Metrics for Metrics: Why It Is Difficult to Compare Interconnection Networks OR How Would You Compare an Alligator to an Armadillo?

Howard Jay Siegel John K. Antonio

Parallel Processing Laboratory
School of Electrical Engineering
Purdue University
West Lafayette, IN 49707-1285

Kathy J. Liszka

Department of Mathematical Sciences
University of Akron
Ayer Hall 325
Akron, OH 44325-4002

Abstract

Difficulties associated with metric-based design and evaluation of interconnection networks, in the context of massively parallel processing systems, are discussed from a relatively philosophical perspective. The issues of "which metric or weighted set of metrics to use" and "how to meaningfully apply metrics" are central themes throughout the paper. It is difficult to isolate meaningful comparative network performance measurements when using application benchmarks on different systems. Furthermore, the strong interdependence among application, system, and network parameters makes it difficult to apply the classical experimental method, which is to "measure" the effect of changing one parameter while holding all other parameters constant. The issue of fairness is discussed in the context of the classical experimental method, and one aspect of fairness related to algorithm mapping is illustrated through an example.

1: Introduction

A myriad of networks have been proposed in the literature for providing the inter-processor communications in a parallel machine (e.g., see [11, 26, 34]). Given this wide variety of approaches, there is much interest in a methodology for the direct comparison of networks with the goal of determining the "best" approach. The problem is one of determining which metric or weighted set of metrics should be used, and how the metrics should be applied to yield meaningful information. There exists a plethora of performance measures that can be used to evaluate networks, including message delay, traffic throughput, fault tolerance, average distance between nodes, maximum distance between nodes, ease of use, and cost effectiveness. In general, it is difficult to select the "important" metrics, because they may vary with the intended application and operating assumptions.

The central themes of this paper are the issues of which metric or weighted set of metrics to use to compare networks, and how to apply these metrics in a meaningful way. Problems associated with the metric-based design and evaluation of interconnection networks for massively parallel processing systems are discussed from a relatively philosophical perspective.

Consider the features that could be used to compare an alligator to an armadillo. In some ways the two are very similar: both have four legs, both have a rugged exterior, both have sharp claws. However, in other ways the two are not very similar: one prefers a marshy environment, the other dry land; one has a long tail, the other a short tail; one is a reptile, the other a mammal. Which of the two, then, is a better animal? What makes one animal (or network) "better"? Is it which one could win in a fight? Alligators are inherently bigger than armadillos, but to compare fairly how well they fight must you consider two that are of equal weight? Or should they be of equal length? Analogously, to compare the average message delay, for a given set of traffic conditions, between a hypercube network and a mesh network, should you compare two networks where the number of links they employ is the same? Or should it be two networks that support the same number of independent processing nodes? These are the types of issues raised in this paper.

Given a set of measures, the next step is to collect network performance data for applying these metrics.

This research was supported in part by Rome Laboratory under contract numbers F30602-92-C-0150 and F30602-92-C-0108.

0-8186-4410-2/93 $03.00 © 1993 IEEE

Two sources of data that can be used are theoretical models and benchmarks on actual systems. In the case of theoretical models, the problem of the practicality of the model becomes a factor. When using application benchmarks on different systems, the many "layers" of software between a task description and its implementation make it difficult to isolate meaningful comparative network performance measurements. The strong interdependence among application, system, and network parameters makes it difficult to apply fairly the classical experimental method, which is to "measure" the effect of changing one parameter while holding all other parameters constant.

The paper is organized in the following manner. In Section 2, a brief overview of network metrics is given to demonstrate the wide range of metrics available to evaluate network performance. Section 3 then addresses the topic of network parameters that can be varied to change network performance. Section 4 discusses modeling and benchmarking as approaches to measuring performance. In Section 5, the classical experimental method is considered in the context of comparing interconnection networks. The issue of fairness of comparison with respect to algorithm mapping is illustrated through an example in Section 6. The example raises the question of whether algorithm mapping should be held constant when applying a benchmark application to compare two topologically distinct networks. A summary of some of the issues raised is included in the final section.

2: Network metrics

There are many possible ways to characterize and measure the performance of interconnection networks. Given below is a collection of metrics commonly found in the literature. This collection is only a representative subset of the metrics that have been proposed in the past; it is here presented to demonstrate the range of metrics one can consider when evaluating networks. Just as this section contains only a representative subset of possible metrics and is not meant to be exhaustive, the references cited for the various metrics are only representative.

One of the fundamental features for evaluating an interconnection network is the set of *delay characteristics*. When data is transferred among the interconnected processors and/or memories, delays through the interconnection network generally increase processor idle times, which can ultimately degrade overall system performance. In *MIMD* (multiple instruction stream – multiple data stream) systems, multitasking can be used to help limit processor idle time caused by network latency, however, this approach may not improve the total execution time of any single task. There are various ways to characterize network latency, including *maximum delay*, *minimum delay*, and *average delay*. Depending upon the particular application, one of these characterizations may be more appropriate than the others. An example of a study that uses delay as the performance measure is [1]. The hypercube and multistage cube are evaluated by varying the buffer size for packets with delay used as one of the performance metrics.

An important characteristic of a network, especially when used within an *SIMD* (single instruction stream – multiple data stream) system, is its *permuting ability*. For a system of N processors numbered from 0 to $N-1$, a *permutation* is a set of source-destination pairs that are mathematically representable as a bijection of the set $\{0,1,\ldots,N-1\}$ onto itself [28]. In this context, a permutation is the transferring of a data item from each processor to a unique other processor, with all processors transmitting simultaneously. There are N! possible permutations, and different networks generally require differing amounts of time to process various permutations and classes of permutations [19].

Another criterion for network performance is *partitionability*, the capability to partition the network into independent subnetworks, each with the same properties as the original network [26]. Partitionable systems include multiple-SIMD machines (e.g., [31]), reconfigurable mixed-mode machines (e.g., PASM [7, 27, 29]), and MIMD machines (e.g., [14]). Advantages of having a partitionable system include: allowing multiple users, fault tolerance, and subtask parallelism [29].

Fault tolerance is another performance feature to consider when evaluating a network. The environment in which the network must operate will influence the importance of this measure (e.g., a system in an inaccessible satellite versus a system used in a university). To compare the fault tolerance of different networks, one must establish a common fault model and a common fault-tolerance criterion [4]. The *fault model* characterizes all faults that may occur (e.g., permanent and transient versus only permanent). The *fault-tolerance criterion* is the condition that must be met for the network to be said to have tolerated the fault (e.g., finding an alternate path through a network versus having to send a message through a multistage network first to an intermediate destination and then in a second pass to the final destination). Once a common model is established, then metrics such as number of faults tolerated and the degradation, if any, that results from a fault, can be quantified.

Some networks have the unique path property, while others are multipath networks. A *unique path* net-

work has only one path between any given pair of source processor and destination processor (e.g., multistage cube [26]). A *multipath network* has multiple paths between any source-destination pair (e.g., hypercube [14], extra stage cube [5], and multipath omega [24]). Multipath networks may be more costly and more difficult to control than unique path networks, however, the multiple paths they provide can provide fault tolerance and/or be used to route around busy links. As is usually the case, metrics affect one another, and the metrics that are weighed most heavily (e.g., fault tolerance versus ease of control) may result in different comparative assessments of networks.

Graph theory can be used to examine attributes of both multistage and *single-stage* networks (which are "point-to-point" networks such as the mesh or hypercube). A single-stage network can be modeled as a graph, with the nodes representing processors and the arcs representing communication links. Various graph parameters can be adapted for comparing networks. Two examples are diameter and degree. The *diameter* is defined to be the maximum distance between any two nodes (maximum number of communication links that must be traversed to send an inter-processor message in the worst case). The *degree* of a network node is the number of arcs incident to it (the number of communication links connected to a processor). Thus, graph attributes such as these can describe properties of networks. The use of "banyan graphs" for this purpose is discussed in [21].

A metric that characterizes a fundamental aspect of a network's topological structure is called the *bisection width*; the minimum number of wires that must be cut to separate the network into two equal halves [30]. A bisection width constraint is useful when considering implementation issues [6].

Another important consideration when comparing interconnection networks is the practicality of implementation. For example, in [2] a variety of multi-dimensional meshes (including the hypercube) are compared under a particular set of pin-out constraints. This involves examining the balance between the number of channels associated with a processor and the width of each channel.

There are other network metrics that are difficult to define quantitatively, but that are very important to consider. For instance, the ease of use of the network is important from a practical point of view. Is it "easy" for the user to understand and make effective use of the important features of the network? Related to ease of use is the ease of extracting maximum performance. Can the user and/or compiler quickly determine ways to utilize the network to obtain the best possible performance? Because certain network types may be better suited for particular applications than others, the effectiveness for certain classes of problems may be a meaningful feature of a network. Also, some networks may be desirable because of their effectiveness for general purpose computing. These qualitative factors (and others like them) are especially important to consider when designing a system for the commercial marketplace.

Based on any one (or weighted combination of) performance metric(s), the *cost-effectiveness* or *cost-performance ratio* can also be used to evaluate networks. The "cost" should be defined in terms of hardware required, rather than actual dollar cost, which can be easily changed by a new marketing policy rather than a technological advance. However, equating "costs" of different types of hardware without converting to dollars is awkward (e.g., how should one trade-off the "costs" of additional optical links versus increased buffer size in a consistent and fair way?).

It is essential when trying to make a real choice between networks to implement that the system designer identify what metrics are most important. Basing design decisions on different sets of metrics will result in different design choices. For example, in [3] the binary hypercube and twisted cube topologies are examined. The twisted cube has a diameter that is approximately one-half of that of the hypercube. However, in some cases the hypercube will have a smaller message delay. One must be careful in considering the meaning and implications of the various performance metrics that have been proposed in the literature.

The complexity of controlling a network is an important factor. For example, a multistage cube network, such as used in the IBM RP3 [25], may be readily controlled in a distributed fashion just using the address of the destination processor [18], while a "Benes" network, such as used in the IBM GF11 [8], requires a centralized control scheme and O(N logN) time to compute the network switch settings (for N processors) [9]. A Benes network can perform any permutation, while the multistage cube cannot. Thus, this is another example of different metrics leading to contradicting comparisons.

Thus, there are a great number of qualitative and quantitative metrics that can be considered, as well as weighted sets of metrics. The problem is compounded by the numerous design and use parameters that may be varied in the comparison. Some of these are discussed in the next section.

3: Parameters to vary for comparison and evaluation

The previous section reveals a wide range of performance evaluation measures. The present section considers how various network parameters and features affect performance. Two examples are given, followed by a list of relevant factors.

One such parameter is the assumed *interprocessor traffic pattern*. The assumption of uniform traffic (i.e., the likelihood that any processor will communicate with any other processor or memory is equal for all processors) generally simplifies mathematical analysis, and is often assumed. However, implementations of realistic applications often have nonuniform traffic patterns. Thus, the *impact of nonuniform traffic* is an important feature to consider when characterizing network performance [17]. An important class of nonuniform traffic, called *hot spot traffic*, occurs when bursts of multiple requests are directed at the same network port [16, 33]. In practice, this can occur when multiple processors make requests to the same memory word (within the same memory module) during a synchronization phase.

A network feature that can lessen performance degradation caused by hot spot traffic is the ability to perform *combining*, which, in this context, is the use of the network to combine simultaneous requests from multiple processors to the same memory word. Combining merges two messages into one that continues on to the memory module, instead of sending both of the original requests [13]. Combined messages can also be combined. This network feature may have significant impact when shared variables are used for tasks that require large numbers of processors to access them frequently, e.g., a job queue or a semaphore (synchronization primitive).

A partial list of parameters that can be varied for the comparison and evaluation of interconnection networks is given next. There is some functional overlap among many of these parameters. This overlap increases the complexity of isolating the effect of any single parameter.

- network loading
  - method of task decomposition and mapping
  - frequency of communications
  - message sizes and distribution of sizes
  - message destination assumptions
  - nonuniform versus uniform traffic patterns [17, 20]
  - hot spots [16]
  - assumed processor speed
  - assumed memory speed
  - modes of parallelism supported
  - statistically-based simulations
  - utilization of program trace information
- routing control
  - distributed
  - centralized
  - table based
  - time required to compute
- switching methodology
  - packet [10]
  - circuit [22]
  - virtual cut-through [15]
  - wormhole [12]
  - combining [13]
  - conflict resolution scheme
- hardware implementation
  - design details
  - implementation technology
  - number and width of communication links [2]
  - packaging constraints [23]
  - number of pins per chip
  - number of chips per board
  - number of layers on PC boards
  - manufacturability issues
  - range of scalability

For comparison, which parameters should be held constant and which should be varied? What values or choices should be selected for the parameters that will be held constant? These questions are discussed further in the next two sections.

4: Modeling and benchmarking

Mathematical models and parameters associated with mathematical models can provide important insight into basic network features. However, when modeling and analyzing networks for parallel processing systems, assumptions are often made for the sake of tractability that may not be totally realistic. Thus, care must be taken when interpreting results based on mathematical models (i.e., the underlying assumptions must be considered). For example, when analyzing packet-switched networks mathematically it is sometimes assumed that the buffers for holding packets at each switch are of infinite length. It is important to show (usually through some example simulation studies) that the analytical results obtained in such a way are reasonable approximations of behavior when buffer sizes are realistic.

Another possible way to characterize and evaluate the performance of interconnection networks is to use benchmarks. *Benchmarking* involves executing a given task (or collection of tasks) on several different machines and then measuring and comparing the performance among the systems. One problem with applying benchmarking techniques to massively parallel systems is discerning what the benchmarks are measuring.

In addition to the interconnection network, many other factors can also affect the measured performance. For example, the mapping of the benchmark algorithm onto the processors of the system affects the communication pattern, which may affect the delay characteristics. Also, the language/compiler used to express/interpret the algorithm can have a significant impact on performance. The measured performance is also a function of the operating system used to execute the algorithm. Other issues such as the hardware technology and the suitability of the algorithm for the architecture must also be considered, because they too will affect performance. Thus, much care must be taken when interpreting benchmarking results.

An example of a study that used benchmarking to evaluate network performance is [32]. The paper proposes eight benchmark algorithms that are used to measure the maximum, minimum, and average message delay and throughput characteristics of a particular network. Each algorithm is intended to place unique types of communication demands on the network, and thereby reveal its effectiveness under a range of conditions.

When considering the results of benchmarks, one must remember there are many software layers impacting performance before the network is used, and many implementation details in the network itself. To summarize the comments above, the original task must become a conceptual parallel algorithm, which must be coded in some language, compiled by some compiler, and executed in some environment provided by the operating system. All of these software layers can have such an impact on the execution time of the application that the effect of different networks are negligible in comparison. Also, there are many hardware and architectural variations among the machines being benchmarked, such as number and type of processors used, and the underlying implementation technology. Thus, isolating the influence of the choice of the network in the system may be problematic.

5: The classical experimental method

In the natural sciences, phenomena are often studied and new knowledge obtained by applying the experimental method. The *experimental method* assumes that a given system is parameterized by a collection of n variables. To understand the impact that a given parameter, say x, has on the system, all parameters except for x are fixed while x is varied and measurements and comparisons are made on the system.

Can the experimental method be used to evaluate and compare networks? If yes, then is it always possible to fix $n-1$ parameters in a meaningful way? If no, because it is desirable to use the "best" $n-1$ parameters for each experiment, then how can the differences among networks be meaningfully measured?

Consider an example of this problem. Assume, for the purposes of this discussion, that the goal is to compare a particular multi-dimensional mesh network topology with a hypercube network topology, that the hypercube performs fastest using a routing tag control scheme, and the mesh performs fastest using a table lookup control scheme at each switch node. Should the comparisons be made with both using routing tags, both using table lookup, or each using what is best? If the "use what is best" option is selected, how can the time and hardware costs for implementing the two different control schemes be balanced in a fair way so that the comparisons will be meaningful?

As another example of this problem, consider comparing a hypercube network to a multistage cube network to try to decide if the single-stage (point-to-point) approach is better than the multistage approach. Assume, again for the purpose of this discussion, that the hypercube under consideration performs faster with circuit-switching and the multistage cube with packet-switching. Once again, should the comparisons be done with both networks using the same transmission methodology, or should each use the one that is "best" for it? If the "use what is best" option is selected, how can the time and hardware costs for implementing the two transmission schemes be balanced in a fair way so that the comparisons will be meaningful?

In summary, there is a basic dilemma. How can all but one aspect of a pair of networks be held fixed to apply the classical experimental method in a way that is fair and meaningful? If there is no way to do this, and many parameters may be varied so that the best implementation of each network is considered, how is it determined if the two implementations are "equal" and therefore a fair and accurate comparison is being made? This is an important issue that needs to be addressed by

the network community.

6: The problem of using a fixed mapping

The example described below illustrates that algorithm mapping has a strong impact on how well a network performs for a given application. This is related to the issues of both benchmarking and the application of the experimental method. It demonstrates what can happen if the same mapping of a task onto a parallel machine is used to compare the performance of two topologically distinct networks.

Consider two hypothetical parallel processing systems where the only parametric difference between the two systems is the topological structure of their interconnection networks; one is a mesh, the other a ring. Other system parameters are assumed to be identical, including the number of independent processing elements (processor-memory pairs or *PEs*), the computing and communication hardware, the operating system, the language, the communication protocols, and the link bandwidth. Figure 1 shows the connection pattern among the PEs for both the mesh topology and the ring topology. Both networks have N PEs labeled from 0 to $N-1$. In the mesh topology, PE i is directly connected to PEs $i-1$, $i+1$, $i+\sqrt{N}$, and $i-\sqrt{N}$. In the ring topology, PE i only has direct connections to PEs $i-1$ and $i+1$.

In an attempt to apply benchmarking or the classical experimental method to compare the impact of network topologies while holding all other factors constant, the proposed action is to execute the same SIMD image smoothing algorithm on both systems and compare overall execution times for the two architectures. The smoothing algorithm involves performing a smoothing operation for each pixel in a given $M \times M$ image. Letting A(I,J) denote the original value of the pixel located at row I and column J of the image, its smoothed value is defined by

$$A'(I,J) = [A(I,J) + A(I-1,J) + A(I+1,J) + A(I,J-1) + A(I,J+1) + A(I-1,J-1) + A(I-1,J+1) + A(I+1,J-1) + A(I+1,J+1)]/9. \quad (1)$$

Figure 2 depicts how pixel values from an image are mapped onto PEs for the case of a 512×512 image and an 32×32 (1024 PE) mesh. As shown in the figure, the pixel values from each of the 1024 16×16 subimages are mapped onto each of the 1024 PEs so that adjacent subimages are mapped to adjacent PEs in the mesh.

During phase 1 of the algorithm, each PE executes the smoothing operation of Eq. (1) for those pixels within the interior of each subimage. For this phase, each PE has all the pixel values required for all the smoothing operations stored locally, thus no inter-PE data transfers are required. During phase 2 of the algorithm, the 16 pixel values along each of the four boundaries of each subimage are transferred to neighboring PEs and the four corner pixels of each subimage are transferred to PEs that are "diagonal neighbors" (through intermediate PEs that are directly connected). In phase 3 of the algorithm, the smoothing operation is applied to those pixels on the boundary of each subimage. For this example, a total of $(16)^2 = 256$ smoothing operations are done, $(14)^2 = 196$ in phase 1 and $4 \cdot 15 = 60$ in phase 3, and a total of $(4 \cdot 16) + 4 = 68$ inter-PE data transfers are required in phase 2.

The mapping for the general case of an $M \times M$ image and an $\sqrt{N} \times \sqrt{N}$ mesh, is depicted in Figure 3. As before, pixel values from N subimages of size $(M/\sqrt{N}) \times (M/\sqrt{N})$ are mapped onto the N PEs so that adjacent subimages are mapped onto adjacent PEs in the mesh. It is straightforward to verify that the smoothing algorithm requires a total of M^2/N smoothing operations and $4(M/\sqrt{N}) + 4$ inter-PE data transfers on the N PE mesh [27].

For the case of 16 PEs, Figure 4 illustrates the required data transfers to PE 6 for both the mesh and ring topologies. It is assumed that image is distributed among the PEs for the ring architecture in the same way described earlier for the mesh (as in Figure 2).

For the general problem of smoothing $M \times M$ images using N PEs, it is shown below that a total of $2M/\sqrt{N} + 2M + 4$ inter-PE data transfers are required for the ring architecture. The $M/\sqrt{N}$ pixels along the right vertical boundary of each subimage must be transferred from each PE $i-1$ to PE i. This requires a total of $M/\sqrt{N}$ data transfers. Likewise, transferring the $M/\sqrt{N}$ pixels associated with each left vertical subimage boundary from each PE $i+1$ to PE i also requires $M/\sqrt{N}$ data transfers. Thus, transferring both vertical subimage boundaries requires a total of $2M/\sqrt{N}$ data transfers. To transfer the $M/\sqrt{N}$ pixels along the upper horizontal subimage boundary from each PE $i+\sqrt{N}$ to PE i requires $\sqrt{N}(M/\sqrt{N}) = M$ data transfers, because PE $i+\sqrt{N}$ and PE i are separated by $\sqrt{N}$ links. Likewise, to transfer the $M/\sqrt{N}$ pixels along the lower horizontal subimage boundary from each PE $i-\sqrt{N}$ to PE i requires M data transfers. Thus, a total of 2M data transfers are required to transfer both horizontal subimage boundaries. Con-

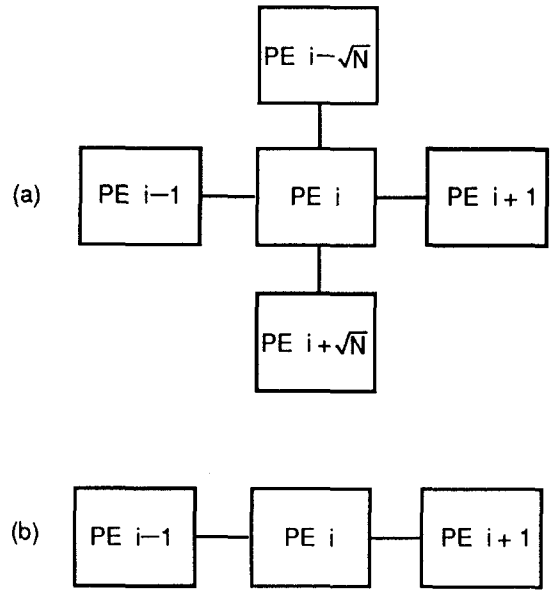


Figure 1. The connection patterns for the (a) mesh and (b) ring topologies.

sider sending the corner pixels needed by PE i. As with the mesh network, pixels from the "diagonal neighbors" can be sent through intermediate PEs. The needed pixel values from the upper and lower right diagonal neighbors will have already been transferred into PE i+1, so only two transfers are needed to move them to PE i. The case for the pixel values from the upper and lower left diagonal neighbors is similar. Therefore, a total of four transfers is needed for the diagonal pixels. The grand total for the ring is $2M/\sqrt{N} + 2M + 4$ inter-PE data transfers. For $1 < \sqrt{N} \leq M$, this is greater than the required number of inter-PE data transfers for the mesh, given by $4(M/\sqrt{N}) + 4$.

Is it fair to conclude that the ring topology is inferior to the mesh, even just for this application, as the above analysis seems to indicate? Consider the following variation in the mapping scheme for the previously described image smoothing algorithm. Instead of dividing the $M \times M$ image into N *square* $(M/\sqrt{N}) \times (M/\sqrt{N})$ subimages, divide it into N *rectangular* $(M/N) \times M$ subimages. A mapping of pixel values from these N rectangular $(M/N) \times M$ subimages onto the ring topology is shown in Figure 5. During phase 1 of the image smoothing algorithm, the smoothing operation is applied to the interior pixels of each subimage (same as before). In phase 2, pixel values along the horizontal boundaries of the rectangular subimages are transferred to neighboring PEs, which requires a total of 2M inter-PE data transfers for the ring topology. In phase 3, the smoothing operation is applied to the boundary pixels (same as before). Thus, for the rectangular subimage mapping scheme only 2M data transfers are required.

Consider if the N rectangular $(M/N) \times M$ subimages approach is used on a system with a mesh network without "wraparound" connections (e.g., no connections from PE 31 to PE 32 in the example shown in Figure 2). Then the number of inter-PE transfers needed would be greater than 2M because not every PE i would be directly connected to PE i+1. Thus, for this mapping strategy, the ring architecture has a better performance than the mesh architecture.

Thus, the example in this section demonstrates two points made in earlier sections. First, when benchmarking, in some cases, the choice of algorithm can determine comparative network performance, rather than any property of the networks themselves. Second, it illustrates the problem of trying to decide what to hold constant and what should be changed to do fair and accurate experiments to compare networks.

7: Summary

The goal of the discussions in this paper has been to illustrate that there are many open fundamental questions in the area of evaluating and comparing interconnection networks. These include:

Which metrics or weighted collection of metrics should be used to evaluate network performance?

Can current or future analytically tractable theoretical models demonstrate sufficiently realistic behavior?

How can benchmarks across different machines be devised to evaluate network features?

How can the classical experimental method be applied fairly to evaluate networks?

What parameters should be held constant when comparing and evaluating networks?

In what sense can the implementation of two different networks be made "equal" for a meaningful comparison?

These open questions are fertile ground on which significant research results can grow. It is important to try to find answers to these questions so that the comparative "goodness" of various network features can be quantified.

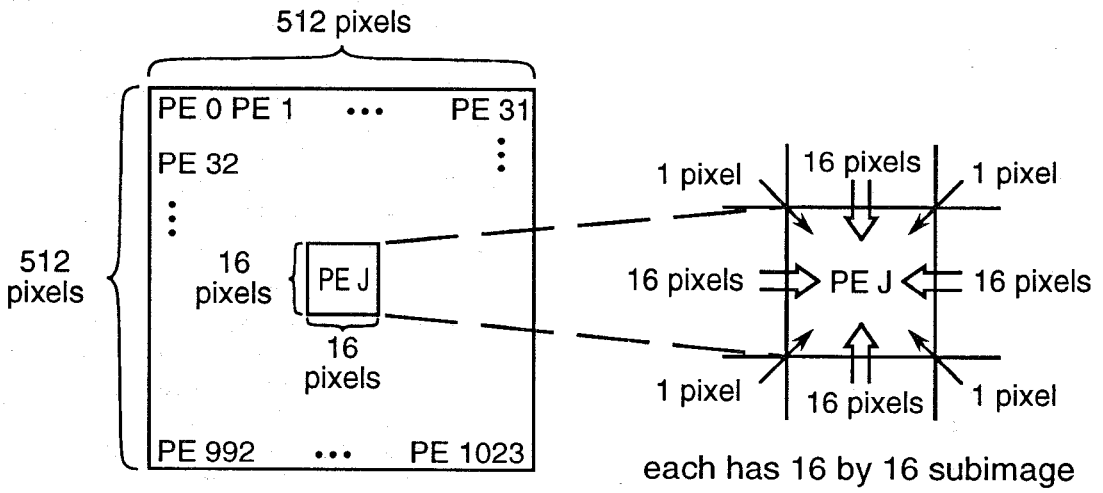


Figure 2. Mapping a 16×16 subimage onto the PE of a mesh.

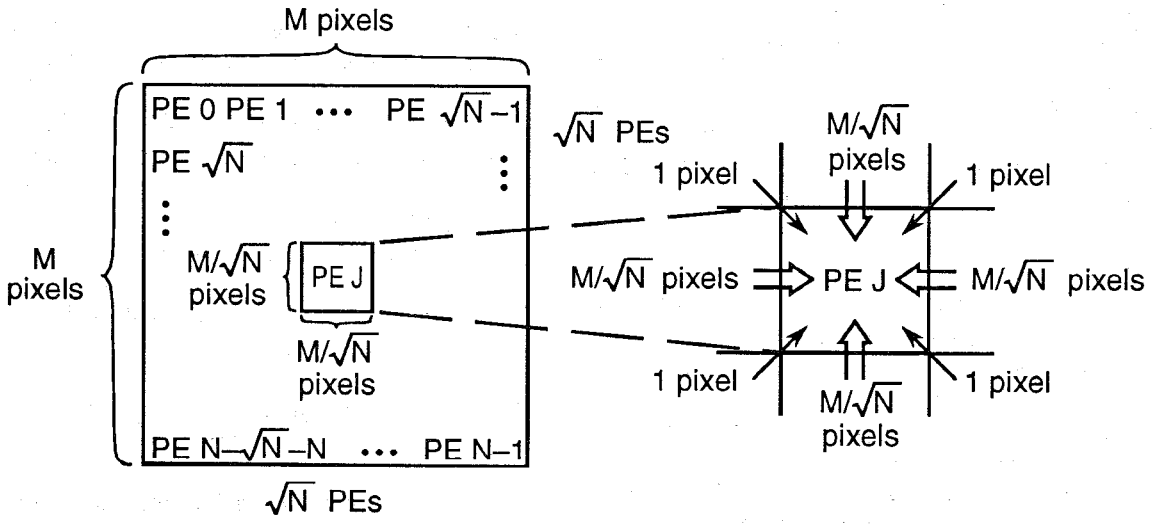
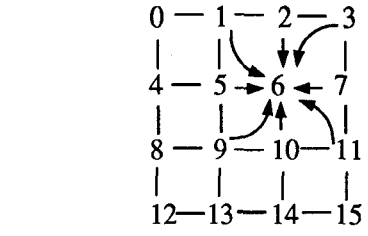


Figure 3. Mapping a (M/√N)×(M/√N) subimage onto the PE of a mesh.



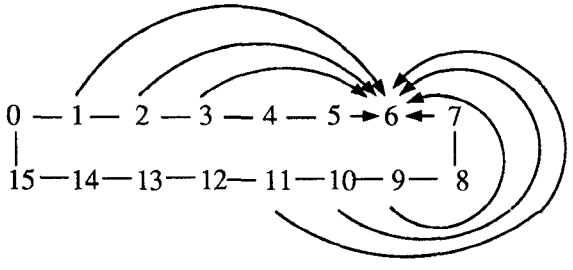


Figure 4. The required data transfers to PE 6 for both the mesh and ring topologies.

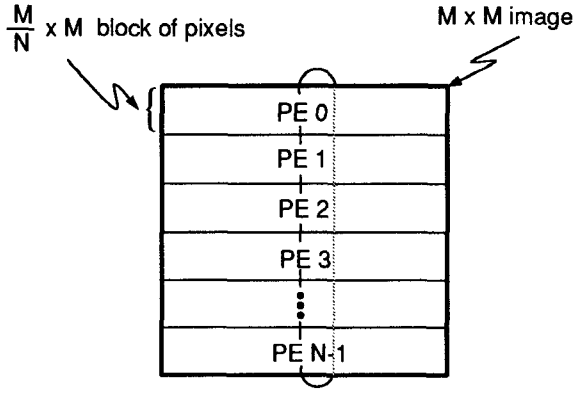


Figure 5. Mapping (M/N)×M subimages onto the PEs of the ring.

Acknowledgments: The authors thank J. McWaid for her comments.

References

[1] S. Abraham and K. Padmanabhan, "Performance of the direct binary n-cube network for multiprocessors," *IEEE Trans. Computers*, vol. 38, no. 7, July 1989, pp. 1000-1011.

[2] S. Abraham and K. Padmanabhan, "Performance of multicomputer networks under pin-out constraints," *J. Parallel and Distributed Computing*, vol. 12, no. 3, July 1991, pp. 237-248.

[3] S. Abraham and K. Padmanabhan, "The twisted cube topology for multiprocessors: a study in network asymmetry," *J. Parallel and Distributed Computing*, vol. 13, no. 1, Sept. 1991, pp. 104-110.

[4] G. B. Adams III, D. P. Agrawal, and H. J. Siegel, "A survey and comparison of fault-tolerant multistage interconnection networks," *Computer*, vol. 20, no. 6, June 1987, pp. 14-27.

[5] G. B. Adams III and H. J. Siegel, "The extra stage cube: A fault-tolerant interconnection network for supersystems," *IEEE Trans. Computers*, vol. C-31, no. 5, May 1982, pp. 443-454.

[6] A. Agarwal, "Limits on interconnection network performance," *IEEE Trans. Parallel and Distributed Systems*, vol. 2, no. 4, Oct. 1991, pp. 398-412.

[7] J. B. Armstrong, D. W. Watson, and H. J. Siegel, "Software issues for the PASM parallel processing system," in *Software for Parallel Computation*, edited by J. S. Kowalik and L. Grandinetti, Springer-Verlag, Berlin, Germany, 1993, pp. 134-148.

[8] J. Beetem, M. Denneau, and D. Weingarten, "The GF11 supercomputer," *12th Ann. Int'l Symp. Computer Architecture*, June 1985, pp. 108-115.

[9] V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press, New York, 1965.

[10] L. N. Bhuyan, "Interconnection networks for parallel and distributed processing," *Computer*, vol. 20, no. 6, June 1987, pp. 9-12.

[11] G. Broomell and J. R. Heath, "Classification categories and historical development of circuit switching topologies," *ACM Computing Surveys*, vol. 15, no. 2, June 1983, pp. 95-133.

[12] W. J. Dally and C. L. Sietz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Computers*, vol. C-36, no. 5, May 1987, pp. 547-553.

[13] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir, "The NYU Ultracomputer – designing an MIMD shared-memory parallel computer," *IEEE Trans. Computers*, vol. C-32, no. 2, Feb. 1983, pp. 175-189.

[14] J. P. Hayes and T. Mudge, "Hypercube supercomputers," *Proceedings of the IEEE*, vol. 77, no. 12, Dec. 1989, pp. 1829-1841.

[15] P. Kermani and L. Kleinrock, "A tradeoff study of switching systems in computer communications net-

works," *IEEE Trans. Computers,* vol. C-29, no. 12, Dec. 1980, pp. 1052-1060.

[16] M. Kumar and G. F. Pfister, "The onset of hot spot contention," *1986 Int'l Conf. Parallel Processing,* Aug. 1986, pp. 28-34.

[17] T. Lang and L. Kurisaki, "Nonuniform traffic spots (NUTS) in multistage interconnection networks," *J. Parallel and Distributed Computing,* vol. 10, no. 1, Sept. 1990, pp. 55-67.

[18] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Computers,* vol. C-24, no. 12, pp. 1145-1155, Dec. 1975.

[19] J. Lenfant, "Parallel permutations of data: A Benes network control algorithm for frequently used permutations," *IEEE Trans. Computers,* vol. C-27, no. 7, July 1978, pp. 637-647.

[20] T. Lin and L. Kleinrock, "Performance analysis of finite-buffered multistage interconnection networks with a general traffic pattern," *Performance Evaluation Review,* vol. 19, no. 1, May 1991, pp. 68-78.

[21] G. J. Lipovski and M. Malek, *Parallel Computing: Theory and Comparisons,* John Wiley, New York, 1987.

[22] G. M. Masson, G. C. Gingher, and S. Nakamura, "A sampler of circuit switching networks," *Computer,* vol. 12, no. 6, June 1979, pp. 32-48.

[23] J. R. Nickolls, "Interconnection architecture and packaging in massively parallel computers," *Packaging, Interconnects, and Optoelectronics for the Design of Parallel Computers Workshop,* Mar. 1992, pp. 4-8.

[24] K. Padmanabhan and D. H. Lawrie, "A class of redundant path multistage interconnection networks," *IEEE Trans. Computers,* vol. C-32, no. 12, Dec. 1983, pp. 1099-1108.

[25] G. F. Pfister, W. C. Brantley, D. A. George, S. L. Harvey, W. J. Kleinfelder, K. P. McAuliffe, E. A. Melton, V. A. Norton, and J. Weiss, "The IBM Research Parallel Processor Prototype (RP3): Introduction and architecture," *1985 Int'l Conf. Parallel Processing,* Aug. 1985, pp. 764-771.

[26] H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies, Second Edition,* McGraw-Hill, New York, 1990.

[27] H. J. Siegel, J. B. Armstrong, and D. W. Watson, "Mapping computer-vision-related tasks onto reconfigurable parallel-processing systems," *Computer,* vol. 25, no. 2, Feb. 1992, pp. 54-63.

[28] H. J. Siegel, W. G. Nation, C. P. Kruskal, and L. M. Napolitano, "Using the multistage cube network topology in parallel supercomputers," *Proceedings of the IEEE,* vol. 77, no. 12, Dec. 1989, pp. 1932-1953.

[29] H. J. Siegel, L. J. Siegel, F. C. Kemmerer, P. T. Mueller, Jr., H. E. Smalley, Jr., and S. D. Smith, "PASM: A partitionable SIMD/MIMD system for image processing and pattern recognition," *IEEE Trans. Computers,* vol. C-30, no. 12, Dec. 1981, pp. 934-947.

[30] C. D. Thompson, "A complexity theory for VLSI," Ph.D. dissertation, Dept. Computer Science, Carnegie-Mellon Univ., 1980.

[31] L. W. Tucker and G. G. Robertson, "Architectures and applications of the Connection Machine," *Computer,* vol. 21, no. 8, Aug. 1988, pp. 26-38.

[32] R. Upton and S. Tripathi, "On the performance evaluation of fine-grained SIMD computer architectures: An analysis of the Connection Machine," in *High Performance Computer Systems,* edited by E. Gelenbe, Elsevier Science Publishers, North-Holland, Amsterdam, Holland, 1988, pp. 129-141.

[33] M.-C. Wang, H. J. Siegel, M. A. Nichols, and S. Abraham, "Reducing the effect of hot spots by using a multipath network," *1993 Int'l Conf. Parallel Processing,* Aug. 1993, to appear.

[34] C.-L. Wu and T. Y. Feng, *Tutorial: Interconnection Networks for Parallel and Distributed Processing,* IEEE Computer Society Press, Silver Spring, MD, 1984.