bound for the number of wires with fewer than $\mu'-1$ tokens. ■

Lemma 5.5 If I is input to $\mathcal{N}_\pi$ then every ϵ_0-good output of Phase 3 will receive $\mu'-1$, μ', or $\mu'+1$ tokens.

Proof: By Lemma 5.4, at most $d2^{\epsilon d + d' + 1}$ outputs of Phase 3 contain more than $\mu'+1$ tokens. Now consider the hypercube-like structure of the wires after Phase 3 (Lemma 4.1). A wire X_i where $\text{bin}(i, d)$ has at least $\epsilon_0 d$ 1's cannot receive more than $\mu'+1$ tokens. Suppose one such wire did. Then all wires corresponding to the subcube with dimension at least $\epsilon_0 d$ defined by fixing all the 0 bits in $\text{bin}(i, d)$ and allowing the others to vary would have more than $\mu'+1$ tokens. But there are more than $d2^{\epsilon d + d' + 1}$ such wires, a contradiction. The argument above can be repeated to show that a wire X_i where $\text{bin}(i, d)$ has at least $\epsilon_0 d$ 0's cannot have fewer than $\mu'-1$ tokens. This proves the lemma. ■

We must ensure that enough good wires are input to the pairing networks in $\mathcal{N}_\pi$ so that all of the bad wires can be matched. Since no pairing network in $\mathcal{N}_\pi$ has depth greater than $\lg d$, the following lemma provides the necessary lower bound on the number of good wires by providing an upper bound on the number of bad wires:

Lemma 5.6 The number of ϵ_0-bad wires in $\mathcal{N}_\pi$ is $o(\frac{2^d}{d})$.

Proof: We have

$$\begin{aligned} |\epsilon_0\text{-bad wires}| & = 2 \sum_{0 \leq i < \epsilon_0 d} \binom{d}{i} \\ & = 2^{d(H(\epsilon_0) + o(1))} \\ & = O(2^{cd}), \end{aligned}$$

where H denotes the entropy function, and c is a constant with $c < 1$ since $\epsilon_0 < \frac{1}{2}$. ■

Lemma 5.7 If I is input to $\mathcal{N}_\pi$ then every output wire of Phase 4 will receive between $\mu'-2$ and $\mu'+2$ tokens. Furthermore, $o(2^{\epsilon_0 d})$ of these outputs will receive exactly $\mu'-2$ (resp., $\mu'+2$) tokens.

Proof: The claim that all wires will contain between $\mu'-2$ and $\mu'+2$ tokens follows immediately from Lemma 4.2 with $a = \mu'-1$, $b = \mu'+1$, and $k = \lg d$. By Lemma 5.4, the number of outputs of Phase 4 that receive exactly $\mu'-2$ (resp., $\mu'+2$) tokens is at most $d2^{\epsilon d + d' + 1} = o(2^{\epsilon_0 d})$. ■

Lemma 5.8 If I is input to $\mathcal{N}_\pi$ then after Phase 5, each ϵ_0-good wire will contain $\mu'-1$, μ', or $\mu'+1$ tokens and each ϵ_0-bad wire will have between $\mu'-2$ and $\mu'+2$ tokens.

Proof: Before Phase 5, the maximum number of tokens per wire is $\mu'+2$ while the minimum number is $\mu'-2$. As a result, for the remainder of the network, the number of wires containing $\mu' + 2$ (similarly $\mu' - 2$) tokens cannot increase. The butterfly in Phase 5 restructures the wires in the form of a hypercube (Lemma 4.1). After this phase, there will be $o(2^{\epsilon_0 d})$ wires with either $\mu' - 2$ or $\mu' + 2$ tokens. Arguing as in the proof of Lemma 5.5, each ϵ_0-good wire will receive $\mu' - 1$, μ', or $\mu' + 1$ tokens. ■

Lemma 5.9 If I is input to $\mathcal{N}_\pi$ then after Phase 6, the good wires will have a counted shape in which each wire contains either $\mu' - 1$ or μ' tokens, or contains μ' or $\mu' + 1$ tokens.

Proof: Immediate from Lemma 5.8 and the definition of a 2-counter. ■

Lemma 5.10 If I is input to $\mathcal{N}_\pi$ then the output of Phase 8 is 2-smoothed.

Proof: Since the good wires have a counted shape, either G_0 or G_1 is homogeneous (i.e., all wires in the set receive the same number of tokens). Assume without loss of generality that the good wires from Phase 6 contain either μ' or $\mu' + 1$ tokens. We consider two cases:

G_0 **is homogeneous:** If each wire in G_0 receives $\mu' + 1$ tokens, then by Lemma 4.2, every output of Phase 7 will receive between μ' and $\mu' + 2$. If each wire in G_0 receives μ' tokens, then by the same reasoning every output of Phase 7 will receive between $\mu' - 1$ and $\mu' + 1$ tokens.

G_1 **is homogeneous:** After Phase 7, only the wires in B' can receive either $\mu' - 2$ or $\mu' + 2$ tokens. Arguing as in the case where G_0 is homogeneous, the output of Phase 8 will be 2-smoothed.

■

Lemma 5.11 If I is input to $\mathcal{N}_\pi$ then the output of Phase 9 will have a counted shape. ■

Theorem 5.2 For any input sequence I, the $O(d)$-depth random counting network $\mathcal{N}^*$ will count I with probability at least $1 - 2^{-2^{\epsilon d}}$. ■

6 An optimal existence result

In this section, we prove our main result, namely, we establish the existence of $O(\lg n)$-depth counting networks. Our networks are non-uniform in that we do not know of any polynomial-time procedure for generating the network with n inputs. However, our results can easily be extended to provide a randomized algorithm that, given n, produces an $O(\lg n)$-depth counting network in polynomial time with extremely high probability.

Our deterministic network is constructed recursively using the random counting network of Section 5 as a building block. Our approach is to recursively construct deterministic counting networks over small sets of input wires and then to "merge" the outputs of these networks via a larger random counting network. In order to achieve a small depth of recursion, it is desirable to partition the n inputs into a large number of small sets. The granularity of the recursive partition will be determined in such a way that the total number of possible inputs to the larger network will be small relative to the probability with which the network succeeds (i.e., small relative to the reciprocal of the probability of failure). As a result, we will be able to argue that some fixed choice for the larger network will be guaranteed to produce a counted output.

Lemma 6.1 If the number of tokens input to each wire of a 2^d-input counting network is no more than l, then the number of possible output shapes is at most $l2^d + 1$. ■

Lemma 6.2 Consider a 2^d-input balancing network $\mathcal{N}$ made up of $2^{d-\lfloor\delta d\rfloor}$ disjoint $2^{\lfloor\delta d\rfloor}$-input counting networks, $0 \le \delta \le 1$. If the number of tokens received by each input wire of $\mathcal{N}$ is no more than d, then the number of possible output shapes is at most $(d2^d + 1)^{2^{d-\lfloor\delta d\rfloor}} < 2^{d2^{d-\lfloor\delta d\rfloor}(1+o(1))}$.

Proof: By Lemma 6.1, each $2^{\lfloor\delta d\rfloor}$-input network can produce at most $d2^{\lfloor\delta d\rfloor} + 1 \le d2^d + 1$ possible output shapes. Since $\mathcal{N}$ contains $2^{d-\lfloor\delta d\rfloor}$ such networks, the result follows. ■

Lemma 6.3 Let $\frac{1}{2} < \delta < 1$, and let S denote any fixed set of at most $2^{d2^{d-\lfloor\delta d\rfloor}(1+o(1))}$ possible input sequences of length 2^d. Then there exists a 2^d-input balancing network of depth $O(d)$ that counts every shape in the set S.

Proof: In Section 5, we constructed a set $\mathcal{N}^*$ of 2^d-input balancing networks with the property that any fixed input sequence is counted by at least a $1 - 2^{-2^{\epsilon d}}$ fraction of the networks in $\mathcal{N}^*$, where ϵ is any constant such that $0 < \epsilon < \frac{1}{2}$. If $2^{d2^{d-\lfloor\delta d\rfloor}(1+o(1))}2^{-2^{\epsilon d}} < 1$, then at least one of the networks in $\mathcal{N}^*$ must count every input sequence in the set S. The desired inequality is satisfied for $\delta > 1 - \epsilon$ (e.g., $\delta = 1 - \epsilon/2$), and d sufficiently large. ■

Theorem 6.1 There exists a 2^d-input counting network of depth $O(d)$.

Proof: Consider the following recursive construction. First, apply a 2^d-input butterfly balancing network. This yields a d-smoothed shape, and we can assume without loss of generality that each output wire receives a number of tokens between 0 and d, inclusive. Second, partition the butterfly outputs into $2^{d-\lfloor\delta d\rfloor}$ sets of size $2^{\lfloor\delta d\rfloor}$, $\frac{1}{2} < \delta < 1$, and count each of these sets recursively. Finally, feed the outputs of these networks into a 2^d-input network $\mathcal{N}$ of depth $O(d)$ that counts every input sequence that it could possibly receive from the smaller counting networks. The existence of network $\mathcal{N}$ is guaranteed by Lemmas 6.2 and 6.3.

Let $D(d)$ denote the depth of such a 2^d-input network. Then

$$\begin{aligned} D(d) &= D(\lfloor \delta d \rfloor) + O(d) \\ &= O(d). \end{aligned}$$

■

7 An explicit construction

In this section, we present a nearly logarithmic depth family of counting networks based on a recursive construction. This is a polynomial-time uniform family of networks: the running time of the (deterministic) construction procedure is polynomial in the size of the network being generated.

Definition 7.1 For any pair of nonnegative integers a and b, and any real value $\Delta \geq 1$, an (a, b, Δ)-clusterer is a 2^{a+b}-input balancing network whose outputs are partitioned into 2^a groups of size 2^b such that (for any input sequence) the total number of tokens output in each group differs by at most Δ.

Definition 7.2 For any pair of nonnegative integers a and b, and any real value $\Delta \geq 1$, let $B(a, b, \Delta)$ denote the optimal depth of an (a, b, Δ)-clusterer.

Definition 7.3 For any nonnegative integer d, let $C(d)$ denote the optimal depth of a 2^d-input counting network.

Lemma 7.1 For any pair of nonnegative integers a and b, and any real value $\Delta \geq 1$,

$$B(a, b, \Delta) \leq C(a + b).$$

Proof: An (a, b, Δ)-clusterer can be constructed from a 2^{a+b}-input counting network by simply partitioning the outputs of the counting network in an appropriate manner. For instance, we could form the ith group of size 2^b by selecting all output wires congruent to i modulo 2^a, $0 \leq i < 2^a$. ■

Lemma 7.2 For any pair of nonnegative integers a and b, and any real value δ such that $\delta 2^{\lceil b/2 \rceil} \geq 3$, we have

$$B(a, b, \delta 2^b) \leq 2B(\lceil a/2 \rceil, \lceil b/2 \rceil, \delta 2^{\lceil b/2 \rceil}/3).$$

Proof: In the following argument, the term "(x, y)-matrix" (where x and y denote nonnegative integers) will be used to refer to a matrix with 2^x rows and 2^y columns. Let $\Delta = \delta 2^{\lceil b/2 \rceil}/3$.

Consider the following approach for constructing an $(a, b, \delta 2^b)$-clusterer. First, view the 2^{a+b} inputs as being arranged in a $(\lfloor a/2 \rfloor + \lfloor b/2 \rfloor, \lceil a/2 \rceil + \lceil b/2 \rceil)$-matrix. View each row of this matrix as consisting of $2^{\lceil a/2 \rceil}$ contiguous *row blocks* of size $2^{\lceil b/2 \rceil}$. Apply a $(\lceil a/2 \rceil, \lceil b/2 \rceil, \Delta)$-clusterer to each of the rows in parallel, permuting the output of the clusterer in such a way that each output group corresponds to a single row block.

Partition the entire matrix into $2^{\lfloor b/2 \rfloor}$ *horizontal blocks*, where the ith horizontal block consists of rows $i2^{\lfloor a/2 \rfloor}$ through $(i+1)2^{\lfloor a/2 \rfloor} - 1$, $0 \leq i < 2^{\lfloor b/2 \rfloor}$. Partition each horizontal block into $2^{\lceil a/2 \rceil}$ *submatrix blocks*, where the ith submatrix block consists of columns $i2^{\lceil b/2 \rceil}$ through $(i+1)2^{\lceil b/2 \rceil} - 1$ of the horizontal block, $0 \leq i < 2^{\lceil a/2 \rceil}$.

Note that within a given horizontal block, the total number of tokens in each submatrix block differs by at most $2^{\lfloor a/2 \rfloor}\Delta$. Now apply a $(\lfloor a/2 \rfloor, \lceil b/2 \rceil, \Delta)$-clusterer to each submatrix block, permuting the output of the clusterer in such a way that each output group corresponds to a single row block. Fix an input sequence and assume that the minimum number of tokens routed to any row block of a given horizontal block H is m. Let this minimum be achieved at row block R within submatrix block S. No row block in S contains more than $m + \Delta$ tokens. Thus the total number of tokens in submatrix S is at most $(m + \Delta)2^{\lfloor a/2 \rfloor}$. Hence, every submatrix block within H must contain at most $(m+2\Delta)2^{\lfloor a/2 \rfloor}$ tokens. This implies that the average number of tokens per row block in any submatrix block of H is at most $m+2\Delta$. Within a given submatrix block, no row block can contain a number of tokens that deviates from the average by more than Δ. Hence every row block of every submatrix block within H contains between m and $m + 3\Delta$ tokens.

We now form the 2^a output groups of size 2^b as follows. Number the row blocks within each horizontal block from 0 to 2^a in an arbitrary order. The ith output group may then be defined as the union of the ith row block from each of the $2^{\lfloor b/2 \rfloor}$ horizontal blocks, $0 \leq i < 2^a$. It is straightforward to verify that the total number of tokens within each output group differs by at most $3\Delta 2^{\lfloor b/2 \rfloor} = \delta 2^b$, as required.

The preceding argument has actually established that

$$B(a, b, \delta 2^b) \leq B(\lceil a/2 \rceil, \lceil b/2 \rceil, \Delta) + B(\lfloor a/2 \rfloor, \lceil b/2 \rceil, \Delta).$$

In order to obtain the result claimed by the lemma, we can replace the set of $(\lfloor a/2 \rfloor, \lceil b/2 \rceil, \Delta)$-clusterers employed in parallel above by a set of half as many $(\lceil a/2 \rceil, \lceil b/2 \rceil, \Delta)$-clusterers applied to adjacent pairs of submatrix blocks whenever a is odd. [Note that whenever a is odd the number of submatrix blocks per horizontal block is even.] ■

Corollary 7.2.1 For any nonnegative integers a, b, and i, and any real value δ such that $\delta 2^{\lceil b/2^i \rceil} \geq 3^i$, we have

$$B(a, b, \delta 2^b) \leq 2^i B(\lceil a/2^i \rceil, \lceil b/2^i \rceil, \delta 2^{\lceil b/2^i \rceil}/3^i).$$

Proof: Apply the preceding lemma i times. ∎

Setting $a = \lfloor d/2 \rfloor$, $b = \lceil d/2 \rceil$, $\delta = 1$, and $i = \lfloor \lg a - \lg\lg a - 1 \rfloor$, we easily find that

$$\begin{aligned} 2^i &\leq \frac{a}{2\lg a}, \\ \lceil a/2^i \rceil &= \Theta(\lg d), \\ \lceil b/2^i \rceil &= \Theta(\lg d), \text{ and} \\ \delta 2^{\lceil b/2^i \rceil} &\geq 3^i. \end{aligned}$$

Applying Lemma 7.1 then yields

$$B(\lfloor d/2 \rfloor, \lceil d/2 \rceil, 2^{\lceil d/2 \rceil}) = O(C(O(\lg d))\, d/\lg d). \tag{2}$$

The following lemma indicates the close relationship between the depth complexity of certain clustering networks and that of counting networks.

Lemma 7.3 For any nonnegative integer d,

$$C(d) \leq B(\lfloor d/2 \rfloor, \lceil d/2 \rceil, 2^{\lceil d/2 \rceil}) + C(\lfloor d/2 \rfloor) + O(d).$$

Proof: Consider the following approach towards constructing a 2^d-input counting network (see Figure 8). First, apply a $(\lfloor d/2 \rfloor, \lceil d/2 \rceil, 2^{\lceil d/2 \rceil})$-clusterer to the 2^d inputs. Second, apply a $2^{\lfloor d/2 \rfloor}$-input counting network to each of the $2^{\lfloor d/2 \rfloor}$ resulting output groups. Fix a particular input sequence and let m denote the least number of tokens routed to any particular output Y. Every output in the same group as Y contains either m or $m+1$ tokens. Thus the total number of tokens in the group containing output Y is at most $(m+1)2^{\lceil d/2 \rceil}$. This implies that no output group contains more than $(m+2)2^{\lceil d/2 \rceil}$ tokens, and hence no single output wire receives more than $m+2$ tokens. It follows that the output of the network is 2-smoothed, and that we can complete our construction by applying the $O(d)$-depth 2-counter of Section 4.3. ∎

Theorem 7.1 We have

$$C(d) = O(c^{\lg^* d} d)$$

for some positive constant c.

Proof: Substituting Equation (2) into the inequality of Lemma 7.3, and making use of the fact that $C(d) = O(1)$ for $d = O(1)$, we obtain

$$C(d) \leq c_0 C(\lfloor c_0 \lg d \rfloor)\, d/\lg d + C(\lfloor d/2 \rfloor) + c_0 d.$$

for some positive constant c_0. Let $a = 2^{1/c_0}$, and let $\log_a^*(d)$ denote the base-a analogue of the $\lg^*$ function. It is sufficient to prove that

$$C(d) \leq c_1^{\log_a^* d} d$$

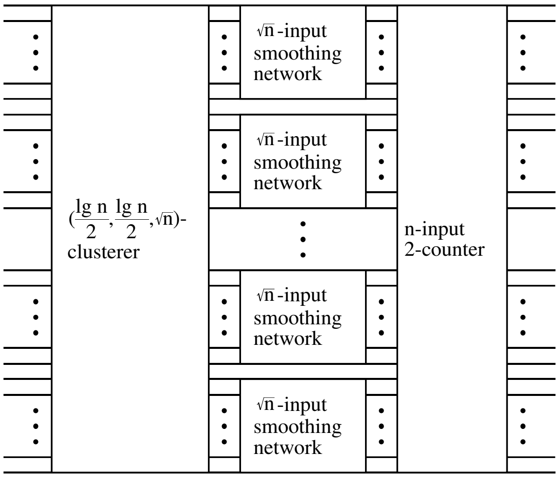

Figure 8: An n-input counting network where $n = 2^d$.

for some positive constant c_1. We prove this claim by induction on d. For the base case, namely $d \leq d_0$ where d_0 denotes an arbitrary positive constant, note that $C(d) = O(1)$. Now assume that the claim holds for all d strictly less than k, $k > d_0$, and consider the case $d = k$. Note that $\log_a^* k = 1 + \log_a^*(c_0 \lg k)$. Thus,

$$\begin{aligned} C(k) &\leq c_0 C(\lfloor c_0 \lg k \rfloor)\, k/\lg k + C(\lfloor k/2 \rfloor) + c_0 k \\ &\leq c_0^2 c_1^{\log_a^* k - 1} k + C(\lfloor k/2 \rfloor) + c_0 k \\ &\leq \left(\frac{c_0^2}{c_1} + \frac{1}{2}\right) c_1^{\log_a^* k} k + c_0 k \end{aligned}$$

by two applications of the induction hypothesis (assuming a sufficiently large choice of the constant d_0). For $c_1 \geq 4c_0^2$ and d_0 sufficiently large, the preceding inequality implies that

$$C(k) \leq c_1^{\log_a^* k} k,$$

as desired. ∎

8 Conclusions

This paper has made substantial progress towards improving our understanding of the depth complexity of counting networks. In Section 7, we presented an explicit construction of a family of counting networks with very close to logarithmic depth (nearly matching the trivial lower bound). In Section 6, we established the existence of optimal $O(\lg n)$-depth counting networks. Unfortunately, we do not know of any polynomial-time procedure for generating this optimal-depth family of networks. However, we *can* give a randomized polynomial-time generation procedure that works (i.e., produces an $O(\lg n)$-depth counting network) with extremely high probability. If we could efficiently test a circuit to determine whether or not it is a counting network then the value of the randomized generation procedure would be significantly enhanced. Unfortunately, we don't even know how to solve this kind of verification problem in the special case of sorting networks.

The Big-Oh depth bound for our optimal counting network construction hides an unrealistic multiplicative constant, largely due to the fact that the AKS sorting network is used as a subroutine. Of course, we cannot hope to improve this constant without also improving the sorting constant, since every counting network corresponds to a sorting network with the same depth/topology (by replacing each balancer with a comparator). With regard to the construction of smoothing networks, however, the situation is not so clear. In view of the fact that every smoothing network produced thus far has incorporated a "sorting" network as a primary component, it would be interesting to either bound the depth complexity of sorting by a small constant times the depth complexity of smoothing, or to construct a small-depth smoothing network that makes no use of sorting networks.

9 Acknowledgements

The authors would like to thank Nabil Kahale, Mauricio Karchmer, Tom Leighton, Stephen Ponzio, and Nir Shavit.

References

[1] E. Aharonson and H. Attiya. Counting networks with arbitrary fan-out. In *Proceedings of the 3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 104–113, January 1992.

[2] M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. *Combinatorica*, 3:1–19, 1983.

[3] N. Alon and J. H. Spencer. *The Probabilistic Method.* Wiley-Interscience, New York, NY, 1992.

[4] J. Aspnes, M.P. Herlihy, and N. Shavit. Counting networks and multi-processor coordination. In *Proceedings of the 23rd Annual Symposium on Theory of Computing*, pages 348–358, May 1991.

[5] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Computer Conference,* vol. 32, pages 307–314, 1968.

[6] V. Chvátal. Personal communication.

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

[8] M. Dowd, Y. Perl, M. Saks, and L. Rudolph. The balanced sorting network. Technical Report DCS–TR–127, Department of Computer Science, Rutgers University, June 1983.

[9] C.S. Ellis and T.J. Olson. Algorithms for parallel memory allocation. *Journal of Parallel Programming*, 17:303–345, 1988.

[10] E. Freudenthal and A. Gottlieb. Process coordination with fetch-and-increment. In *Proceedings of the 4th International Conference on Architecture Support for Programming Languages and Operating Systems*, April 1991.

[11] 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*, 5:164–189, 1983.

[12] M.R. Klugerman. Lecture 17: Counting networks. In F.T. Leighton, C.E. Leiserson, and N. Kahale, editors, *Research Seminar Series 15: Advanced Parallel and VLSI Computation*, pages 153–161. MIT Press, 1991.

[13] D. E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, Reading, MA, 1973.

[14] F. T. Leighton and C. G. Plaxton. A (fairly) simple circuit that (usually) sorts. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 264–274, October 1990.

[15] C. G. Plaxton. A hypercubic sorting network with nearly logarithmic depth. May 1992.

[16] H.S. Stone. Database applications of the fetch-and-add instruction. *IEEE Transactions on Computers*, C–33:604–612, 1984.