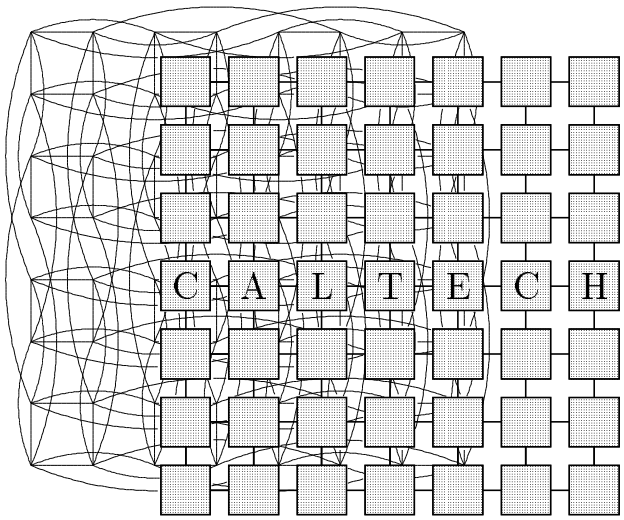

SUBMICRON SYSTEMS ARCHITECTURE PROJECT
Department of Computer Science
California Institute of Technology
Pasadena, CA 91125

A Critique of Adaptive Routing

by

Michael J. Pertel

Caltech Computer Science Technical Report

Caltech-CS-TR-92-06

June 18, 1992

The research described in this report was sponsored by
the Defense Advanced Research Projects Agency.

A Critique of Adaptive Routing

Michael J. Pertel

June 18, 1992

Abstract

This report refutes claims that adaptive routing performs better than dimension-order routing. Simulation results are presented that show dimension-order routing achieves both higher throughput and lower latency than adaptive routing. Specious claims for the advantages of adaptive routing are critiqued.

1 Introduction

Adaptive routing [1] is an alluring idea. Although there are usually many minimum-distance paths between two nodes in a multicomputer message-passing network, the dimension-order routing used in existing networks [2] always routes a packet from a given source to a given destination along the same path. Adaptive routing allows a packet to follow any minimal path from source to destination, and would seem to offer an opportunity to decrease latency, diffuse local areas of congestion, increase channel utilization (throughput), and improve fault tolerance.

The most attractive feature of adaptive routing was its promise to double network throughput. Previous simulation studies [1] indicated that networks using dimension-order routing could utilize only $\approx 50\%$ of their bisection bandwidth, whereas adaptive-routing networks could utilize $\approx 90\%$ of their bisection bandwidth. Improving throughput was the primary motivation for adaptive routing because throughput is more important than latency, and claims for traffic diffusion and fault tolerance had not been demonstrated.

A VLSI implementation of adaptive routing was undertaken [3] with the belief that adaptive routing would increase throughput, diffuse hot-spots, and improve fault tolerance. In preparation for making a detailed design, an architecture [4] and simulators [5] were developed to reproduce, refine, and extend the earlier studies. The architecture was sufficiently general that it could implement either dimension-order routing (DOR) or adaptive routing (AR). The simulators were significantly faster and more flexible than simulators used in earlier studies, and a wider range of network topologies, sizes, and characteristics were simulated. In particular, it was possible to compare DOR and AR for realistic network sizes [6] while keeping everything but the routing algorithm fixed.

Throughput: It was discovered that the earlier results showing a performance advantage for AR over DOR were an artifact of giving the adaptive routers more buffering than the dimension-order routers:

> **Fallacy 1** *Dimension-order routers can utilize only* $\approx 50\%$ *of the bisection bandwidth, whereas adaptive routers can utilize* $\approx 90\%$ *of the bisection bandwidth.*
>
> **Fact 1** *Dimension-order routing allows optimal bandwidth utilization. As the network radix increases, the bisection utilization of DOR approaches 100%. When given equal buffering, DOR can support* **higher** *throughput than AR.*

Latency: The intuitive notion that AR can reduce latency by circumnavigating local congestion is also incorrect. Indeed, blocking due to output competition is very rare with DOR, especially for networks with large radices. With DOR, an output channel is used almost exclusively by the corresponding input channel in the same dimension, except for injections from the previous dimensions with probability $O(\frac{1}{R})$ for network radix R. Since contention is rare, AR does not perform better than DOR. Moreover, DOR performs better than AR for heavy applied loads. DOR utilizes all bisection channels equally, and achieves the maximum possible throughput. AR does not utilize all the mesh's bisection channels evenly: a packet is allowed to follow any minimal path from source to destination, and many more paths cross the middle of the bisection than the cross the edges of the bisection. AR creates a surfeit of congestion in the center of the mesh and

under-utilizes the edges; this prevents AR from achieving maximal throughput, and leads to much higher latency than DOR, for heavy traffic. In summary:

> **Fallacy 2** *Adaptive routing decreases latency by routing packets around congestion.*
>
> **Fact 2** *Adaptive routing* **increases** *latency for heavy traffic. Output contention is rare,* $O(\frac{1}{R})$*, for dimension-order routing. Dimension-order routing produces less network congestion under heavy traffic, and achieves higher throughput.*

"Hot Spots": The notion that adaptive routing would improve performance by avoiding "hot spots" is specious. Claims about hot spots appeal to intuition, but they have not been accompanied by a precise, realistic definition of "hot spot." If a hot spot is a random, local fluctuation in traffic, then the simulations of random traffic presented in this report show that DOR, not AR, gives better performance. If a hot spot is a chronic region of abnormally heavy traffic, then hot spots are pathological cases. Chronic regions of congestion can be attributed to poor program design or poor process placement, and networks are not designed to remedy the ills of a particular program. For a specific definition of "hot spot," the burden of proof lies with the AR proponent, who must show that such hot spots occur in real networks, that they degrade network performance, that they are best handled in the network itself, and that AR improves their handling significantly. One could almost certainly concoct a pathological traffic pattern that favors AR, but the design of general-purpose routing networks cannot be based upon a special-case traffic pattern; random traffic is the most general model. Communication patterns that exhibit locality cannot benefit from AR because there is negligible path multiplicity for short paths. Communication patterns that do not exhibit locality can use random process placement to avoid pathological congestion, and DOR outperforms AR for random traffic.

> **Fallacy 3** *Adaptive routing improves performance by diffusing "hot spots."*
>
> **Fact 3** *This nebulous claim has never been substantiated. Localized traffic cannot benefit from AR because it lacks*

path multiplicity. Random traffic is the accepted worst-case model for non-localized traffic, and DOR outperforms AR for random traffic.

Fault Tolerance: Fault tolerance is a popular concept, but the term is often used loosely. Since AR allows a packet to choose from more than one path through the network, "fault tolerance" is sometimes listed among the virtues of AR. When actual studies of the fault tolerance of AR have been done [1], the results for realistic topologies [6] have been poor. AR does not provide redundancy for all paths; this fact alone is sufficient to discredit claims that AR provides fault tolerance.

Note 1 *In a radix-R d-dimensional mesh, the number of minimum-distance paths between two nodes separated by* $(\Delta x_1, \ldots, \Delta x_d)$ *is* $\frac{(\Delta x_1 + \ldots + \Delta x_d)!}{(\Delta x_1)! \ldots (\Delta x_d)!}$.

Note 2 *There are* $R^d\,(d(R-1)+1)$ *node pairs that have only one minimum-distance path between them. Even when multiple paths exist, they may overlap significantly and are not independent.*

Building a fault-tolerant network requires an intentional design effort, and well-defined reliability goals. The fact that a routing algorithm incidentally yields path redundancy for some $(src, dest)$ pairs does not justify claims of fault tolerance.

Indeed, some legitimate approaches to fault tolerance [1] are easier to layer atop a network that uses DOR. With AR, any one of many paths might be taken by a packet, so delivery can be guaranteed only if **all** such paths are fault free. With DOR and static faults, a (src, dst) pair always works or always fails.

Fallacy 4 *Adaptive routing provides fault tolerance.*

Fact 4 *Adaptive routing does not provide redundancy for all paths, and it does not provide practical fault tolerance. The possibility that a packet might follow any one of many*

[1] A faulty (src, dst) path can be rerouted through an intermediate node: $(src, i), (i, dst)$. A fixed intermediate can be chosen for each broken path when static faults are recorded. Such an approach requires no special hardware and can be used in existing machines.

different paths makes it harder to guarantee that a packet will be delivered in a faulty network using AR.

2 Simulation Results

This section presents measurements of network latency obtained from a simulator [5]. The simulation parameters are (d, R, L, A), where d is the mesh dimension, R is the mesh radix, L is the packet length in *flits*,[2] and A is the applied load. The number of nodes is $N = R^d$, and the bisection bandwidth is $B = R^{d-1} \frac{flits}{cycle} = \frac{NW}{R} \frac{bits}{cycle}$. The simulator uses random, homogeneous traffic: on every cycle each node generates a packet with probability $q = \frac{4A}{RL}$, and all (src, dst) pairs are equiprobable.

$$q = \frac{4\,A}{R\,L} \frac{packets}{cycle} = \frac{4\,A}{R} \frac{flits}{cycle} = \frac{4\,A\,W}{R} \frac{bits}{cycle}$$

The applied load (A) is expressed as a fraction of the network's bisection bandwidth (B); in steady state, the bisection utilization (U) is equal to the applied load (A), and the throughput is $\mathcal{T} = 4BU$.[3]

The steady-state average cut-through latency T is measured for a range of applied loads: A = 10%, 30%, 50%, 70%. Measurements are made for several mesh radices R and dimensions d. The measured latency T is the average time between the sending of a packet from the source and its arrival at the destination, including *injection* and *cut-through* latency but not *spooling* latency. Cut-through latency is the head-to-head transmission delay. Since the injection queue is just another input FIFO, injection latency and cut-through latency are not separated. Spooling latency refers to the L-cycle head-to-tail delay.

The basic simulator has been described in a previous report [5], so only minimal discussion is included here. Results are reported for two simulator variants: SNS (which uses DOR) and ANS (which uses AR). The versions of SNS and ANS used to produce the data for this report differ from those previously reported [5] in some minor respects. The `main()` function has been changed to wait for the throughput to converge before waiting for the latency to converge. Also, a measurement of the average queue length (AQLEN) has been added. A listing of the

[2] One flit is W bits, where W is the channel width.

[3] For random traffic, $\frac{1}{2}$ of all packets cross the bisection, $\frac{1}{4}$ in each direction.

SNS code used for this report is included as an appendix. ANS differs from SNS only in the following function:

```
int allowed(n,in,out) int n,in,out; {
        packet *p=node[n].head[in]; int pc,nc,dim=DIMOF(out);
        if(!p || p->tin>curtime) return 0;        /* p arrived? */
        if(node[n].tnh[in]>curtime) return 0;     /* p at head? */
        if(node[n].tfree[out]>curtime) return 0;  /* out free?  */
        if(p->dest == n) return out==0;           /* p at dest? */
        pc=COORD(p->dest,dim), nc=COORD(n,dim);
        if(nc<pc) return out==SUCC(dim);
        if(nc>pc) return out==PRED(dim);
        return 0;                                 /* not profitable */
}
```

Except where otherwise noted, the simulations use packet length `L=32` *flits*, which is realistic for fine-grained computations [7, 8]. For comparison, `L=8` and `L=128` results are presented in an appendix.

The simulations were run using `ACCURACY=.03`, which is more than sufficient to show that DOR is preferable to AR. The execution times for the simulations would increase by an order of magnitude if `ACCURACY=.01` were used instead.

The simulator measures the *network-average* queue-length at termination, not the *time-average* queue-length: `AQLEN` is the number of **packets** that have been sent but not received divided by the number of queues. The `AQLEN` values are listed to provide insight, but the simulator does not check their convergence. Inspecting the `AQLEN` values shows that the average queue length is typically only a fraction of a packet. It can also be seen that the average queue length is greater with AR than with DOR for heavy traffic. **The `AQLEN` values are coarse and may appear noisy.**

2.1 One-Dimensional Results

There is no difference between AR and DOR for 1D because there is only one path between any two nodes. ANS and SNS produce identical results for `d=1`. Although 1D results cannot be used to compare AR and DOR, they are included for completeness; 1D is the simplest case and will be studied before proceeding to 2D and 3D. Some 1D simulation results are shown in Table 1.

> **Observation 1** *Average cut-through latency is not directly proportional to network radix, so it is not directly proportional to average distance.*

	A=.1		A=.3		A=.5		A=.7	
R	T	AQLEN	T	AQLEN	T	AQLEN	T	AQLEN
4	3.95	0.00	10.8	0.00	34.8	0.46	335	3.18
8	5.47	0.00	11.5	0.04	26.4	0.13	73.6	0.65
16	8.37	0.00	14.9	0.00	29.7	0.06	71.5	0.11
32	13.8	0.01	20.6	0.00	36.1	0.04	72.2	0.04
64	24.7	0.00	31.9	0.00	45.8	0.02	80.6	0.01
128	46.1	0.00	53.7	0.00	67.0	0.00	103	0.02

Table 1: 1D Data

Observation 2 *For a given applied load there is an "optimal" radix for which the cut-through latency is minimal. The optimal radix increases with throughput.*

When a packet traverses an empty network, its cut-through latency is $T_{cut} = D + 1$: there is a one-cycle delay for each hop along a path of distance D, and a one-cycle delay at the destination. The average distance in a radix-R d-dimensional mesh is $\overline{D} = \frac{d}{3}\left(R - \frac{1}{R}\right)$ [10]. Congestion in the network increases the cut-through latency. Even for light traffic (A= 10%) congestion cannot be neglected; the congestion-free latency formula is correct only when the applied load is extremely small (a few percent of network capacity). The congestion-free latency formula is not even qualitatively correct: according to the formula, average cut-through latency should increase linearly with average distance, but latency is not proportional to distance. Latency grows only linearly with average distance (radix), but it grows supra-linearly with channel utilization (throughput). According to the Pollaczek-Khinchin formula, average queue delay grows with queue utilization (u) as $\frac{u}{1-u}$. As radix increases, blocking becomes rarer, so queue utilization decreases.

Note 3 *With DOR, an incoming packet continues in the same dimension unless its offset in that dimension is zero. If the offset in the current dimension has been reduced to zero, the packet is dejected into the next dimension. A packet continuing in its present dimension is blocked by output competition only if the output is being used by an*

injection from the previous dimension. The average injection rate is $q = \frac{4A}{R}\frac{flits}{cycle}$*, so the probability of blocking is* $O(\frac{1}{R})$.

As radix increases, the average distance increases but the average delay per hop decreases. For a given applied load, there is a latency-optimal radix that gives the smallest cut-through latency. The radix that gives the smallest latency increases with applied load as queueing delay becomes more important.

To give a better sampling of the latency versus applied load curve, more data is presented in Table 2.

R	A=.55	A=.6	A=.65	A=.75	A=.8	A=.85	A=.9	A=.95
4	47.1	73.0	123	∞	∞	∞	∞	∞
8	33.2	44.1	55.4	112	193	824	∞	∞
16	35.0	44.1	52.9	96.3	140	205	1233	∞
32	40.3	48.0	57.4	91.5	126	180	353	∞
64	51.7	58.5	68.6	97.4	125	182	284	833
128	72.4	81.3	89.9	121	144	190	295	611

Table 2: More 1D Latencies

Observation 3 *Latency is a smooth, super-linear, monotonically increasing function of throughput (*A*) that diverges for applied loads above a radix-dependent maximum bisection utilization* U_{max}.

Observation 4 U_{max} *increases with radix.*

The existence of a maximum bisection utilization less than 100% can be readily deduced for random traffic. Consider the bisection channel and the node that feeds that channel. The channel can be used either by a new injection at that node or by a packet continuing from the node's predecessor. When the bisection channel is not used for either injection or forwarding, it is idle; thus, the bisection utilization is limited to less than 100%. The probability that the node injects a packet that uses the bisection is $\frac{q}{2}$, and $A < 1 \Rightarrow q < \frac{4}{R}$. [4] The probability that a packet continues from the predecessor across the bisection

[4] Unless $R > 4$, this bound is vacuous.

channel is $\frac{(R/2)}{(R/2)+1}\,u$, where u is the utilization of the channel from the predecessor. Thus, the probability that the bisection channel is idle is $P_{idle} > \left(1 - \frac{2}{R}\right)\left(1 - \frac{1}{1+(2/R)}\right)$ and $U_b = 1 - P_{idle}$ is bounded by

$$U_b < 1 - \frac{2}{R}\frac{(R-2)}{(R+2)}$$

2.2 Two-Dimensional Results

Although data comparing AR to DOR for 1D and 3D meshes is also presented, the comparison for 2D meshes is the most realistic, practical, and interesting. This research is part of a large multicomputer design project called the Mosaic [6]. The Mosaic is a 128×128 mesh of single-chip multicomputer nodes; each node contains an 11 MIPS processor, 64 KB memory, and asynchronous DOR network router with 60 MB/s channels. Since the 2D mesh is the topology used in state-of-the-art machines [6, 11, 12], it is the most appropriate topology for comparing AR to DOR. [5]

The 2D simulation results are shown in Table 3. `T` and `AQLEN` are listed as "∞" when the latency and queue lengths did not converge, i.e., when they grew without bound.

Observation 5 *DOR can support higher throughput than AR. U_{max} is a decreasing function of dimension for AR.*

Observation 6 *DOR yields much lower latency than AR for heavy traffic. AR yields slightly lower latency for light traffic.*

Observation 7 *The performance advantage of DOR increases with radix. DOR begins beating AR at lower applied loads for larger radices.* **DOR beats AR at any applied load for a 128×128 mesh.**[6]

Observation 8 *Average queue length is typically only a fraction of a packet, and queue lengths decrease as radix increases.*

Observation 9 *Differences in latency between DOR and AR reflect differences in average queue length.*

[5] For planar wiring, 2D networks are throughput optimal [13].

[6] In an empty network, $T_{cut} = D + 1 \Rightarrow \overline{T} = \overline{D} + 1$, for both AR and DOR.

		DOR (SNS)		**AR** (ANS)	
R	A	T	AQLEN	T	AQLEN
4	.1	6.31	0.00	6.17	0.00
4	.3	18.5	0.06	17.5	0.03
4	.5	54.7	0.44	52.9	0.43
4	.6	108	0.33	127	0.50
4	.65	184	0.44	∞	∞
4	.7	382	2.60	∞	∞
8	.1	9.79	0.00	9.09	0.00
8	.3	21.6	0.02	18.8	0.02
8	.5	53.3	0.13	44.1	0.10
8	.7	179	0.44	256	0.92
8	.75	246	0.85	∞	∞
8	.8	688	2.00	∞	∞
16	.1	15.8	0.00	14.9	0.00
16	.3	29.6	0.02	25.9	0.01
16	.5	62.9	0.05	52.7	0.05
16	.7	156	0.16	261	0.29
16	.75	223	0.21	∞	∞
16	.8	354	0.45	∞	∞
32	.1	26.9	0.00	26.2	0.00
32	.3	41.6	0.01	39.2	0.01
32	.5	74.5	0.03	70.7	0.03
32	.6	102	0.05	105	0.05
32	.7	159	0.08	256	0.15
32	.8	294	0.20	∞	∞
64	.1	48.6	0.00	48.2	0.00
64	.3	63.9	0.01	63.4	0.01
64	.5	95.5	0.02	100	0.02
64	.7	182	0.05	253	0.07
64	.8	296	0.09	∞	∞
128	.1	90.1	0.00	89.7	0.00
128	.3	107	0.01	110	0.01
128	.5	140	0.01	152	0.01
128	.7	222	0.03	293	0.04
128	.75	265	0.04	> 461	$> .07$
128	.8	332	0.05	> 2440	$> .70$

Table 3: 2D Data

Even when AR latency is less than DOR latency, the difference is far too small to justify the greater cost (e.g., area and complexity) of AR. Moreover, it is precisely when network performance matters most (i.e., at high applied loads) that DOR performs significantly better than AR. The extreme difference in latency between AR and DOR for heavy applied loads can be attributed to the difference in U_{max}.

The approximate formula for U_{max} that was derived for 1D is still applicable for 2D with DOR, since 2D DOR can be regarded as two independent 1D routings. The U_{max} bound is a property of random traffic, not a property of the routing algorithm. DOR can support maximal throughput.

AR does not support maximal throughput. AR has a lower U_{max} than DOR. The lower throughput of AR is a property of the routing algorithm. DOR achieves maximal throughput because every channel of the bisection is utilized maximally. AR cannot support maximal throughput because it does not utilize all bisection channels evenly. With AR, channels at the center of the bisection are utilized more than channels at the edges of the bisection because more of a packet's possible paths pass through the center of the mesh. If the utilizations of the bisection channels are independently monitored, then U_{max} is seen to be the DOR value multiplied by the ratio of the mean to the peak of the AR bisection-utilization profile [14]. Note that, even if an AR implementation could be devised that performs as well as DOR, AR cannot beat DOR.

For a given number of nodes and a given bisection bandwidth, the highest throughput is achieved by the network of the largest radix (smallest dimension) because U_{max} increases with `R`. If bisection bandwidth were a realistic measure of network cost, this would imply that 1D networks were optimal. However, in reality, a 1D network is much more expensive than a 2D network with the same bisection bandwidth. Wiring area is a much more realistic cost metric than bisection bandwidth, and for a given wiring area, the highest bisection bandwidth is achieved by using a 2D network [13]. Therefore, large-`N` machines (e.g., `N=16384`) have large radices (e.g., `R=128`), and for large radices DOR outperforms AR *a fortiori*.

As noted earlier, the simulators used for this report give only a crude measure of the average queue length. `AQLEN` is the network-average queue length when the simulator terminated. A proper mea-

surement of the average queue length would be the time average of the network average. If one were interested in measuring the average queue length properly, one could modify the simulator to record `AQLEN` every cycle; the time average of `AQLEN` could be monitored for convergence the way `T` is monitored. Given that `AQLEN` is a crude measurement, not much should be deduced from the tabulated values. However, it seems safe to conclude from the above data that fairly short FIFOs are sufficient for real router implementations. Existing routers use FIFO lengths that are only a fraction of the average packet length,[7] and the simulation results suggest that little performance improvement would result from using longer FIFOs. [8]

2.3 Three-Dimensional Results

Using higher-dimensional meshes should favor AR. As the number of dimensions increases, so does the path multiplicity. With AR, a packet is allowed to reduce its offset in any of the d dimensions at any time, so it may have up to d allowed outputs. With DOR, there is always exactly 1 allowed output. Thus, the "difference" between DOR and AR is proportional to d. For $d = 1$, DOR and AR are identical. The performances of AR and DOR are expected to differ more for 3D than they did for 2D. The 3D simulation results are shown in Table 4.

Observation 10 *For fixed radix, latency is not directly proportional to dimension.* [9]

Observation 11 *For a fixed number of nodes, lower dimension and larger radix gives better performance for heavy traffic.*

Comparing the 3D data to the 2D data shows that the difference between DOR and AR performance increases with dimension. The slight latency advantage of AR for light traffic is greater for 3D networks. The lower throughput and higher latency of AR for heavy traffic is more pronounced for 3D networks. Path multiplicity is both the virtue and the vice of AR, and it increases with dimension.

[7] A packet can be spread over multiple nodes when it cannot fit into a single FIFO.

[8] With the FIFO length equal to the packet length, the performance is the same as for infinite buffering for a 128×128 mesh with `A<80%` [5].

[9] If `T` were proportional to average distance, it would be proportional to `d`.

		DOR (SNS)		**AR** (ANS)	
R	A	T	AQLEN	T	AQLEN
4	.1	8.90	0.01	7.92	0.01
4	.3	25.2	0.04	20.4	0.03
4	.5	74.7	0.20	65.8	0.21
4	.6	149	0.56	∞	∞
4	.7	467	1.74	∞	∞
8	.1	14.1	0.00	12.0	0.00
8	.3	32.8	0.03	22.9	0.02
8	.5	79.1	0.10	55.8	0.07
8	.7	265	0.45	∞	∞
16	.1	23.2	0.00	20.6	0.00
16	.3	44.4	0.02	34.0	0.01
16	.5	93.2	0.05	77.5	0.05
16	.7	247	0.20	∞	∞
32	.1	39.8	0.00	37.3	0.00
32	.3	62.8	0.01	54.3	0.01
32	.5	112	0.03	113	0.03
32	.7	250	0.10	∞	∞

Table 4: 3D Data

Compare the results for a $4 \times 4 \times 4$ mesh to the results for an 8×8 mesh, and compare the results for a $16 \times 16 \times 16$ mesh to the results for a 64×64 mesh. These are the cases in which a 2D mesh and a 3D mesh have the same number of nodes. The average distance is smaller for the 3D meshes than for the 2D meshes: 3.75 versus 5.25 and 15.9 versus 42.7. The smaller average distance leads to lower latency for light traffic. However, the 2D meshes beat the 3D meshes for large applied load. Output contention is proportional to dimension and inversely proportional to radix, so 3D meshes have longer queue lengths. Since latency grows only linearly with average distance but superlinearly with queue utilization, a low-dimensional mesh will always beat a high-dimensional mesh for sufficiently heavy traffic.

The 2D meshes exhibit lower injection and cut-through latency than the 3D meshes without reference to spooling latency. It is well known that, for fixed wire bisection, a low-dimensional mesh can have lower latency than a high-dimensional mesh [2, 8]; however, this is a consequence of the low-dimensional mesh having wider channels, thus less spooling latency. It was thought that the cut-through latency increased with radix (decreased with dimension) while the spooling latency decreased with radix (increased with dimension). The competition of these two effects formed the basis for computing the latency-optimal dimension [2, 8]. Since cut-through latency does not decrease as dimension increases, there is no basis for such an optimization.

2.3.1 Comparing Networks of Different Dimension

The equal-`L` comparison between 2D meshes (8×8, 64×64) and 3D meshes ($4 \times 4 \times 4$, $16 \times 16 \times 16$) was biased in favor of the 3D meshes. Note that the 2D meshes showed better heavy-traffic performance despite being handicapped in the comparison. However, it is customary to compare networks of equal wire bisection, since wire bisection gives a measure of the network's cost/complexity. [10] If two meshes have the same bisection, then equal `A` values lead to equal throughput (message volume). An $R \times R$ mesh with channels of width W has bisection $B = R\,W$. An $R^{2/3} \times R^{2/3} \times R^{2/3}$ mesh must have channel width $\frac{W}{R^{1/3}}$

[10] If 2D and 3D networks are compared using equal layout area, rather than equal bisection bandwidth, then the 2D networks look even better. For a fixed layout area, a 2D network can be given more bisection bandwidth than a 3D network [13].

to have bisection RW. Equal bisection leads to equal injection rate:

$$q_{3D} = \frac{4\,\mathtt{A}\,\left(\frac{W}{\mathtt{R}^{1/3}}\right)}{\mathtt{R}^{2/3}}\frac{bits}{cycle} = \frac{4\,\mathtt{A}\,W}{\mathtt{R}}\frac{bits}{cycle} = q_{2D}$$

Since the channels are narrower in the 3D mesh, the packets must be longer to convey the same information per packet.[11] An 8×8 mesh with packet length `L=32` should be compared to a $4 \times 4 \times 4$ mesh with packet length `L=64`. A 64×64 mesh with packet length `L=32` should be compared to a $16 \times 16 \times 16$ mesh with packet length `L=128`. When a fair comparison is made, the longer packet lengths for the 3D meshes further reduces their performance relative to the 2D meshes. Table 5 presents simulations in which $\mathtt{L_{3D}} = \mathtt{R}^{1/3}\mathtt{L_{2D}}$.

With $\mathtt{L_{3D}} = \mathtt{L_{2D}}$, the 2D meshes gave lower latency only for heavy traffic. With $\mathtt{L_{3D}} = \mathtt{R}^{1/3}\mathtt{L_{2D}}$, the 2D meshes give lower latency even for light traffic. Again, the 2D meshes are showing lower **cut-through** latency, not just lower total latency. Even if the cut-through latency were higher, the 2D meshes would have lower total latency because their wider channels reduce spooling latency. In addition to having lower latency, the 2D meshes can support higher throughput (U_{max}).

3 Summary of DOR Advantages

1. **Throughput:** DOR can support higher throughput than AR on the same network: U_{max} is greater for DOR than for AR.
2. **Latency:** DOR gives significantly lower latency than AR for heavy traffic. It is for heavy traffic (communication-limited computations) that network performance is critical. The difference in latency is small for light traffic.
3. **Simplicity:** VLSI implementation of DOR [9] is considerably simpler than implementation of AR [4].
   (a) **Area:** Dimension-order routers are smaller than adaptive routers and require only a tiny fraction ($\approx 3\%$) of the area of a single-chip multicomputer node [6].

[11] Rather than making the packets longer to compensate for narrower channels, more packets could be used to send each message. However, increasing the number of packets increases the overhead due to packet headers, which increases the applied load, which increases latency.

Parameters				**DOR** (SNS)		**AR** (ANS)	
d	R	L	A	T	AQLEN	T	AQLEN
2	8	32	.1	9.79	0.00	9.09	0.00
3	4	64	.1	13.2	0.01	11.1	0.01
2	8	32	.3	21.6	0.02	18.8	0.02
3	4	64	.3	46.2	0.05	34.8	0.04
2	8	32	.5	53.3	0.13	44.1	0.10
3	4	64	.5	144	0.25	123	0.12
2	8	32	.7	179	0.44	256	0.92
3	4	64	.7	892	1.59	∞	∞
2	64	32	.1	48.6	0.00	48.2	0.00
3	16	128	.1	41.9	0.00	31.3	0.00
2	64	32	.3	63.9	0.01	63.4	0.01
3	16	128	.3	127	0.01	83.2	0.01
2	64	32	.5	95.5	0.02	100	0.02
3	16	128	.5	316	0.05	253	0.04
2	64	32	.7	182	0.05	253	0.07
3	16	128	.7	919	0.19	∞	∞

Table 5: 2D vs 3D with Equal Bisections

(b) **Speed:** DOR implementations are smaller and simpler than AR implementations, so they are faster. The simulation results underestimate the performance advantage of DOR because they assume equal cycle times for DOR and AR.

(c) **Packet-Order:** With multipath routing (like AR), packets between a given source and destination may take different paths through the network and arrive out of order; this introduces the non-trivial problem of reconstructing packet order at the destination (and reduces performance). With single-path routing (like DOR), packet-order is always preserved.

(d) **Design:** Simpler routers are easier to design, so for a given amount of design effort, more attention can be paid to optimizing the design.

4. **Scaling:** The performance advantage of DOR over AR increases with radix. DOR becomes more advantageous as radix increases. In particular, DOR beats AR for every applied load in a 128×128 mesh.

4 Future Work

The data presented in this report shows more than the preferability of DOR to AR; the data also exposes several misconceptions about routing network behavior and performance. The data shows that average cut-through latency is neither directly proportional to nor even a monotone function of average distance. The data shows that DOR performs better than previously realized; indeed, so well that there is no reason to consider a more complicated algorithm. The misunderstandings that were incidentally exposed by this adaptive-routing research impact other aspects of network design. For example, it is clear that the "optimal" dimension for a routing network is not determined by minimizing the expression for congestion-free latency [8]. Networks of different dimensionality and equal cost do not have the same throughput, so the dimension should be chosen to maximize throughput rather than minimize latency. The congestion-free latency formula is both quantitatively and qualitatively incorrect, even for light traffic. A future report will show that 2D networks are throughput-optimal for fixed area [13].

Another future report will elucidate the mechanism responsible for the superior performance of DOR. It is easy to understand why adaptive routing does not outperform dimension-order routing. Since there is only $O(\frac{1}{R})$ contention for outputs with DOR, there is little opportunity for AR to reduce contention. However, this does not explain why DOR performs **better** than adaptive routing. The reason why adaptive routing does not perform as well as dimension-order routing is that AR does not utilize all the bisection channels evenly. Simulation results proving this explanation of AR's inferior performance will be presented in [14]. It is important to remember that, even though a different AR implementation might perform as well as DOR, AR cannot outperform DOR.

5 Acknowledgements

The research described in this report was sponsored by the Defense Advanced Research Projects Agency. Support was provided by a National Defense Science and Engineering Graduate Fellowship. I gratefully acknowledge the direction and assistance of my advisor, Dr. C.L. Seitz.

References

[1] Ngai JY: *A Framework for Adaptive Routing in Multicomputer Networks.* Caltech Computer Science Technical Report: CS-TR-89-09. 1989.

[2] Seitz CL: Chapter 5 "Multicomputers," pp. 131–200, in Hoare CAR (ed): *Developments in Concurrency and Communication.* Addison-Wesley, 1990.

[3] Pertel M, Seitz C: §4.8 "Implementing Adaptive Routing in Multicomputer Networks," p. 13, in Caltech Computer Science Technical Report: CS-TR-89-12 *Semiannual Technical Report*, 1989.

[4] Pertel MJ, Seitz CL: §4.9 "A Silicon Architecture for Adaptive Cut-Through Routing," pp. 17–19, in Caltech Computer Science Technical Report: CS-TR-90-14 *Submicron Systems Architecture Project*, 1990.

[5] Pertel MJ: *A Simple Simulator for Multicomputer Routing Networks.* Caltech Computer Science Technical Report: CS-TR-92-04. 1992.

[6] Seitz CL et al: §2.1 "The Mosaic Project," pp. 2–10, in Caltech Computer Science Technical Report: CS-TR-91-10 *Semiannual Technical Report*, 1991.

[7] Boden N: *Runtime Systems for Fine-Grain Multicomputers.* Caltech Computer Science Technical Report: CS-TR-92-10. *(Ph.D. thesis, in preparation)*

[8] Dally WJ: *A VLSI Architecture for Concurrent Data Structures.* Kluwer Academic Publishers, 1987.

[9] Flaig CM: *VLSI Mesh Routing Systems.* Caltech Computer Science Technical Report: 5241:TR:87. 1987.

[10] Pertel MJ: *Mesh Distance Formulae.* Caltech Computer Science Technical Report: CS-TR-92-05. 1992.

[11] Seitz CL et al: "The Architecture and Programming of the Ametek Series 2010 Multicomputer," in: *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications*, ACM Press, 1988.

[12] Lillevik SL: "The Touchstone 30 Gigaflop DELTA Prototype," in *Sixth Distributed Memory Computing Conference Proceedings*, pp. 671–677. IEEE Computer Society Press, 1991.

[13] Pertel MJ: *The Optimal Dimension for Multicomputer Routing Networks.* Caltech Computer Science Technical Report: CS-TR-92-09. *(in preparation)*

[14] Pertel MJ: *Multicomputer Routing Network Throughput.* Caltech Computer Science Technical Report: *(in preparation)*

A Listing of Simulator Code

```
/*      sns4.c --- Improved convergence detection.
 */
#include <stdio.h>
#include <malloc.h>
double  drand48();
#define CHECK(c,m) {if(!(c)){printf("ERROR: %s\n",m); exit(7);}}
#define PBY(p) (drand48()<(p))
#define MAX(a,b) (((a)>(b))?(a):(b))
#define ABS(x)  (((x)<0)?-(x):(x))
#define DIFF(old,new) ABS(((new)-(old))/(old))
int pwr(x,y) int x,y; {int r=1; for(;y>0;y--) r*=x; return r;}

/* Parameters
 */
#ifndef N
#define N       16384           /* number of nodes */
#define R       128             /* radix */
#define d       2               /* dimension */
#endif
#ifndef A
#define A       .5              /* applied load */
#endif
#define L       32              /* packet length in flits */

#define ACCURACY .03
#define TOL (ACCURACY/3.)
#define MINPKT (1./(TOL*TOL))
int INTERVAL = (int)(MINPKT*L/(8*A*N/R));

#define B       (N/R)           /* bisection BW in flits/cycle */
#define NIN     (2*d+1)         /* number of router inputs */
#define NOUT    (2*d+1)         /* number of router outputs */
#define NUMQ    (N*NIN-d*B)     /* number of active queues */
#define AQLEN   ((numsent-numrecd)/NUMQ)

/* Measurements
 */
double numsent=0.;
double numrecd=0.;              /* number of packets received */
double totlat=0.;               /* sum of received-packet latencies */
double tothops=0.;              /* sum of received-packet distances */

#define T (totlat/numrecd)      /* average TOTAL latency */
#define TP ((numrecd*L)/curtime)/* throughput in flits/cycle */
#define U (.25*TP/B)            /* bisection utilization */
#define D (tothops/numrecd)     /* average distance */

/* Data Structures
 */
typedef struct packet packet;
struct packet{int dest,tsent,tin,nhops; packet *next;};
typedef struct nodestate nodestate;
struct nodestate
```

```
{packet *head[NIN],*tail[NIN]; int tnh[NIN],tfree[NOUT],tls,pin,pout;};

/* Numbering Conventions:
 * Dimensions numbered from 0: x=0, y=1, z=3, etc...
 * Channels numbered: local=0, xpred=1, xsucc=2, ypred=3, ...
 * Node (x,y,z,...) numbered: ... + zR^2 + yR + x
 */
#define DIMOF(in) (((in)-1)/2)
#define PRED(dim) (2*(dim)+1)
#define SUCC(dim) (2*(dim)+2)
#define END(out)  (((out)%2)?((out)+1):((out)-1))
#define COORD(n,dim) (((n)/pwr(R,dim))%R)
#define NGHBR(n,o) (((o)%2)?(n)-pwr(R,DIMOF(o)):(n)+pwr(R,DIMOF(o)))

/* Simulator
 */
int curtime=0;                  /* current simulation time */
nodestate node[N];              /* simulator state */
initnode(n) nodestate *n; {
        int i; n->pin=n->pout=n->tls=0;
        for(i=0;i<NIN;i++)
                {n->head[i]=n->tail[i]=(packet*)0; n->tnh[i]=n->tfree[i]=0;}
}
init(){ int n; for(n=0; n<N; n++) initnode(&(node[n])); }

main() {
        int n,etime=INTERVAL; double oldT,curT;
        init(); printf("\n\n"); nice(10);
        printf("SNS4, N=%d, R=%d, d=%d, L=%d, A=%g, ACCURACY=%g\n",
                        N,R,d,L,A,ACCURACY); fflush(stdout);
        printf("Wait for throughput to converge.\n");
        do{     for(  ; curtime<etime; curtime++)
                        for(n=0; n<N; n++) simulate(n);
                printf("curtime=%d\n",curtime);
                printf("numsent=%g, numrecd=%g\n",numsent,numrecd);
                printf("AQLEN=%g/%d=%g\n",numsent-numrecd,NUMQ,AQLEN);
                printf("D=%g, d*(R-1/R)/3.=%g\n",D,d*(R-1./R)/3.);
                printf("T=%g\n",T);
                printf("U=%g, A=%g, DIFF(A,U)=%g\n",U,A,DIFF(A,U));
                printf("\n"); fflush(stdout);
                curT=T; etime*=2;
        } while(DIFF(A,U)>TOL);
        printf("Wait for latency to converge\n");
        do{     oldT=curT;
                for(  ; curtime<etime; curtime++)
                        for(n=0; n<N; n++) simulate(n);
                printf("curtime=%d\n",curtime);
                printf("numsent=%g, numrecd=%g\n",numsent,numrecd);
                printf("AQLEN=%g/%d=%g\n",numsent-numrecd,NUMQ,AQLEN);
                printf("D=%g, d*(R-1/R)/3.=%g\n",D,d*(R-1./R)/3.);
                printf("U=%g\n",U); printf("T=%g\n",T);
                printf("DIFF(oldT,T)=%g, TOL=%g\n",DIFF(oldT,T),TOL);
                printf("\n"); fflush(stdout);
                curT=T; etime*=2;
        } while(DIFF(oldT,curT)>TOL); return 0;
```

```
}

/* Node Behavior
 */
#define PIN     node[n].pin
#define POUT    node[n].pout
simulate(n) int n; {
        int in,out;
        if(PBY(4.*A/(R*L))) inject(n); findpin(n);
        for(in=0; in<NIN; in++) for(out=0; out<NOUT; out++)
                if(allowed(n,(PIN+in)%NIN,(POUT+out)%NOUT))
                {
                        forward(n,(PIN+in)%NIN,(POUT+out)%NOUT);
                        if(!in) {PIN=(PIN+1)%NIN; findpin(n);}
                        if(!out) POUT=(POUT+1)%NOUT;
                }
}

findpin(n) int n; {
        int i; packet *p; nodestate *nd = &(node[n]);
        for(i=0; i<NIN; i++)
                if((p=nd->head[PIN])&&(MAX(p->tin,nd->tnh[PIN])<=curtime))
                        return;
                else PIN=(PIN+1)%NIN;
}

inject(n) int n; {
        packet *p=(packet*)malloc((unsigned)sizeof(packet));
        node[n].tls = p->tin = p->tsent = MAX(curtime,node[n].tls+L);
        p->dest = drand48()*N; p->nhops=0; p->next=0;
        enqueue(p,n,0); numsent++;
}

int allowed(n,in,out) int n,in,out; {
        int dim,nc,pc; packet *p=node[n].head[in];
        if(!p || p->tin>curtime) return 0;        /* p arrived? */
        if(node[n].tnh[in]>curtime) return 0;     /* p at head? */
        if(node[n].tfree[out]>curtime) return 0;  /* out free?  */
        for(dim=0; dim<d; dim++) {
                nc=COORD(n,dim); pc=COORD(p->dest,dim);
                if(nc<pc) return out==SUCC(dim);
                        else if(nc>pc) return out==PRED(dim);
        }
        CHECK(p->dest==n,"profitable()"); return out==0;
}

forward(n,in,out) int n,in,out; {
        packet *dequeue(); packet *p=dequeue(n,in); p->tin=curtime+1;
        node[n].tnh[in]=node[n].tfree[out]=curtime+L;
        if(out==0) deject(p);
                else p->nhops++, enqueue(p,NGHBR(n,out),END(out));
}

deject(p) packet *p; {
        numrecd++; totlat+=p->tin-p->tsent; tothops+=p->nhops;
```

```
        free((char*)p);
}

/* Misc Functions
 */
packet *dequeue(n,in) int n,in; {
        packet *p=node[n].head[in]; node[n].head[in]=p->next;
        if(p==node[n].tail[in]) node[n].tail[in]=(packet*)0;
        p->next=(packet*)0; return p;
}

enqueue(p,n,in) packet *p; int n,in; {
        nodestate *nd=&(node[n]);
        if(nd->head[in]) nd->tail[in]->next=p; else nd->head[in]=p;
        nd->tail[in]=p; p->next=0;
}
```

A.1 Correction

Line 33 of the listing is:

```
#define NUMQ (N*NIN-d*B)   /* num. active queues */
```

but it should be:

```
#define NUMQ (N*NIN-2*d*B) /* num. active queues */
```

The simulation results were obtained with the listed code. The missing "2" leads to a small systematic error in the `AQLEN` values. `AQLEN` values are reported only for insight; the network average queue length at termination is not an adequate substitute for the time average. The small systematic error is insignificant given the rough nature of the `AQLEN` values. The true network-average queue-length can be obtained by multiplying `AQLEN` by $\mathcal{C} = \frac{2d+1-\frac{d}{R}}{2d+1-2\frac{d}{R}}$. The correction factors are tabulated below.

$R =$	4	8	16	32	64	128
$d = 1$	1.10	1.05	1.02	1.01	1.01	1.00
$d = 2$	1.13	1.06	1.03	1.01	1.01	1.00
$d = 3$	1.14	1.06	1.03	1.01	1.01	1.00

B Questions and Answers

When a draft of this report was circulated, it elicited some questions and requests for more detail. It was felt that these questions would not have arisen if the conclusions of this report did not challenge some well-entrenched misconceptions. Rather than add digressions to the body of the report, it was decided to add an appendix that specifically addressed the concerns of AR proponents. A reader with no predisposition in favor of adaptive routing should find the main body of the report sufficiently convincing, and need not be bothered with fussy details. Readers inclined to find fault with the report because of the nature of its conclusions will hopefully find satisfactory answers to most of their questions below.

Question 1 *What traffic pattern was used in the simulations, and what justification is there for that choice?*

Answer 1 *Random, homogeneous traffic was used for the simulations. During each cycle, every node injects a packet with probability* $\frac{4A}{RL}$*, and all destinations are equally likely. For further details, the reader is referred to the report describing the simulator [5]. Studies of routing networks use random traffic for at least two reasons:*

1. *Random traffic is the worst case. Performance is better for localized traffic.*
2. *Random traffic is the most general. Special-case traffic patterns are of limited interest.*

In particular, the previous reports comparing AR to DOR [1] used random traffic, so this report must use random traffic to refute those results. Indeed, **random traffic favors AR** *because localized traffic would lead to shorter path lengths and less path multiplicity; without path multiplicity, "adaptive" routing is impossible. For localized traffic, there would be no reason to consider AR. It might be possible to concoct a non-localized pathological traffic pattern that favored AR, but such chimera are not relevant to the practical engineering of general-purpose routing networks. Random traffic is the worst-case model for non-localized communication, since any pattern can be randomized. DOR is at least as good as AR for localized communication, which lacks path multiplicity. DOR beats AR for non-localized communication because it beats AR for random traffic.*

Question 2 *What effect would finite buffering have on the results? Are infinite-buffering results realistic?*

Answer 2 *Finite buffering would increase the performance advantage of DOR, since* **infinite buffering favors AR**. *Infinite buffering eliminates AR's deadlock problem. The techniques (e.g., misrouting [1]) required to avoid deadlock when using AR with finite buffering would only decrease the performance of AR. Moreover, as can be seen from the* `AQLEN` *values, AR needs more buffering than DOR (under heavy traffic), so restricting the buffering would only further decrease AR's performance relative to DOR.*

The `AQLEN` *values also indicate that the infinite-buffering results are completely realistic because the queue lengths are typically very, very short. Indeed, a DOR simulator with FIFOs only one packet long gives the* **same** *results as a simulator with unbounded FIFOs for practical topologies [5]. In short, finite buffering would leave DOR results unchanged and worsen the AR results.*

Question 3 *What would be the effect of non-uniform traffic where there was a concentration of traffic to/from a subset of the nodes?*

Answer 3 *This question is a red herring, as explained in the Introduction. The most general rebuttal to this question is that DOR is the extant routing technique, so the burden of proof lies with the proponent of an unproven alternative. If there is a non-uniform traffic model that is sufficiently common to be of interest, then the AR proponent is obliged to demonstrate significantly superior performance for AR under that traffic model. In fact, no such non-uniform traffic model has been proposed. Indeed, a chronically non-uniform traffic pattern is liable to arise only if either the program design or process placement is poor. Dynamic hot spots are handled better by DOR than by AR. DOR dissipates traffic fluctuations more quickly because it has lower latency and higher throughput. The technical rebuttal to this question is that random traffic is the accepted standard for network analysis, so there is no obligation to consider every special case that might be concocted.*

Question 4 *How does ANS choose an output when several are allowed, and how does this choice effect performance?*

Answer 4 *As described in the simulator report [5], ANS chooses the first free allowed output, and the choice does not effect performance significantly. If several profitable outputs are free, advancing the packet in the dimension with the largest offset would preserve the most path multiplicity, but previous studies have shown no significant effect from the details of output selection [1]. Even if the output selection had a noticeable impact on path multiplicity, it would not effect the conclusions of this report. If path multiplicity were decreased, there would be less difference between AR and DOR (as in 1D); if AR is no better than DOR, then there is no justification for its greater cost/complexity. If path multiplicity were increased, the differences between AR and DOR would be amplified (as in 3D), in which case the advantages of DOR would be magnified.*

C Effect of Packet Length

The simulations presented in this report have used a packet length of `L=32`, which is realistic for the Mosaic and appropriate for comparing AR to DOR. However, it is desirable to study the effects of every parameter — all other parameters have been varied over a wide range — so this appendix will present some simulations using different packet lengths. It would be very time-consuming to repeat all of the previous simulations using different packet lengths; the simulations for this report have used > 3000 hours of CPU time on a network of ≈ 30 Sun SPARCstations. Two-dimensional meshes are the topology of interest, so this section will present data for 8×8, 32×32, and 128×128 meshes. Short packets (`L=8`) and long packets (`L=128`) will be compared with the medium packets (`L=32`) used so far. The data is presented in Table 6.

Observation 12 *The latency difference between AR and DOR generally increases with packet length.*

Observation 13 *Cut-through latency generally increases with packet length.*

The performance difference between AR and DOR is amplified by packet length. If AR latency is slightly lower than DOR latency, then this difference increases with `L`. If AR latency is greater than DOR

Parameters				**DOR** (SNS)		**AR** (ANS)		Difference
d	R	A	L	T	AQLEN	T	AQLEN	$\frac{T_{ANS}}{T_{SNS}} - 1$
2	8	.1	8	7.20	.01	6.98	.01	-3%
2	8	.1	32	9.79	.00	9.09	.00	-7%
2	8	.1	128	20.5	.00	17.5	.00	-15%
2	8	.3	8	10.6	.05	9.58	.04	-10%
2	8	.3	32	21.6	.02	18.8	.02	-13%
2	8	.3	128	70.1	.02	55.3	.01	-21%
2	8	.5	8	18.5	.12	16.3	.11	-12%
2	8	.5	32	53.3	.13	44.1	.10	-17%
2	8	.5	128	191	.07	153	.05	-20%
2	8	.7	8	50.5	.56	∞	∞	$+\infty$%
2	8	.7	32	179	.44	256	.92	+43%
2	8	.7	128	713	.52	∞	∞	$+\infty$%
2	32	.1	8	23.7	.01	23.4	.01	-1%
2	32	.1	32	26.9	.00	26.2	.00	-3%
2	32	.1	128	40.5	.00	37.6	.00	-7%
2	32	.3	8	27.7	.02	27.0	.02	-3%
2	32	.3	32	41.6	.01	39.2	.01	-6%
2	32	.3	128	99.7	.01	87.6	.01	-12%
2	32	.5	8	36.5	.06	36.2	.06	-1%
2	32	.5	32	74.5	.03	70.7	.03	-5%
2	32	.5	128	230	.02	215	.02	-7%
2	32	.7	8	60.1	.11	106	.27	+76%
2	32	.7	32	159	.08	256	.15	+61%
2	32	.7	128	555	.08	915	.14	+65%
2	128	.1	8	88.2	.01	88.0	.01	0%
2	128	.1	32	90.1	.00	89.7	.00	0%
2	128	.1	128	106	.00	105	.00	-1%
2	128	.3	8	93.6	.02	94.2	.02	+1%
2	128	.3	32	107	.01	110	.01	+3%
2	128	.3	128	170	.00	176	.00	+4%
2	128	.5	8	104	.04	109	.04	+5%
2	128	.5	32	140	.01	152	.01	+9%
2	128	.5	128	295	.01	337	.01	+14%
2	128	.7	8	128	.07	162	.09	+27%
2	128	.7	32	222	.03	293	.04	+32%
2	128	.7	128	618	.02	861	.03	+39%

Table 6: Effect of L

latency, then this difference also increases with `L`. The `L=32` value used in this report is slightly higher than the $\texttt{L} \approx \texttt{20}$ estimated for fine-grained computations [8]. A shorter packet length would increase the advantage of DOR over AR.

Cut-through latency increases with packet length. Spooling latency also increases with packet length: $T_{spool} = \texttt{L}$. The increase in latency with packet length suggests using short packet lengths. However, short packets have relatively more overhead. For example, if two flits are required for a packet header, and there are P flits of payload per message, then $n = \lceil \frac{P}{L-2} \rceil$ packets are required for each message, and there are $2n$ overhead flits per message. In this case, the applied load would increase as the packet length decreased: $\texttt{A} \propto 1 + \frac{2n}{P} \approx 1 + \frac{2}{L-2}$. Thus, decreasing the packet length increases the applied load and thereby increases latency. The competition between latency increasing with packet length and applied load decreasing with packet length implies an optimal packet length. Determining the optimal packet length for interesting topologies might be an interesting topic for future study.