An Experimental Analysis of Counting Networks*

Eric N. Klein, Costas Busch, David R. Musser
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street, Troy, NY 12180
USA
emtklein@yahoo.com, buschc@cs.rpi.edu, musser@cs.rpi.edu

Abstract

Counting networks are highly distributed data structures that provide low contention solutions to distributed coordination problems. Counting networks are structured with balancers interconnected in a networked fashion. We study experimentally (with simulations and real implementation) the performance of counting networks according to three criteria:

(i) The way the balancers are distributed among the processors in the system, where we show that in most cases the best performance is to assign a connecting chain of balancers to each system node;

(ii) The various sizes of counting networks, where we examine the tradeoffs between size and number of nodes in the system;

(iii) The different kinds of counting networks, where we compare the bitonic, periodic, and BM counting networks where we find that for high loads the BM performs better than all the other networks.

Key–Words: Distributed data structures, Counting networks, Load balancing, Bitonic, Periodic.

1 Introduction

1.1 Counting Networks

Counting networks were introduced by Aspnes, Herlihy and Shavit [2] as a class of highly concurrent data structures that implement distributed counters. Counting networks are suitable for asynchronous distributed environments where multiple concurrent processes or processors have to solve solve distributed coordination problems such as distributed counters, queues, stacks, load balancing, and barrier synchronization.

The basic constituent of a counting network is the *balancer*. A (p, q)-balancer has p input wires and q output wires. Figure 1 depicts each balancer as a vertical line and the respective input/output wires as horizonal lines. Let $0, \ldots, q-1$ be the output wires of the balancer. In Figure 1, the wires are drawn in order so that wire 0 is at the top, while wire $q-1$ is at the bottom; the input wires

*A preliminary version of this work appears in the Master's Thesis of Eric N. Klein, "A Generic Simulation of Counting Networks," Rensselaer Polytechnic Institute, July 2003 [11].

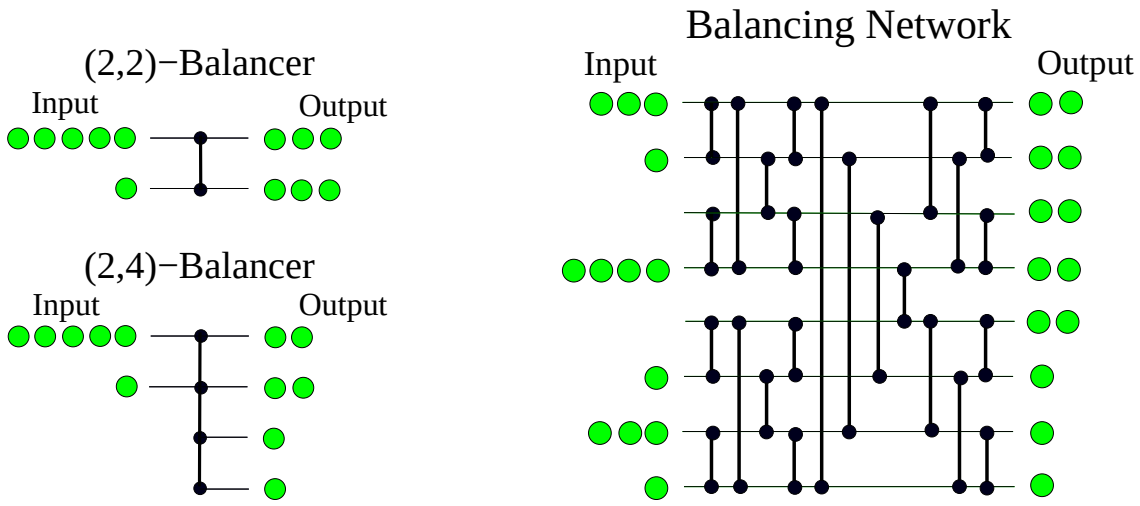

Figure 1: Two balancers and a balancing network (instance of the bitonic counting network of width 8)

are drawn similarly. The balancer acts as a toggle mechanism that receives *tokens* on its input wires and forwards them to the output wires. When the first token arrives on any input wire, it is forwarded on output wire 0, the second token exits on output wire 1, and this process repeats so that the kth token to be received by the balancer (on any input wire), is forwarded to output wire $(k-1) \bmod q$. In this way, the tokens are balanced on the output wires. In particular, if we denote by y_i the number of tokens that were exited on output wire i, then the balancer satisfies the *step property*: $0 \leq y_i - y_j \leq 1$ for any $0 \leq i < j \leq q-1$. We note that even if multiple tokens arrive simultaneously and asynchronously in the balancer, the balancer will handle the tokens sequentially in some arbitrary order.

A *balancing network* is formed by connecting together balancers in an acyclic fashion, where outputs of balancers are connected to inputs of other balancers (see the bottom of Figure 1). The set of unconnected wires on the left and right are the input wires and output wires, respectively, of the balancing network. The number of input and output wires is the *input* and *output* width, respectively of the balancing network. Similar to the balancer, the wires are drawn from top to bottom, so that if the output width is n, output wire 0 is at the top, while output wire $n-1$ is at the bottom. The balancing network can be partitioned into *levels* of balancers as follows. The balancers at the first level are those whose inputs are only connected to the input wires of the balancing network. Level l consists of those balancers whose inputs are connected to outputs of balancers at level $l-1$, and possibly to the outputs of lower levels and the inputs of the balancing network. The *depth* of the balancing network is the total number of levels that it contains. For example, the balancing network in Figure 1 has depth 6. The *size* of the balancing network is the number of its balancers.

A balancing network receives tokens on its input wires and then the tokens propagate through the network from balancer to balancer until they reach the output wires from which they exit the balancing network. The path that each token follows depends on the current state of each balancer. Many tokens may arrive asynchronously and traverse the balancing network simultaneously. Any

two tokens that access different balancers are processed simultaneously. Thus, balancing networks are distributed, enabling multiple tokens to enter, traverse, and exit the network concurrently.

Balancing networks are classified according to the distribution of the tokens on the output wires. A *counting network* is a balancing network whose output distribution satisfies the step property. In particular, consider a *quiescent state* in which no tokens remain in the counting network. Let y_i be the number of tokens that have exited on output wire i, then the counting network satisfies the step property: $0 \leq y_i - y_j \leq 1$ for any $0 \leq i < j \leq n-1$, where n is the output width of the network. For example, the balancing network in Figure 1 is a counting network. As we explain below, counting networks are suitable for solving the counting problem. There also exist other kinds of balancing networks. For example in *k-smoothing networks*, we require that the difference on the output wires does not exceed k, which is useful for load balancing in distributed systems. *Threshold networks* require balanced output on only one wire, which is adequate to implement parallel synchronization barriers.

In distributed/parallel programming environments often arise *counting problems* in which processors (processes or threads) need to increment shared program variables. For example, in the program each process may need to execute the line of code "v++", where v is a shared variable (suppose that v is initially set to 0). A simple solution to the counting problem is to use locks (mutual exclusion) that guarantee that only one process increments the shared variable at a time. However, this solution causes a sequential bottleneck to the program since all the processes have to serialize (in some arbitrary order) in the lock.

Counting networks are highly concurrent solutions to the counting problem. The variable v can be replaced by a counting network of appropriate output width n. Whenever a process wishes to increment v, it issues a token which traverses the network. The counter value is determined according to which output wire the token exits from the counting network. In particular, each output wire i is associated with some variable o_i. Initially $o_i = i$. If a token exits from wire i, it returns value o_i and then increments o_i by n. In this way, the tokens get values $0, 1, 2, 3, \ldots$, which is the solution to the counting problem.

Here, we study the following fundamental counting network constructions:

Bitonic counting network [2]:
this counting network has input/output width $n = 2^k$, $k \geq 1$, and is built from $(2,2)$-balancers. The bitonic network construction is recursive, where it is based on first building smaller counting networks whose outputs are combined with *merging* balancing networks. The depth is $(\lg^2 n + \lg n)/2 = O(\lg^2 n)$. At each level there are $n/2$ balancers. Thus, the size is $O(n \lg^2 n)$. The balancing network in Figure 1 is an instance of the bitonic counting network with $n = 8$.

Periodic counting network [2]:
this counting network has input/output width $n = 2^k$, $k \geq 1$, and consists of $(2,2)$-balancers. Its construction is based on a building *block* which has depth $\lg n$ and which is repeated $\lg n$ times. Thus, the total depth is $\lg^2 n$, which is about a factor of 2 deeper than the bitonic network. Each level of the periodic network has $n/2$ balancers, hence, the size is $O(n \lg^2 n)$. The benefit of the periodic network is that it is easier to be implemented in hardware, since someone needs to build only the basic block and repeat it a logarithmic amount of times. Figure 2 depicts the periodic network of width 8.

Busch-Mavronicolas (BM) counting network [6]:

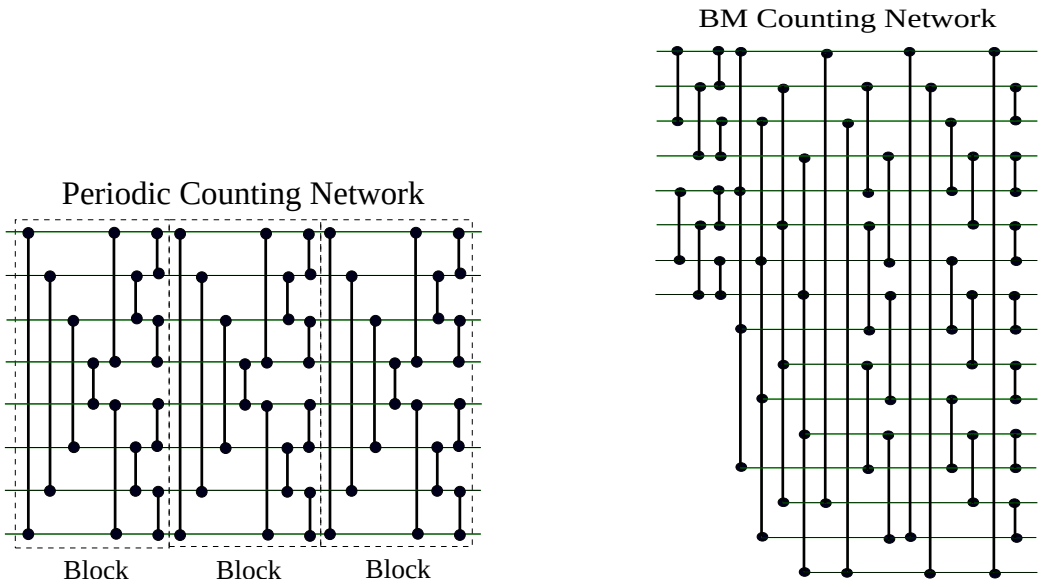

Figure 2: Left: The periodic counting network of width 8; Right: The BM counting network with input width 8 and output width 16

this counting network has input width $n = 2^k,\ k \geq 1$, and output width $t = pn$, for any $p \geq 1$. The network consists mainly of $(2, 2)$-balancers, and has a single level of $(2, 2p)$-balancers. The network construction is based on the idea of merging smaller counting networks, as in the bitonic network. The BM counting network has depth equal to the depth of the bitonic network with width n, namely, the depth is $(\lg^2 n + \lg n)/2 = O(\lg^2 n)$. For $n > 2$, the first $\log n$ levels consist of $n/2$ balancers each, while the remaining levels consist of $t/2$ balancers each; thus, the size is $O(t \lg^2 n)$. Figure 2 depicts the BM network with input width 8 and output width 16 (the depth of this network is 6, equal to the bitonic depth for width 8). The benefit of the BM network is that for the same input width n, we can chose larger output width $t \geq n$, while preserving the same depth. This implies higher concurrency capabilities.

1.2 Experiments

We give an experimental study on the performance of various kinds and sizes of counting networks. The performance is measured according to the average *traversal time* of the tokens in the counting network, which is the average time that a token spends in the counting network. Two parameters affect the traversal time: (i) the counting network depth, which represents the number of balancers that a token has to access, and (ii) the counting network *contention* which represents the token collisions in the balancers, that may cause delays since a balancer processes tokens sequentially. Ideally, a counting network should have small depth and small contention.

We perform simulations and execute a real implementation to measure the performance of counting networks. For the simulations we used the Component-Oriented Simulation Toolkit (COST) [7], which provides a generic framework for performing discrete event simulations. For the real implementation we used 10 UltraSparc-10 servers. We give evidence to show that the real implementation performance is similar to that seen in our simulation. In our experiments (simulation

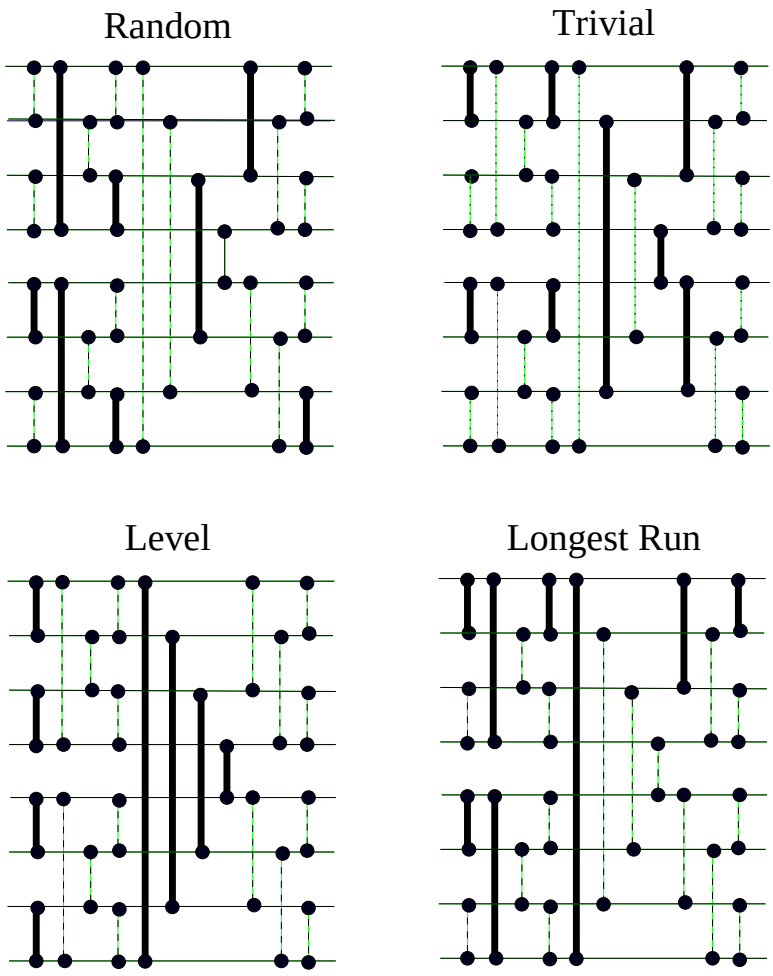

Figure 3: Assignments of balancers according to the balancer distribution methods; the bold balancers are assigned to one host (when the system has 3 hosts)

and real implementation) we consider message passing systems of multiple processors, which we refer to as *hosts*, and which communicate over an underlying communication network. In the simulation, we assume that the communication cost is the same among all pairs of hosts, and the hosts have the same computation speed.

When we simulate or implement counting networks, we map the balancers of a counting network to the hosts. Each host may hold multiple balancers, if the number of balancers is more than the hosts. On the other hand, if the hosts are more than the number of balancers, some hosts may not have any balancer. The hosts forward tokens to other hosts over the communication network. Tokens do not traverse the communication network when they are passed between balancers of the same host.

In our experiments, we study three factors that affect the performance of counting networks: (i) *distribution method*, which is the method that maps balancers to hosts; (ii) *counting network size*, which is the number of balancers of the network and it is directly related to the input and

output width, (iii) *counting network kind*, which in our case is either the bitonic, periodic, or BM counting networks. In particular, we examine the following distribution methods of balancers to hosts (see Figure 3):

Random distribution: Each balancer is assigned to a host at random and uniformly.

Trivial distribution: The balancers are assigned an arbitrary deterministic order when the counting network is created. The balancer with order i is assigned to host $i \bmod h$, where h is the number of hosts.

Level distribution: The balancers are assigned to hosts according to their level number, so that each host holds as many balancers of the same level as possible.

Longest run distribution: This method repeatedly finds the longest path through adjacent balancers that have not yet been assigned to any host and assigns these balancers to the same host.

The random and trivial distributions are the simplest distribution methods. The level distribution has the potential for a level-by-level pipelining of tokens by the hosts; however, it does not utilize the parallelism of balancers in the same level. The longest run distribution attempts to minimize the communication cost among hosts, since a token is likely to remain in the same host as it moves from one level to the next; at the same time it exploits the parallelism of the balancers in the same level, since same level balancers may reside in different hosts.

In the experiments, in most cases the best performance is exhibited by the longest run distribution, while the worst by the level distribution. We also determine the best size of the counting network given a particular token load. In terms of counting network kind, we show that the BM counting network outperforms the bitonic and periodic counting networks under heavy token load. This is a consequence of the structure of the BM counting network which has more inherent parallelism. However, under lighter token load the bitonic network is better due to the smaller number of balancers.

1.3 Related Work

Although the theoretical aspects of counting networks have been extensively studied, there is limited work on their experimental performance. Counting networks have been studied experimentally in [2] using a real implementation on an Encore Multimax multiprocessor using the Mul-T [12] parallel programming language, and also in [9] using a simulator for the 64-processor MIT Alewife distributed memory multiprocessor [1]. Those experiments show that counting networks outperform other distributed techniques for solving counting related problems. However, these papers don't compare various balancer distribution methods and they don't focus on the comparison of different counting network constructions. Here, we give a comprehensive experimental evaluation of counting networks.

A survey on counting networks can be found in [4]. Counting networks are isomorphic to sorting networks [13, Chapter 27]. Sorting networks are used for parallel sorting, which is a problem inherently synchronous, while counting networks are used for solving asynchronous coordination problems. It is known that every counting network is a sorting network, but not vice-versa [2]. The bitonic counting network is isomorphic to the bitonic sorting network due to Batcher [3], and the periodic counting network is isomorphic to the periodic sorting network [8]. An extension of the

bitonic counting network to arbitrary widths appears in [5]. The adaptability and self-stabilizing nature of counting networks in distributed environments has been studied in [10, 14].

Outline of Presentation

We proceed by describing the results of the simulations in Section 2), followed by the results of the real implementation in Section 3). We give our conclusions in Section 4

2 Simulations

Here we study the performance of the counting networks with simulations. In order to do so, we specify particular parameters that model the communication network and the computation of the hosts.

Consider a token which traverses a link in the communication network from some balancer in host h_1, to a balancer in a host h_2, where $h_1 \neq h_2$. In particular, the parameter *network_delay* expresses the communication cost imposed by the token traversal. The network is modelled so that while the token traverses the link between the two hosts, the hosts h_1 and h_2 are not allowed to use the communication network for a period of $network\_delay/2$; the factor of $1/2$ models the bidirectional links in the network, where two tokens can be sent over a link simultaneously. While the hosts are blocked, the requests for using the communication network by the remaining balancers in the same hosts are queued.

In a host, the tokens are processed sequentially. Consider a balancer b in a host h_i. Balancer b forwards each token after a calculated delay which is determined by three parameters: (i) a base access delay to the balancer *balancer_delay*, (ii) the number of balancers k on host h_i which are currently handling tokens, and (iii) a parallelism factor *parallelism* that tells how well each host can execute multiple threads for the balancers in parallel. In our experiments, we set the total balancer delay to $balancer\_delay \cdot (parallelism + k \cdot (1 - parallelism))$. When $parallelism = 0$, the delay is sequential while when $parallelism = 1$, the delay is purely parallel. We used $parallelism = 0.75$ to model modern computing machinery which is able to benefit by running processes or threads in parallel in multiprocessor machines.

A token is generated randomly in the system every *source_delay* time period. For example, 100 tokens are generated in a period of $100 \cdot source\_delay$ time steps. As the source delay increases, the token load decreases, and vice-versa. The source for each token is a randomly selected host. The first link followed by a token is toward the host of a random input balancer (level 1 balancer). Some hosts may act only as sources in the case where there are more hosts than balancers.

2.1 Simulation Results

2.1.1 Distribution Method

In the first experiment, we wish to determine the best distribution method of balancers to hosts. We use the bitonic network of width 16. The network consists of 80 balancers which are distributed over 20 hosts using the four distribution methods. The parameters of the simulation are set to $balancer\_delay = 10$, and $network\_delay = 100$. We vary the *source_delay* over a range of 1 to 100 in order to change the token load on the counting network. At most 1000 tokens are generated, and the experiment runs for at most 10^6 time units.

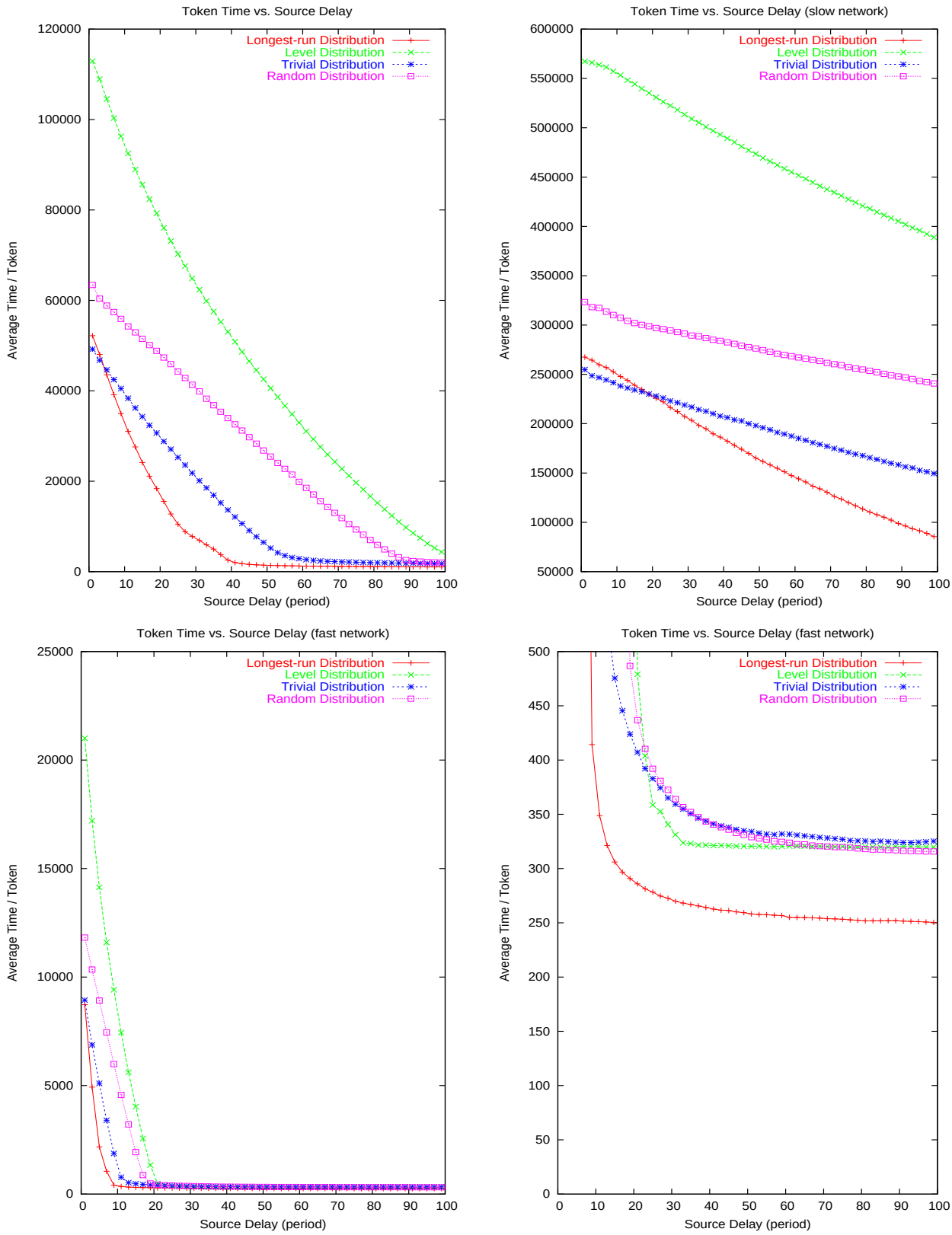

Figure 4: Comparing various balancer distribution algorithms in the bitonic network of width 16; Top left: original experiment; Top right: slow network; Bottom left: fast network; Bottom right: zoom in the fast network graphs

The results of this experiment are graphed in the top left part of Figure 4. The curves are monotonically decreasing. This corresponds to the fact that the performance of counting networks diminishes under increased contention. The point at which the slope levels off corresponds to the token load that the counting network can handle with a negligible effect due to contention.

We observe that the level distribution algorithm performs the worst while the trivial and longest-run perform significantly better. The longest-run is the best distribution for most of the token loads, while for very heavy token loads the trivial distribution is slightly better. We also performed two more experiments where we vary the *network_delay*. In one experiment we set $network\_delay = 20$, and the other $network\_delay = 500$, which is 5 times faster and slower, respectively, than the first experiment. The results are graphed in Figure 4, and are similar as before with the difference that everything is scaled by a factor of 5, including the points where the curves level off. This shows that *network_delay* and contention (due to *source_delay*), are both important factors for the performance of counting networks.

2.1.2 Network Size

Here we study the problem of determining the best network size given a number of hosts. Ideally, when we distribute a counting network over a communication network of hosts we need to have one balancer per host. However, this may be hard to achieve since the number of balancers in a counting network may not be equal to the number of hosts. If the network is too small, some of the available hosts may not contain any balancer, which under-utilizes the communication network, since some balancers and hosts will have to handle more load. If we use a larger counting network, then we may need to place multiple balancers on hosts, which may increase the contention on the hosts and degrade the performance of the counting network.

Our experiment compares the performance of different sizes of the bitonic counting network distributed over a range of 1 to 100 hosts. In particular, we use the bitonic network of width 2 to 32. The balancers are distributed among hosts using the longest-run distribution algorithm. The parameters are set to $balancer\_delay = 10$, $network\_delay = 100$, and $source\_delay = 10$. At most 1000 tokens are generated, and the experiment runs for at most 10^6 time units.

The results are shown in Figure 5. There is an overall downward slope in all of the performance curves until a certain point at which the curves level off completely to a constant value (this is the point where the number of hosts is getting more than the balancers). For up to 3 hosts the single balancer is a good choice. For up to 10 hosts, a width 4 network performs better. For up to 38 hosts, a width 8 network is good. Above that, the results show that either a 16 or 32 width network is a good choice.

2.1.3 Network Kind

Different counting network constructions perform differently under different conditions based on traits such as their depth, width, and internal structure. For example, the BM network can potentially handle higher contention better than the bitonic network since it has more balancers for most of its levels, and thus higher parallelism, with no change in the depth of the network. On the other hand, the periodic network is expected to be worse than the bitonic network due to the higher depth.

In this experiment, we compare the bitonic network with width 16, the periodic with width 16, and the BM network with 16 inputs and 32 outputs. The hosts are distributed using the longest-run

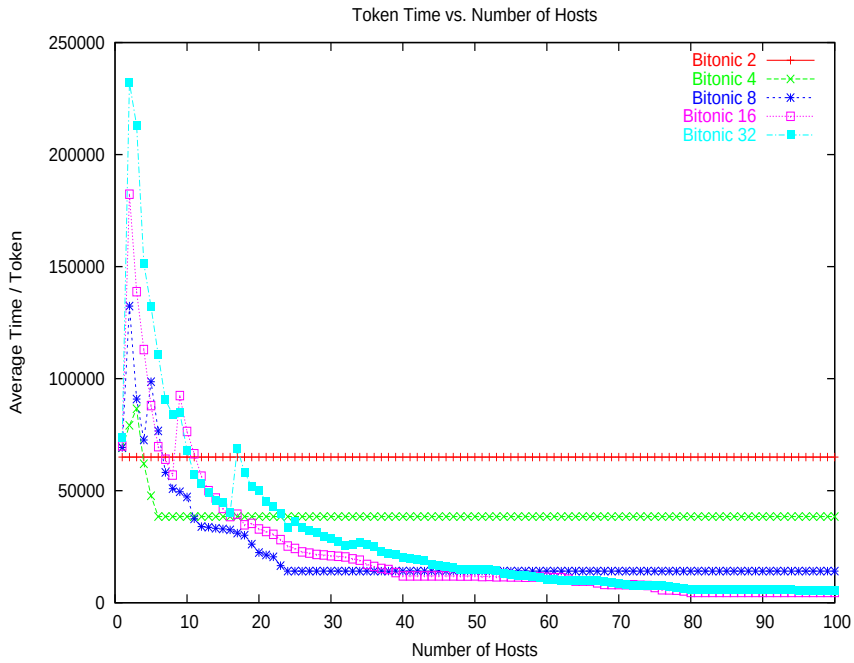

Figure 5: Performance of various sizes of the bitonic network on variable number of hosts

distribution algorithm. The parameters are set to $balancer\_delay = 10$ and $network\_delay = 100$. The *source_delay* ranges from 1 to 100. At most 100 tokens are generated, and the experiment runs for at most 10^6 time units.

Figure 6 shows the results of our experiment. The periodic network performs the worst because it has both the largest depth and the most balancers. The BM network has better performance under the highest load situations, with a source delay between 2 and 10, but the bitonic network performs a little better when the load is decreased with source delay greater than 10. Since the BM network requires more balancers than the bitonic network of the same input width, the decrease in performance of the BM network could be due to the increased contention of having more balancers placed on each host. We also conducted experiments where we vary the *network_delay* by a factor of 5 slower and faster. As observed before, the counting network performance scales by a factor of 5, including the points where the curves level off.

3 Real System Implementation

The purpose of this section is to validate that a real counting network implementation and the COST simulation have similar performance characteristics. The experiments of the real implementation were performed using 10 UltraSPARC-10 servers. Balancers are assigned to the hosts using the longest-run distribution algorithm.

Figure 7 depicts the performance of the counting networks. Note that the curves are with respect to the number of tokens injected per time step, since *source_delay* is not any more a parameter that can be controlled by the hosts as in the simulations. In particular, the x axis corresponds to the total number of tokens which are generated at every microsecond, which is between 1 to 100;

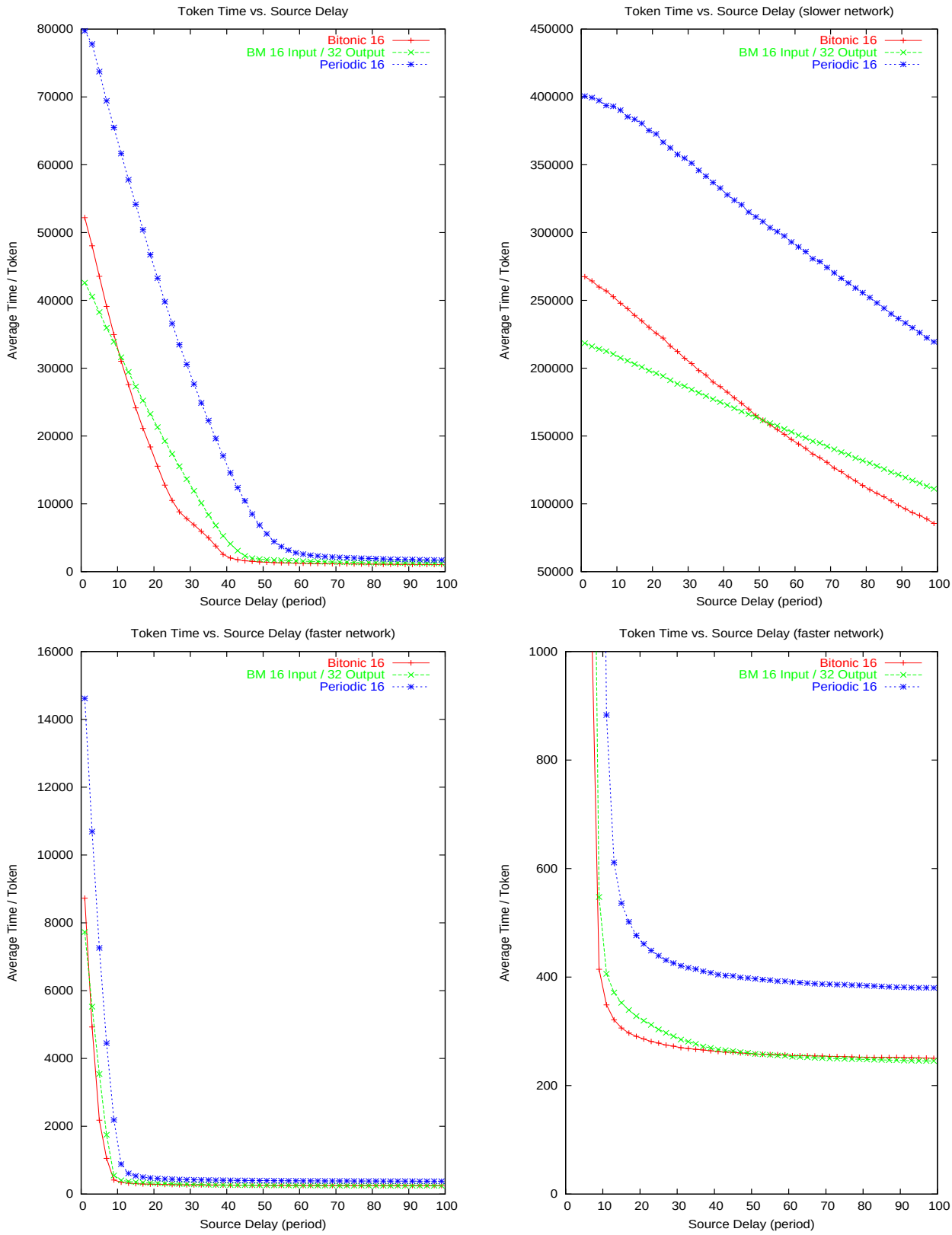

Figure 6: Comparing various counting network constructions; Top left: original experiment; Top right: slow network; Bottom left: fast network; Bottom right: zoom in the fast network graphs

we will refer to this number as *token_load*. The total number of tokens generated and propagated through the network are $100 \cdot token\_load$. We observe, that as a greater token load is placed on the network, the performance of all network decreases due to contention.

The top part of Figure 7 depicts the comparative performance of the periodic, bitonic, and BM networks of input width 8 and 16. The performance of the BM is comparable with the bitonic but with higher load, the BM performs better. The improvement in performance of the BM becomes more clear as the input width increases from 8 to 16. The performance of the periodic network is inferior to the other two networks due to its larger depth.

The bottom of Figure 7 depicts the comparative performance of the various sizes of the bitonic network and BM networks (we do not show the periodic network because its performance is inferior due to its larger depth). Because the number of hosts is small, the performance decreases with larger counting networks, due to contention. The best performance is for width 4. Overall, the implementation on a real system has similar characteristics as with the simulations.

4 Conclusions

We performed simulations of counting networks and provided real implementations that verify the simulated results. Our simulation helps determine the best counting network construction, size, and distribution method that should be used under various load conditions.

We introduced and studied four novel methods for distributing the balancers in a counting network among multiple hosts. Using our simulation, we found that the longest-run distribution algorithm usually outperforms the other three distribution methods that we use in this paper. When the counting network is subjected to a condition causing maximum contention, the trivial distribution method performs better than the longest-run distribution method. We also found that the BM counting network outperforms the bitonic and periodic networks under most conditions. The bitonic counting network performs better than the BM counting network under conditions of minimal contention.

The architecture of our simulation software can be extended to support any balancing network. Furthermore, our software makes it easy to introduce new algorithms for distributing balancers among hosts. We believe that there are balancer distribution algorithms that perform better than longest-run distribution, and their discovery requires further investigation. Our results support the benefit of using counting networks to relieve contention in algorithms. We believe that our counting network simulation and implementation will facilitate using counting networks in distributed applications. Our counting network simulation can be used to help determine the best counting network that performs best under a variety of conditions.

References

[1] A. Agarwal, R. Bianchini, D. Chaiken, K. L. Johnson, D. Kranz, J. Kubiatowicz, B.-H. Lim, K. Mackenzie, and D. Yeung. The MIT Alewife machine: architecture and performance. In *Proceedings of the 22nd annual international symposium on Computer architecture*, pages 2–13, S. Margherita Ligure, Italy, June 1995.

[2] J. Aspnes, M. Herlihy, and N. Shavit. Counting networks. *Journal of the ACM*, 41(5):1020–1048, September 1994.

[3] K. E. Batcher. Sorting networks and their applications. In *Proceedings of AFIPS Joint Computer Conference*, volume 32, pages 307–314, April 1968.

[4] C. Busch and M. Herlihy. A survey on counting networks. In *Proceedings of Workshop on Distributed Data and Structures*, pages 13–20, March/April 1998.

[5] C. Busch and M. Herlihy. Sorting and counting networks of arbitrary width and small depth. *Theory of Computing Systems*, 35(2):99–128, January 2002.

[6] C. Busch and M. Mavronicolas. An efficient counting network. In *Proceedings of the First Merged IPPS & SPDP Symposium*, pages 380–385, 1998.

[7] G. Chen and B. K. Szymanski. Cost: A component-oriented discrete event simulator. In *Proceedings of the 2002 Winter Simulation Conference*, pages 776–782, 2002.

[8] M. Dowd, Y. Perl, L. Rudolph, and M. Saks. The periodic balanced sorting network. *Journal of the ACM*, 36(4):738–757, October 1989.

[9] M. Herlihy, B.-H Lim, and N. Shavit. Scalable concurrent counting. *ACM Transactions on Computer Systems*, 13(4):343–364, November 1995.

[10] M. Herlihy and S. Tirthapura. Self stabilizing smoothing and balancing networks. *Distributed Computing*, 18(5):345–357, 2006.

[11] E. N. Klein. A generic simulation of counting networks. Master's thesis, Computer Science Department, Rensselaer Polytechic Institute, July 2003.

[12] D.A. Kranz, R. Halstead, and E. Mohr. Mul-T: A high-performance parallel lisp. In *Proceedings of the SIGPLAN '89 Conference on Programming Language Design and Implementation*, pages 81–90, Portland, OR, June 1989.

[13] R. L. Rivest T. H. Cormen, C. E. Leiserson and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[14] S. Tirthapura. Adaptive counting networks. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2005.

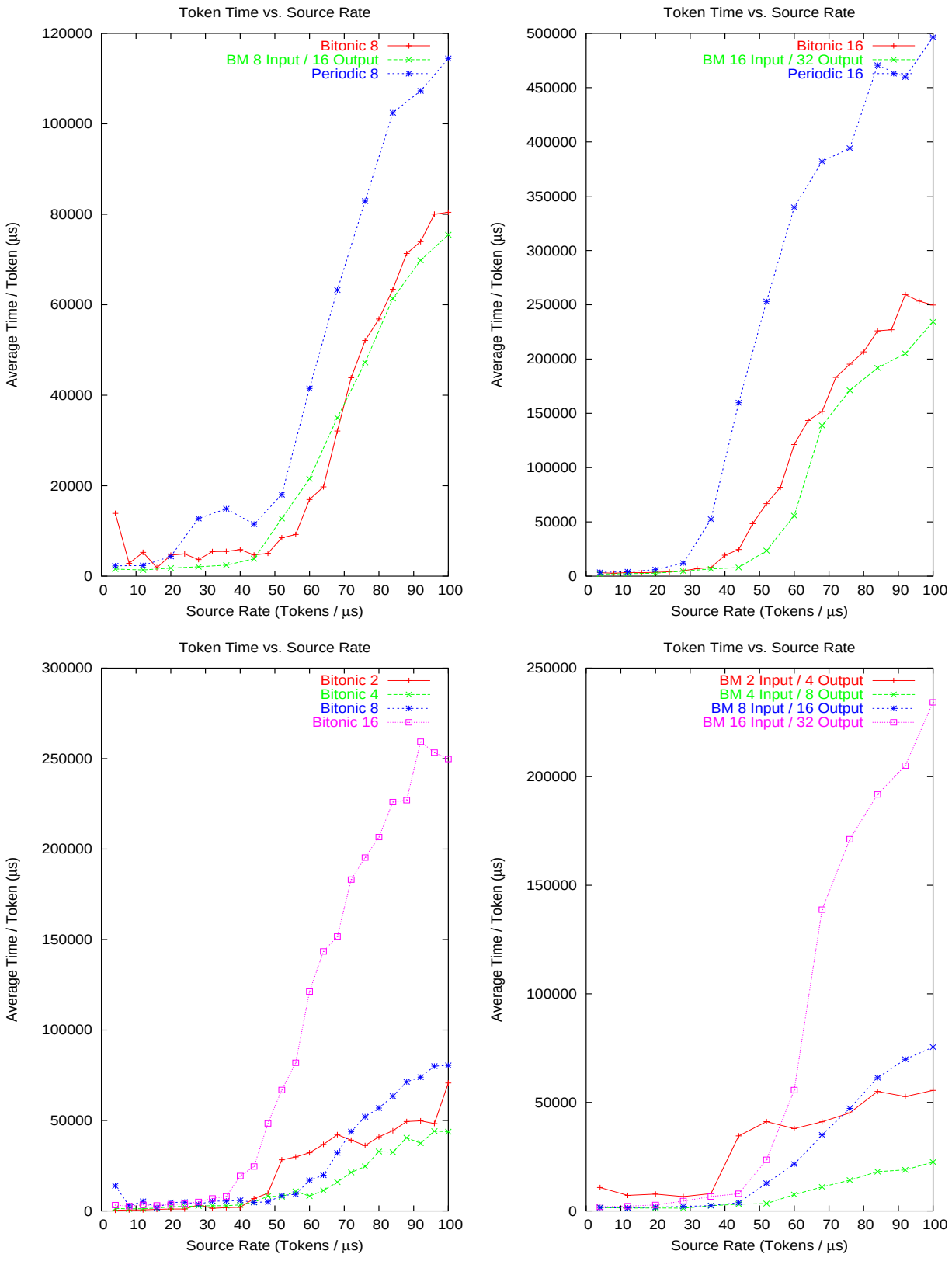

Figure 7: Real system implementations of counting networks; Top: comparative performance of different networks with input widths 8 and 16; Bottom: performance of various network sizes of the bitonic and BM networks