Throughput Models of Interconnection Networks: the Good, the Bad, and the Ugly

Peyman Faizian, Md Atiqul Mollah, Md Shafayat Rahman, Xin Yuan
Department of Computer Science
Florida State University
Tallahassee, FL
{faizian,rahman,mollah,xyuan}@cs.fsu.edu

Scott Pakin, Mike Lang
Computer, Computational, and Stat. Sci. Div.
Los Alamos National Laboratory
Los Alamos, NM
{pakin,mlang}@lanl.gov

***Abstract*—Throughput performance, an important metric for interconnection networks, is often quantified by the aggregate throughput for a set of representative traffic patterns. A number of models have been developed to estimate the aggregate throughput for a given traffic pattern on an interconnection network. Since all of the models predict the same property of interconnection networks, ideally, they should give similar performance results or at least follow the same performance trend. In this work, we examine four commonly used interconnect throughput models and identify the cases when all models show similar trends, when different models yield different trends, and when different models produce contradictory results. Our study reveals important properties of the models and demonstrates the subtle differences among them, which are important for an interconnect designer/researcher to understand, in order to properly select a throughput model in the process of interconnect evaluation.**

I. Introduction

As large scale computer systems such as supercomputers and data centers continually increase not only their total core counts but also their per-node computational performance, the interconnection network performance must increase correspondingly to prevent it from becoming the primary limiter of application performance and scalability. Future interconnect designs have focused on supporting sufficiently large aggregate throughput performance while meeting various constraints such as cost and power.

The throughput performance of an interconnection network is often quantified by the aggregate throughput for a set of representative traffic patterns. Modeling the aggregate throughput for a given traffic pattern on an interconnect is widely employed in interconnection design, exploration, and evaluation. Such a modeling problem can be described as follows: *Given an interconnection network (a topology and a routing scheme) and a traffic pattern that consists of a set of flows, what is the aggregate throughput that can be achieved?*

Common throughput models include Max-Min Fairness (MMF) [1]–[3], Maximum Concurrent Flows (MCF) [4]–[7], Hoefler's method (HM) [8]–[10], and Jain's method (JM) [11]. All of these models have recently been used in practice to evaluate and/or predict interconnect performance. Each of the models includes a method to determine the flow rate for each flow in the traffic pattern. The aggregate throughput is the sum of the rates for all flows in the pattern.

MMF has been accepted in the networking community as a fair resource allocation scheme, and has been the ideal theoretical target for many practical network resource management algorithms. Routing, congestion control, and packet scheduling schemes in many systems thrive to achieve MMF. Informally, for a given network and a traffic pattern, an allocation of rates to the set of flows in the pattern is said to be max-min fair if it is not possible to increase the allocated rate for any flow while decreasing only the rates of flows that have larger rates. Computing the MMF rate allocation often requires to iteratively solve linear programming problems and can be computationally infeasible for large networks. To attain models that can be applied to large-scale systems, MCF, HM, and JM have been developed, all approximating MMF with reduced computational complexity. MCF is the first-order approximation of MMF: instead of using multiple iterations to find the max-min fair rate for each flow, MCF performs the same calculation as MMF, but stops at the first iteration: the rate computed in the first iteration is used as the rate for all flows. HM employs a different approach, estimating the rate for each flow by counting the number of flows using each link. JM is a hybrid of the iterative approach and HM for rate approximation. The details of MMF, MCF, HM, and JM will be described in Section II.

Since all of the models quantify the throughput performance of interconnection networks, ideally, all three metrics should yield similar results. However, approximating MMF in different ways may result in systematic biases for certain network or traffic parameters; and it is unclear whether the different models produce similar performance results or even reveal similar performance trends when they are used for evaluating interconnection designs. In this work, we examine these four commonly used throughput models and identify the cases when all models can give a similar trend (the good), when different models yield different trends (the bad), and when different models produce contradictory results (the ugly). The study reveals important properties of the models and demonstrates the subtle differences among them, which are important for an interconnect designer/researcher to understand in order to properly select a throughput model in the process of interconnect evaluation.

The rest of this paper is organized as follows. In Section II,

we discuss models and describe how the aggregate throughput with different network settings are computed using the models. Section III reports the findings of our study. We discuss related work in Section IV and conclude the paper in Section V.

II. Throughput models

We will first introduce the notations used in this paper. After that, we will formally define Max-Min Fairness (MMF) and Maximum Concurrent Flow (MCF), and describe how the rates for the flows are computed for interconnects with single path and multi-path routing using MMF, MCF, Hoefler's method (HM), and Jain's method (JM). Practical interconnect resource management optimizes not only for throughput but also for fairness. MMF, MCF, TM, and JM all consider both throughput and fairness.

Let S be a set, we denote $|S|$ the size of the set. The network is modeled as $G = (V, E)$, where V is the set of nodes, and E is the set of links in the network. We use link $(i, j) \in E$ to denote a unidirectional link from node i to node j. The corresponding capacity is $C = \{C_{ij} \mid (i, j) \in E\}$. A flow is a communication from a source $s \in V$ to a destination $t \in V$ and is denoted as (s, t). A *communication pattern* is a set of flows.

We consider networks with single-path routing and multi-path routing. For networks with single-path routing, all traffic in each flow follows one path. For networks with multi-path routing, traffic in a flow may split among multiple paths. Given a flow $d = (s, t)$, let there be at most k paths for the flow. For single-path routing, $k = 1$. We denote f_m^d, $m = 1..k$, the m-th path for the flow. Each path consists of a set of links. If f_m^d uses link (i, j), $(i, j) \in f_m^d$.

Given a communication pattern D with $|D|$ flows, a rate allocation scheme decides an allocation vector $\gamma = (\gamma_0, ..., \gamma_{|D|-1})$, where γ_d is the rate allocated to flow $d \in D$. For a communication pattern D, an allocation vector γ is said to be *feasible* if $\gamma_d \geq 0$ for all $d \in D$, and all link capacity constraints are satisfied. Once the rate for each flow is determined, the aggregate throughput for the communication pattern is the sum of all of the rates. MMF, MCF, HM, and JM basically use different methods to compute the feasible rate allocation for a given communication pattern on a network.

A. Max-min fairness (MMF)

An allocation of bandwidth to a set of flows is said to be *max-min fair* if an increase of any flow rate can only be achieved at the cost of a decrease in smaller flow rates [1], [2]. Formally, the MMF rate allocation is defined as follows.

Definition 1 (Max-min fair rate allocation): A feasible allocation γ is max-min fair if and only if for any other feasible allocation γ', if $\gamma'_d > \gamma_d$ for some flow d, then there must exist some flow $d1$ such that $\gamma_{d1} < \gamma_d$ and $\gamma'_{d1} < \gamma_{d1}$.

For networks with single-path routing and multi-path routing, the calculation of MMF rate allocation has different algorithms. The MMF rate allocation for networks with single path routing is referred to as the MMF Single Path routing Problem (MMF-SP). An iterative filling algorithm has been given [1]. The algorithm starts by assigning a 0 rate to all flows. Then, flow rates are uniformly increased until some link is saturated (further increasing the rates will require the link to have more bandwidth than its capacity). At this point, the MMF rate of the flows that use any saturated link is determined; such flows are said to be saturated and are removed from consideration; and the link capacities in the network are modified. The algorithm repeats until the rates for all flows are determined.

The MMF multi-path problem (MMF-MP) considers the more general multi-path routing with traffic in a flow splittable among multiple paths [2]. The algorithm for solving MMF-MP [2], shown in Figure 1, finds the MMF rate allocation iteratively. In each iteration, the maximum rate for all flows under consideration, as well as the set of flows that saturate, is computed by solving the flow-based linear programming formulation showed in Figure 2. The rate for the saturated flows is then fixed for the iteration. This process continues until the rates for all flows are determined.

In Figure 1, D_i is the set of flows saturated at iteration i and L_i is the set of flows saturated in the first i iterations. In each iteration, the algorithm solves the LP formulation in Figure 2 that determines the set (D_k) of flows whose MMF rate can be decided in the current iteration as well as the MMF rate for the flows. The rate for flows in D_k is then set; and L_k is updated to include D_k. The algorithm terminates when the rates for all flows are computed, that is, $L_k = D$.

The LP formulation for identifying saturated flows and computing the rate at iteration k is shown in Figure 2. v_m^d is the rate for f_m^d (the m-th path of flow d), $d \in D$, $1 \leq m \leq k$. $\sum v_m^d$ is the flow rate for flow d. Constraints (1) are capacity constraints: the sum of the rates of all paths that use a link must be no more than the capacity of the link. Constraints (2) fix the flow rate for the flows whose MMF rates have been determined in previous iterations. For $d \in D_{n-1}$, the flow rate γ_d has been calculated in earlier iterations as show in Figure 1. Constraints (3) are traffic constraints for flows whose rates have not been determined. The rate α is to be maximized in the current iteration. Constraints (4) indicate that the flow rate values for each path must be non-negative. This is the path-based LP formulation that assumes the paths for all flows are known.

The LP formulation for the MMF multi-commodity flow problem for the general multi-path routing with splittable flows have also been developed [2]. The solution for the MMF multi-commodity flow problem gives the optimal MMF rate allocation of any multi-path routing scheme. In this paper, we assume that the up-to k paths for each flow are known and use the algorithm described in Figure 1 to compute the rate allocation for MMF-MP.

B. Maximum Concurrent Flow (MCF)

As discussed in the previous sub-section, calculating MMF rates for multi-path routing involves iteratively solving LP problems. In the worst case, $|D|$ LP problems need to be solved, which can be computationally intensive, especially

Data: A capacitated network (V, E, C) and a set of flows D
Result: A max-min fair allocation vector γ of size $|D|$

Set $k = 0$ and $L_0 = \phi$;
while $L_k \neq D$ **do**
- Set k = k+1;
- Solve the MMF LP problem, and compute the maximum rate α for saturated flows;
- Identify the set D_k of saturated flows; set $\gamma_d = \alpha$ for all $d \in D_k$;
- $L_k = L_{k-1} \cup D_k$; update the LP formulation for the next iteration;

end
The flow vector γ obtained at the last step is max-min fair allocation rates for the flows

Fig. 1: Algorithm to calculate MMF rates for multi-commodity flows

Maximize α
Subject to:

$$\sum_{d \in D,\ (i,j) \in f_m^d} v_m^d \leq C_{ij},\ \forall (i,j) \in E \quad (1)$$

$$\sum v_m^d = \gamma_d,\ \forall d \in D_{k-1} \quad (2)$$

$$\sum v_m^d = \alpha,\ \forall d \in D - D_{k-1} \quad (3)$$

$$v_m^d \geq 0, \forall d \in D \quad (4)$$

Fig. 2: Linear programming formulation for iteration k

when the network size or the number of paths is large. The Maximum Concurrent Flow (MCF) is the first-order approximation of MMF. The MCF rate is the maximum attainable throughput for all flows in a pattern. In other words, MCF is the single rate which can be assigned to all flows without violating the link capacity constraints. It is therefore the lower bound of the flow rates for all flows in the traffic pattern, which itself is an important performance metric.

Computing MCF rates can be done by using the same algorithm as that for computing MMF rates. The algorithm for MCF stops at the first iteration in the MMF algorithm; the rate computed in the first iteration is the MCF rate for the traffic pattern.

C. Hoefler's Method (HM)

The Hoefler's Method (HM) was originally proposed for networks with single-path routing [8], [10]. The basic idea is simple: if X flows sharing a link of capacity C_l, then each of the X flows can at most have a rate of $\frac{C_l}{X}$. For a given pattern, this approximation estimates the rate for each flow by counting the number of times each link is used. The throughput of each flow is equal to the smallest bandwidth assigned to it among all links in the path. Let f_1^d be the path for flow d. To compute the flow rate with HM, we first compute the number of times each link is used, $u_{i,j}$, $(i,j) \in E$.

$$u_{ij} = number\ of\ times\ that\ (i,j) \in f_1^d, d \in D$$

Once u_{ij} is computed, the rate for a flow d is the minimum amount of bandwidth allowed by any of the links along the path.

$$\gamma_d = \min_{(i,j) \in f_1^d} \frac{C_{ij}}{u_{ij}}$$

The model was extended for multi-path routing as follows [9]. For multi-path routing, each flow is split into k sub-flows, each using a distinct path. Each sub-flow is treated as a flow in the single path routing case to decide its rate. The rate of a flow with multiple sub-flows is the sum of the rates of all of its sub-flows.

D. Jain's method

Jain's method [11] is a hybrid of HM and the iterative approach for solving the MMF rate allocation. It follows the structure of the iterative algorithm in Figure 1. However, within each iteration, HM is used to compute the additional rate that can be assigned to unsaturated flows. In the following, we will describe the algorithm we use for calculating rate allocation for JM. This algorithm is a minor adaption of the algorithm in [11]. For the network $G = (V, E)$, the algorithm maintains two copies of the graph $G^A = (V^A, E^A)$ and $G^R = (V^R, E^R)$. G^A records the bandwidth that has already been allocated while G^R maintains the remaining bandwidth. Let the traffic pattern be D and each message d have a set of k paths P_d, $|P_d| = k$. Initially, $E^A = 0$, $E^R = E$, all paths are unsaturated. The algorithm repeatedly performs the following steps until all paths are saturated:

1) Use HM to compute the additional rate that can be assigned to all unsaturated paths over G^R.
2) Record the additional rate for each path in the iteration; decrement the remaining bandwidth on links in G^R and increment the allocated bandwidth on links in G^A. If the bandwidth on a link in G^R is reduced to 0, mark all paths that use the link as saturated since no more bandwidth can be allocated to the path.

III. Throughput models: the good, the bad, and the ugly

Performance models can be used in different ways. Among others, the most important use of models includes predicting performance of an interconnect, probing the performance trend of an interconnect, and comparing the performance of different interconnect designs. All of the four models have been used recently in different studies to investigate interconnect performance [3], [5]–[8], [11].

In this study, we compare the four throughput models in predicting the throughput performance of different traffic patterns, different network sizes, and routing schemes with different number of paths for multi-path routing. The objective is to obtain a better understanding of these models. In particular, we would like to answer the following questions: (1) Do the models produce similar performance results for different network topologies, networks with different routing schemes, and different traffic patterns? (2) Do the models produce similar performance trends when network size increases, when the number of paths used in the multi-path scheme increases, and when the traffic pattern changes? (3) Do different models yield similar conclusions when they are used to compare different interconnect designs?

We performed extensive experiments with different topologies, routing schemes, and traffic patterns. To illustrate the main findings, we will only report the results on the selected configurations described in the following.

Two types of network topologies are reported in the study. The first is the random regular topology (also known as the Jellyfish topology [12]) that is recently proposed for future data center networks. The random regular topology consists of randomly connected switches. It can be specified with three parameters: the number of switches (N), the number of ports for each switch (r), and the number of ports in each switch that connect to local computing elements (p). We use $jfish(N, r, p)$ to refer to an instance of random regular topology with the three parameters: a random regular network with N switches; each switch has r ports; among the ports p are connected to local computing elements and $r-p$ are randomly connected to other switches. Five random regular topology instances with different sizes, $jfish(64, 6, 1)$, $jfish(125, 6, 1)$, $jfish(216, 6, 1)$, $jfish(343, 6, 1)$ and $jfish(512, 6, 1)$ are used in the study. The second topology is the torus topology. We use $torus(dim, a[0], a[1], ..., a[dim-1], p)$ to denote an instance of torus topology with dim dimensions, $a[0], ...a[dim-1]$ as the size of each dimension and p processing nodes connected to each switch. Five 3-dimensional torus topology instances $torus(3, 4, 4, 4, 1)$, $torus(3, 5, 5, 5, 1)$, $torus(3, 6, 6, 6, 1)$, $torus(3, 7, 7, 7, 1)$ and $torus(3, 8, 8, 8, 1)$ are used.

The routing considered is the k-shortest path routing. When $k = 1$, the routing is the (single path) shortest path routing scheme. The k paths are calculated by first computing a set of paths that are within a given hop count limit from a *source* to a *destination* node, and then selecting the shortest k from the set. In the study, $k = 1, 2, 4, 8$.

Two types of traffic patterns are used in this paper. The first is a random communication pattern with each processing node being a source communicating to X receivers. The notation $random(X)$ is used to denote such a pattern. The second is random permutation pattern. For each random permutation pattern, each processing node is sending to and receiving from at most one other processing node. Communication patterns that consist of multiple permutation patterns are also considered. We use the notation $perm(X)$ to denote the communication pattern with X random permutation patterns. Note that conventional permutation is denoted as $perm(1)$.

Throughput results are reported as average throughput per flow, that is, aggregate throughput divided by the number of flows. The throughput is normalized to the link bandwidth. For example, an aggregate throughput of 0.8 means that the average throughput for each flow is 80% of the link speed. For each data point, the average and the 95% confidence interval of 36 random traffic patterns, are reported. Since MMF is the most commonly accepted performance model and since all other models approximate MMF in different ways, we will use MMF as the benchmark to evaluate other models in the discussion.

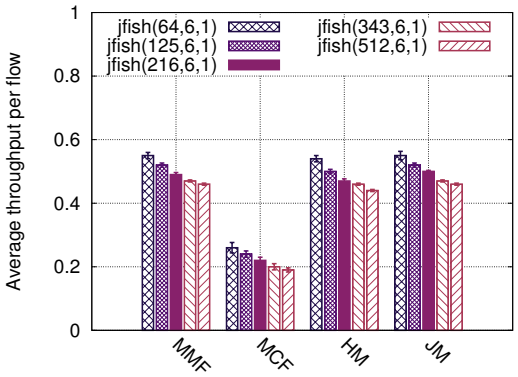

Fig. 3: Performance trend of the Jellyfish topology as the network size increases, shortest path routing, random(1) pattern

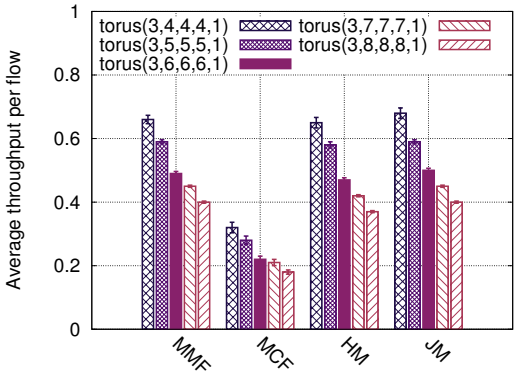

Fig. 4: Performance trend of the torus topology as the network size increases, shortest path routing, perm(1) pattern

A. The good: network and traffic scalability study

As discussed earlier, one important use of the models is to probe the performance trend. We find that all models produce similar performance trends for network scalability study regardless of traffic patterns although the absolute throughput results may not be the same. Figure 3 shows the performance trend when the network size of the Jellyfish topology increases with the random(1) traffic pattern. The routing is single path routing in this experiment. As can be seen from the figure, by increasing the network size, all models observe a decrease in average throughput. This is expected as the number of inter-switch links per switch stays the same. A similar trend is shown in Figure 4 for torus with permutation patterns. This trend occurs in all of our studies with different combinations of topologies, routing schemes, and traffic patterns, which demonstrates that the models yield a similar trend for the network scalability study.

Besides network scalability, the models also yield similar performance trends for traffic scalability study when more traffic flows of the same characteristics are added to the system. Figure 5 shows the performance trend when more permutation traffic is added to $jfish(216, 6, 1)$ with shortest path routing. As can be seen from the figure, all models

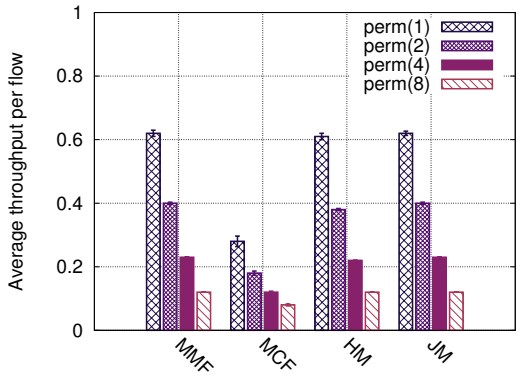

Fig. 5: Performance trend of $jfish(216, 6, 1)$ as the communication pattern becomes more dense, shortest path routing

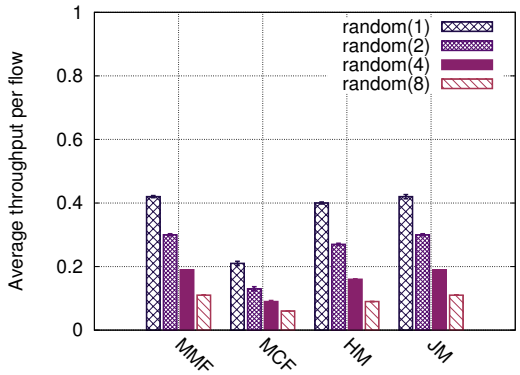

Fig. 6: Performance trend of $torus(3, 6, 6, 6, 1)$ as the communication pattern becomes more dense, shortest path routing

yield a similar performance trend as more permutation is added. Figure 6 shows the performance trend when more random traffic are added to $torus(3, 6, 6, 6, 1)$ with shortest path routing, which is the same as that in Figure 5. This trend occurs in all of our studies with different combinations of topologies, routing schemes, and traffic patterns.

In all of the experiments, the absolute throughput values with JM closely track the values with MMF. This is also observed in all of the experiments reported in this paper as well as other experiments not reported. This strongly indicates that JM is a very good approximation model for MMF in many network settings. Given its low complexity, JM is an excellent approximation of MMF for large scale interconnects.

B. The bad: impact of routing and traffic pattern

Although all of the models yield similar trends in the network and traffic scalability studies, different models can point to different trends when they are used to study the impact of routing and traffic patterns. Figures 7 and 8 illustrate an example. The figures show the modeling results for k-shortest path routing with different values of k on $jfish(216, 6, 1)$ for the two traffic patterns, random and permutation, respectively.

Let us first compare the trend of MMF to that of MCF in these two figures. With MMF, the two figures show that for both random and permutation patterns, the throughput performance increases along with the increase of the value k in k-shortest path routing. For MCF, the trends depend on the traffic pattern. For the random pattern in Figure 7, the MCF throughput is much smaller than MMF throughput. In addition, the k-shortest path routing with different values of k has almost no impact on the MCF throughput. Hence, for the random pattern, MMF will conclude that increasing the number of paths in multi-path routing will increase the throughput while MCF will conclude that the number of paths in multi-path routing will have no effect on the throughput performance.

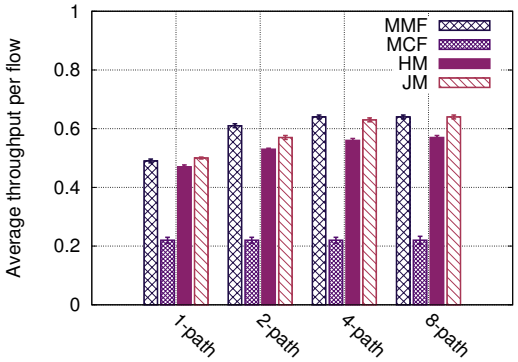

Fig. 7: Performance of $jfish(216, 6, 1)$ with multi-path routing, random(1) pattern

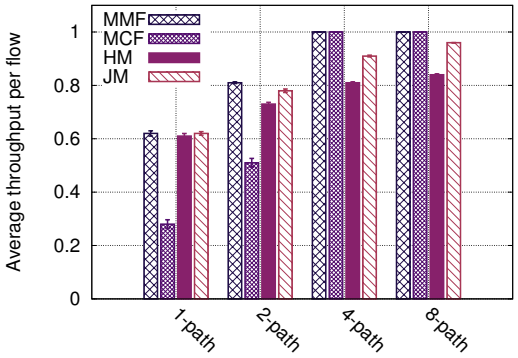

Fig. 8: Performance of $jfish(216, 6, 1)$ with multi-path routing, perm(1) pattern

Now, consider the permutation pattern in Figure 8. For this pattern, the MCF throughput is less than half of the MMF throughput when $k = 1$. When $k = 8$, the MCF throughput is the same as the MMF throughput. Hence, for the permutation pattern, MCF will conclude that the value of k has twice as much impact on the throughput as what MMF would conclude. This is opposite to the conclusion for the random pattern. Thus, using MCF and MMF will come to a different conclusion regarding the impact of value k in k-path routing: MCF can either under-estimate the impact or over-estimate the impact depending on the traffic pattern selected.

As shown in this experiment, the conclusions drawn with MCF are sensitive to the traffic pattern. In particular, we have found that MCF throughput is particularly sensitive to traffic patterns when an end node becomes the bottleneck. In such cases, the MCF throughput will remain the same regardless of all other parameters such as the routing scheme as in the case in Figure 7. Thus, when using the MCF model to probe interconnect parameters, the traffic patterns used in the study must be very carefully selected.

Let us now compare the trend of MMF to that of HM in these two figures. We see that for both patterns, the trend for HM is somewhat similar to that for MMF. However, as can be seen from both figures, HM tracks MMF very well when $k = 1$ while significantly under-estimating the MMF throughput when $k = 8$. This trend is observed in other experiments with different topologies and traffic patterns. We conclude that HM systematically under-estimates MMF throughput when k is large. Hence, HM will always under-estimate the impact of k for k-shortest path routing in comparison to MMF.

Figures 9 and 10 show the results on a torus topology $torus(3, 6, 6, 6, 1)$. MCF under-estimates the impact of k for the random pattern while over-estimating the impact for the permutation pattern in comparison to MMF. HM always under-estimates the impact of k in comparison to MMF. JM tracks the MMF trend well although it slightly under-estimates the throughput of MMF.

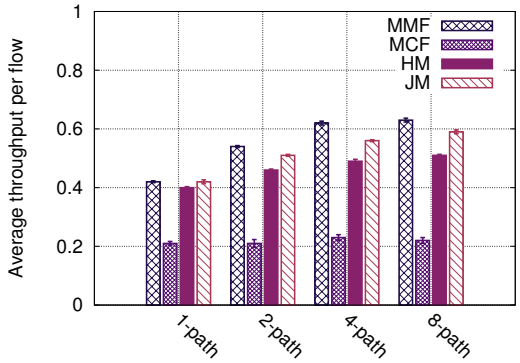

Fig. 9: Performance of $torus(3, 6, 6, 6, 1)$ with multi-path routing, random(1) pattern

C. The ugly: evaluating different interconnect designs

One key application of a throughput model is to compare the relative performance among different interconnect designs. It is clear that the absolute throughput performance with different models will not be the same. However, since all models are capturing the throughput performance, ideally the relative performance with different models should be consistent. Unfortunately, models with different approximations of MMF are affected by different parameters such as traffic patterns and routing schemes as shown in the previous sub-section. As we will show later, different models can yield contradictory results when they are used to evaluate different interconnect designs.

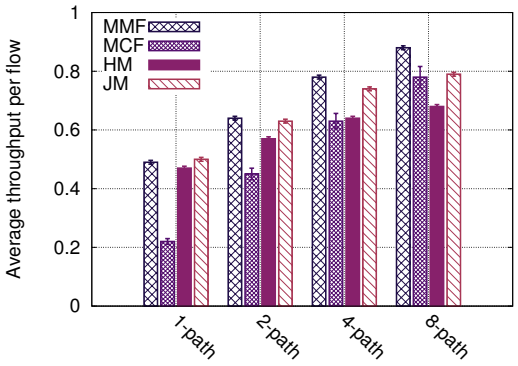

Fig. 10: Performance of $torus(3, 6, 6, 6, 1)$ with multi-path routing, perm(1) pattern

Figure 11 shows the relative performance of two Jellyfish topologies with MMF, MCF, HM, and JM. The two interconnects compared are $jfish(50, 10, 5)$ with 4-path routing and $jfish(216, 6, 1)$ with 1-path routing. As can be seen from the figure, using random(1) patterns in the evaluation, with MMF, $jfish(50, 10, 5)$ has 22% higher throughput than $jfish(216, 6, 1)$; with HM, $jfish(50, 10, 5)$ is 9% worse than $jfish(216, 6, 1)$; with MCF, the two networks have almost the same throughput. Only JM has the similar relative performance as MMF for the two networks: with JM, $jfish(50, 10, 5)$ has 10% higher throughput than $jfish(216, 6, 1)$. The relative performance of the two networks reported by MMF, MCF, and HM is completely different.

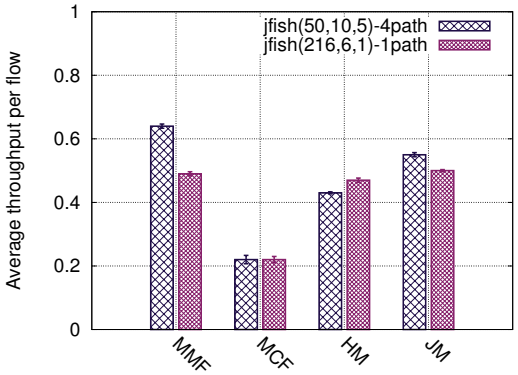

Fig. 11: Relative performance of random(1) pattern on $jfish(50, 10, 5)$ with 4-path routing and $jfish(216, 6, 1)$ with single-path routing

Figure 12 shows another example, the relative performance of $jfish(216, 6, 1)$ with 2-path routing and $tours(3, 6, 6, 6, 1)$ with 4-path routing. As can be seen from the figure, using perm(1) traffic pattern in the evaluation, with MMF, the two networks have very similar performance ($jfish(216, 6, 1)$ is slightly better than $torus(3, 6, 6, 6, 1)$); with MCF, $torus(3, 6, 6, 6, 1)$ offers 20% higher throughput than $jfish(216, 6, 1)$; with TM, the $jfish(216, 6, 1)$ offers 11% higher throughput; with JM, $jfish(216, 6, 1)$ is slightly

better than $torus(3, 6, 6, 6, 1)$. JM has the same trend as MMF with a slightly larger performance gap. Similar to the case in Figure 11, MMF, MCF, and HM also report completely different relative performance in this experiment.

These results indicate that the throughput performance of an interconnect is sensitive to the models: using different models may yield different conclusions. Note that all of the results reported are the average of 36 random traffic patterns, which indicates that the differences in the models are not random. Hence, users must carefully select proper models in the interconnect evaluation, which will require an understanding of the subtle differences among the models.

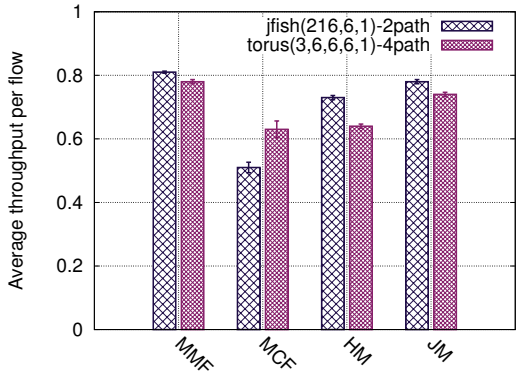

Fig. 12: Relative performance of perm(1) pattern on $jfish(216, 6, 1)$ with 2-path routing and $torus(3, 6, 6, 6, 1)$ with 4-path routing, perm(1) pattern

D. Summary

We examine throughput modeling results with MMF, MCF, HM, and JM, and identify the cases when the models give similar performance indications, when different models result in different performance trends, and the different models yield contradictory results. The main conclusions of the study are summarized as follows.

- Although MCF is the first order approximation of MMF, it often yields significantly lower throughput value than MMF. Additionally, MCF modeling results are very sensitive to the traffic patterns. In particular, for communication patterns that create bottleneck on processing nodes, MCF modeling results give little indication of the impact of other parameters such as the routing scheme.
- HM in general achieves good approximation of MMF for single path routing. However, it may significantly underestimate the throughput for k-path routing when k is a large value.
- JM has consistently showed good approximation of MMF for all of the experiments in this study. This indicates that JM is a good approximation of MMF for many scenarios.
- MCF and HM have systemic biases towards some interconnect and traffic parameters in comparison to MMF. Using MMF, MCF, and HM to evaluate different interconnect designs may result in completely different conclusions.

IV. Related Work

Several interconnection network performance models have been proposed over years. Here we focus on widely accepted models in literature. Max-Min Fairness (MMF) [1] has been the ideal model for many resource management algorithms. Algorithms for computing MMF rates for different networks with different routing schemes have been developed [2], [3]. However, computing MMF rate allocation for general networks is too computationally intensive to be applied to large scale networks. To overcome this limitation, several MMF approximations have been used. Among those, Maximum Concurrent Flow [4] is the first order approximation of MMF. Finding MCF rate only requires solving the rate allocation problem once, instead of iteratively, as in solving the MMF rate allocation problem. For this reason, MCF can be applied to larger networks than MMF and has been used in several recent studies [5]–[7]. Although calculating MCF rates are faster than corresponding MMF rates, it still requires solving a linear programming problem which is slow and computationally prohibitive for larger interconnection networks. In order to avoid the computational complexity of using linear programming formulations, Hoefler's method (HM) and Jain's method (JM) were proposed. HM [10] defines the throughput of each link in the network as its respective capacity divided uniformly among all flows using the link under a specific communication pattern. JM [11] extends HM by iteratively adding the remaining bandwidth on each link to the active flows. In this work, we compare these common throughput models and investigate whether different models can yield similar conclusions when they are used to evaluate interconnect designs.

V. Conclusion

We compared four common throughput models for interconnection networks, max-min fairness (MMF), maximum concurrent flow (MCF), Hoefler's method (HM) and Jain's method (JM). We show that although these models are either MMF or approximate MMF, they have subtle differences and may produce contradictory results when used to evaluate different network designs. While it is pre-mature to conclude that one model is better or more accurate than others, our results demonstrate that there are pitfalls in using an arbitrary model for interconnect performance evaluation and that it is important to select the proper model to study certain interconnect features.

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Numbers 0000219853 and DE-SC0016039.

References

[1] Dimitri P. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 2nd edition, January 1992.

[2] Dritan Nace, Linh Nhat Doan, Olivier Klopfenstein, and Alfred Bashllari. Max-min fairness in multi-commodity flows. *Computers and Operations Research*, 35(2):557–573, February 2008.

[3] M. A. Mollah, X. Yuan, S. Pakin, and M. Lang. Fast calculation of max-min fair rates for multi-commodity flows in fat-tree networks. In *2015 IEEE International Conference on Cluster Computing*, pages 351–360, Sept 2015.

[4] Farhad Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, April 1990.

[5] S. A. Jyothi, A. Singla, P. B. Godfrey, and A. Kolla. Measuring and Understanding Throughput of Network Topologies. Nov. 2016. Accepted at The International Conference for High Performance Computing, Networking, Storage and Analysis (SC'16).

[6] Ankit Singla, P. Brighten Godfrey, and Alexandra Kolla. High throughput data center topology design. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, April 2014.

[7] P. Faizian, M. A. Mollah, X. Yuan, S. Pakin, and M. Lang. Random regular graph and generalized De Bruijn graph with k-shortest path routing. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 103–112, May 2016.

[8] T. Hoefler, T. Schneider, and A. Lumsdaine. Optimized Routing for Large-Scale InfiniBand Networks. In *17th Annual IEEE Symposium on High Performance Interconnects (HOTI 2009)*, Aug. 2009.

[9] Xin Yuan, Santosh Mahapatra, Wickus Nienaber, Scott Pakin, and Michael Lang. A New Routing Scheme for Jellyfish and its Performance with HPC Workloads. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 36. ACM, 2013.

[10] T. Hoefler, T. Schneider, and A. Lumsdaine. Multistage switches are not crossbars: Effects of static routing in high-performance networks. In *2008 IEEE International Conference on Cluster Computing*, pages 116–125, Sept 2008.

[11] Nikhil Jain, Abhinav Bhatele, Xiang Ni, Nicholas J. Wright, and Laxmikant V. Kale. Maximizing throughput on a dragonfly network. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 336–347, Piscataway, NJ, USA, 2014. IEEE Press.

[12] Ankit Singla, Chi-Yao Hong, Lucian Popa, and Philip Brighten Godfrey. Jellyfish: Networking Data Centers Randomly. In *NSDI*, volume 12, pages 17–17, 2012.