

Bufferless all-optical networking with erasure codes

Authors

Sagnik Ghosh
University of California, Berkeley
13334 Rose St.
Cerritos, CA, 90703
(714) 588-5235
sghosh@berkeley.edu

Venkat Anantharam
271 Cory Hall
EECS Department
University of California, Berkeley
(510) 643-8435
ananth@eecs.berkeley.edu

Abstract - An idea for enabling all-optical packet switched networks is to use deflection routing together with erasure recovery coding at the packet level. Deflection routing has been generally considered in the past only in highly structured networks. The purpose of this paper is to report results of a simulation experiment to evaluate several topology-based routing algorithms for deflection routing in unstructured networks of switches in order to determine how much packet level erasure recovery coding may be needed in such networks.

Keywords – networking, optical networking, erasure recovery coding

Bufferless all-optical networking with erasure codes¹

Sagnik Ghosh and Venkat Anantharam

EECS Department,
University of California
Berkeley, CA 94720

Abstract – An idea for enabling all-optical packet switched networks is to use deflection routing together with erasure recovery coding at the packet level. Deflection routing has been generally considered in the past only in highly structured networks. The purpose of this paper is to report results of a simulation experiment to evaluate several topology-based routing algorithms for deflection routing in unstructured networks of switches in order to determine how much packet level erasure recovery coding may be needed in such networks.

I. INTRODUCTION

One of the major bottlenecks to the large scale deployment of all-optical networks is the absence of a viable device that can function as an optical buffer. Fiber loops are often proposed for this purpose, but are unsatisfactory since they are difficult to realize in integrated optics and do not functionally mimic a buffer from an input-output point of view. Deflection routing in networks, in both slotted and unslotted environments, is often proposed for optical networks [3] to deal with the absence of optical buffers. However, this concept is generally considered viable only in regular interconnection networks, such as the ShuffleNet or Manhattan networks [4], and the most extensive investigation of deflection routing has only been for regular networks. In recent years considerable attention has been paid to developing high performance erasure recovery codes [5]. An application that has been proposed for such codes is to use them with packet level redundancy to develop a push type protocol for multicast [1]. This mitigates the problem of acknowledgment overload: each receiver can wait until it receives enough packets to guarantee that it can reconstruct the desired information and then send a short acknowledgement to end the session.

Our investigation in this paper was inspired by these two ideas. We envision an all-optical packet switched network that uses packet level erasure recovery coded traffic streams, with deflection routing at the individual switches. With such an architecture, a given stream may have only a fraction of its packets received by the intended destination, but adequate coding would, in principle, permit recovery of the intended transmitted information. An additional advantage of this approach is that deflection routing strategies require little overhead for routing [2].

In this paper, we examine deflection routing in networks with randomly generated interconnection patterns. The success rate of deflection routing strategies is expected to depend on how the choices are made when packets need to be deflected. Besides traditional hop count based ways of assigning intended exit routes from a switch in the network,

we also consider a class of algorithms that is based on determining the eventual probabilities of going to the correct external port; this is a relatively novel approach that is motivated by the presence of randomness in deflection routing. We implemented several such algorithms on a simulation platform in order to gain experience on how to compare different styles of deciding preferred routes. The algorithms considered will be covered in section II. Details of the simulation environment are given section III. In section IV, we look at two kinds of networks: one that is superior using the shortest hop count approach and the other that is superior using the highest probabilities approach. This illustrates that there can be wide variations in the amount of coding required, depending on the algorithms and the structure of the network. We then report on our results from our randomly generated networks in section V.

II. ALGORITHMS CONSIDERED

Seven algorithms were considered: (1) the Random Routing Algorithm (RRA); (2) the global Priorities minimum Hop count Algorithm (PHA); (3) the Random Priorities minimum Hop count Algorithm (RPHA); (4) the Random slowly Changing Priorities minimum Hop count Algorithm (RCPHA); (5) the global Priorities highest Probability Algorithm (PPA); (6) the Random Priorities highest Probability Algorithm (RPPA); and (7) the Random slowly Changing Priorities highest Probability Algorithm (RCPPA).

Algorithm 1 (RRA) randomly matches the inputs of the switch to the outputs. It is an impractical algorithm, but useful nonetheless as a reference case and to run diagnostics for testing the simulation.

Algorithm 2 (PHA) maintains a global list of priorities between the various external outputs (in the simulation, the user is prompted to provide this list). These priorities are fixed throughout the simulation. In order to determine where to route the inputs, the switch uses a minimum hop table which tells the algorithm how many hops a packet must take from each output in the switch to get to the desired external output: i.e. if the switch being considered has the desired external output as an output, the hop count for this external output will be 0; if instead the switch has an output connected to another switch with hop count 0 for that external output, its hop count for this external output will be 1, and so on. Note that the minimum hop count table is computed just once, before the simulation starts. PHA first considers the packets with external output destinations of the highest global priority and assigns them to the best possible outputs in the switch, then goes on to the packets with external output destinations

of the next highest priority and assigns them to the remaining best possible outputs of the switch, and so on.

Algorithm 3 (RPA) is similar to PHA, but the priorities, which were global and fixed for all time in PHA, are generated randomly at each switch at every time step. Algorithm 4 (RCPA) is also similar to PHA, but the priorities are initially randomly generated for each switch and then stay the same, with some probability, at each time (in the simulation the user can specify, at each switch, the probabilities with which new random priorities are chosen). When it needs to change, a fresh priority is chosen at random.

Algorithm 5 (PPA) is like PHA, but instead of using a minimum hop table, it uses a maximum probability table. This table has the probability from each output of each switch of reaching each external output, assuming random assignment of packets at each switch. This table also needs to be computed only once, before the simulation starts. Algorithm 6 (RPPA) and Algorithm 7 (RCPPA) are identical to their RPHA and RCPHA counterparts, except instead of using the minimum hop count table, they use the maximum probability table.

III. SIMULATION DETAILS

A. How it Works

The simulation package was written in C++. The program takes as arguments the number of switches, the interconnection pattern of the switches, the algorithm for each switch, and any additional information required by the algorithm. The program also takes in as an input the load factors for each of the inputs as well as the number of timesteps for which to introduce packets into the network.

Packets in the network are represented by their input source and their output destination. The output destinations for a given input are chosen randomly on a packet by packet basis, based on the prescribed load factors. Packets move timestep by timestep using the chosen algorithm applied at each switch. The number of times each packet is deflected (based on the algorithm used) as well as the time the packet has been in the system is also updated at each timestep (these indicators are carried by each packet). When a packet reaches an external output, whether it is the desired external output or not, its input source, desired external output, accumulated deflection count, and accumulated time in the system are all written to file. One can determine from this, of course, whether the packet got to the correct (desired) external output. Statistics of the deflection count can be computed, after the simulation is completed, from the data written to file. The simulation is run to exhaustion even after the inputs have ceased arriving, but the numbers reported in this paper are based on the gathered output data up to the time the inputs cease arriving.

B. What was Tested

To test our algorithms, several network configurations were considered. Here we report on three kinds of networks: (A) with twenty four-by-four switches; (B) with forty four-by-four switches; and (C) with ten eight-by-eight switches. The parameters along which we report results are the number of external inputs/outputs and the load factor into the network. We selected the number of external inputs/outputs based on a fraction of the total number of inputs and outputs in the system as a whole, and report results for percentages of five percent, fifteen percent, and twenty-five percent. For each of these external input/output percentages, we report results averaged over ten different networks with randomly chosen interconnection patterns. We ran each network under load factors of seventy percent and ninety percent, run for each of the seven algorithms, each for 100,000 timesteps. In each case, all the switches in the network are assumed to use the same algorithm. The probabilities for a given packet desiring to go to an external output entering an external input were taken to be the same. Thus at any given timestep at each external input the probability of having no input would be thirty percent and ten percent, respectively, in the two cases we report results on. As for algorithm parameters, the output priorities for Algorithm 2 (PHA) and Algorithm 5 (PPA) were randomly assigned, though the priorities for each in a given network were the same. For Algorithm 4 (RCPHA) and Algorithm 7 (RCPPA), the probability that the priorities change at a switch at any timestep was taken to be 0.05, so on average each switch would change its priorities every twenty timesteps.

IV. SPECIAL CASES FOR ALGORITHMS

Before reporting our results, which are for random networks, we point out that the topology of the network can considerably influence which kinds of algorithms are superior. We give two prototypical examples to illustrate a case where minimum hop count table based algorithms are better and one where maximum probability table based algorithms are better.

A. Scenario Where Shortest Paths is Superior

For certain load factors, the network in Figure 1 exhibits clear superiority using the hop count algorithm as opposed to the probabilities algorithm. To illustrate this, we made inputs 1, 2, 3, 4, 5, and 6 always send packets to outputs 9, 10, 11, 12, 13, and 14, respectively, and left all the rest of the inputs to send no packets at all. We used the PHA and the PPA with arbitrarily assigned priorities run for 10,000 trials. The probabilities algorithm, being loaded into the 4x4 switch, would select to go to the two 2x2 switches as opposed to the 8x8 switch, since the packets have double the probability of reaching their destination going through the 2x2 switch. However, since we have loaded inputs 1, 2, 3, and 4 as well with the same switch destination as the inputs

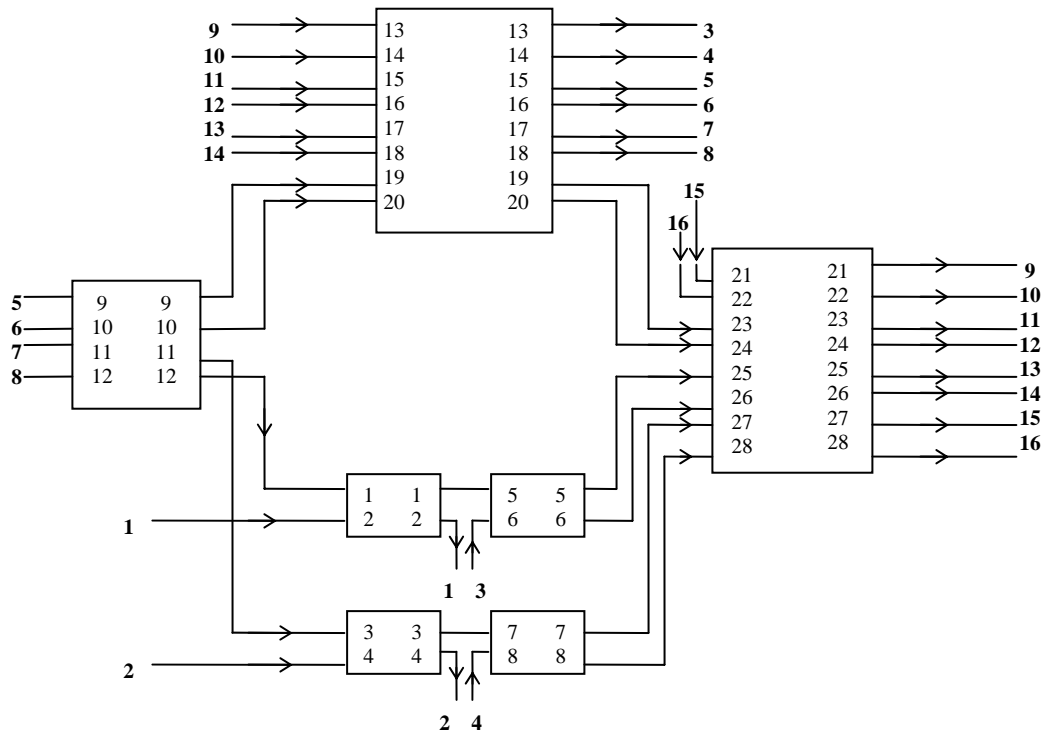


Figure 1. Superior Hops Network

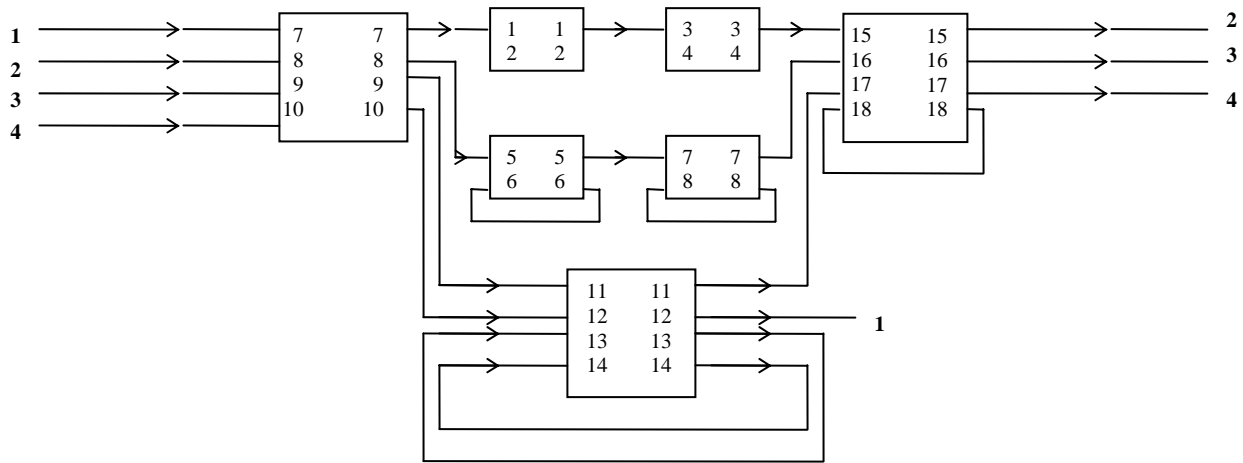


Figure 2. Superior Probabilities Network

coming from 5 and 6 (the second 8x8 switch), we see many packets going to the wrong output. For the PHA, however, since the shortest number of hops is through the first 8x8 switch, all the packets reached their destination. The PPA yielded a success rate of 0.667, while the PHA got every packet through successfully. The success of the PHA over the PPA is attributed to the difference in the load factors from the inputs here; neither algorithm takes the load factors into

account, but in this case the PPA fails to be as effective because the loads are not optimal for it.

B. Scenario Where Probabilities is Superior

The network in Figure 2 shows clear superiority of the PPA over the PHA, once again with arbitrary priorities. For this network, we loaded inputs 1, 2, and 3 with equal

chances to reach outputs 2, 3, and 4, and loaded nothing into input 4. The key feature here is that the PHA will route as many packets as possible through the second 4x4 switch, while the PPA will try to route its packets through the 2x2 switches, as they have a fifty percent chance of reaching the desired switch (the third 4x4 switch). The PHA had overloaded the second 4x4 switch with packets, so many were sent through output 1 even though that was not their true destination. Since there is only one path from this switch to the third 4x4 switch, many packets cannot get through to their destination if this switch is loaded with inputs. Our results show that for the PPA, the packets reached their desired destination with probability 0.97, while for the PHA the probability was only 0.5. The PPA manages to get through almost double the amount of packets in this case; once again, however, this is dependent on the load factors involved. The load factors are not optimized for the PHA in this scenario.

V. RESULTS

We looked at the successful transmission rate averaged over all ten networks for the set of parameters: the percent of external inputs/outputs, the load factor, and the algorithm used. We also looked at the average time and deflection count in the system per packet for this set of parameters and the histogram of the time and deflection count. The RRA yielded the expected result, with an average success rate being the reciprocal of the number of external inputs/outputs. All of the rest of the algorithms for the five percent external input/output case were almost always successful, with success rates very close to 1. In the case of the fifteen percent external input/output case, we begin to see some variation among the algorithms.

In the case of ten 8x8 switches, we see very high success rates for all conditions under the 70 percent load factor, and we begin to see that the hop count algorithms are slightly less successful than the probabilities algorithms, and the randomly changing priorities algorithms are more successful than their fixed priorities counterparts by about 5 percent. As we go from 15 percent external i/o's to 25 percent external i/o's, we see these trends accentuated, the difference here between the PHA and the RCPA at 15.5 percent.

As we go to the case of twenty 4x4 switches, we continue to see the same trends, but further magnified as the success rates decrease. Although the number of internal inputs/outputs is the same, we see a significant decrease in the success rates across the board from the ten 8x8 case to this case. The success rate differences between the PHA and the RCPA however are not as significant, holding all else constant, so we see that the RCPA's advantage over the PHA tends to decrease with switch size.

In the case of forty 4x4 switches, we see the same trends once again, but although the percentage of external inputs/outputs stays constant as we increase the number of

internal inputs/outputs, we see a significant decrease in the success rate after the 5 percent external i/o case.

For the deflection counts, we see these in the ten 8x8 case tend to increase with the percent of external i/o's at a 70 percent load factor. However, at a 90 percent load factor, the deflection counts for the 15 percent external i/o case are lower than the other i/o cases. This is perhaps attributed to the lower success rate in general, and that packets instead of accumulating a high deflection count simply get routed to a wrong output. In the twenty 4x4 case, we see an overall increase in the deflection count over the ten 8x8 case. This also holds true for the forty 4x4 case over the ten 8x8 case. We should also note the differences among the algorithms. The deflection counts between the random priorities algorithms for the hop count algorithms and the probabilities algorithms varies little, but tends to show a difference of about 0.2 as we see a drop in success rates. The fixed priorities algorithms have the same trend, but magnified, having a difference of about 1.5 at the case of forty 4x4 switches at a 90 percent load with 25 percent external i/o's.

TABLE I
10 8x8 SWITCH AVERAGE SUCCESS RATES

i/o %'s:	70% load			90% load		
	5 percent	15 percent	25 percent	5 percent	15 percent	25 percent
alg 1:	0.2501	0.08322	0.04999	0.2499	0.08319	0.04995
alg 2:	1	0.9945	0.9145	0.9978	0.8796	0.6308
alg 3:	1	0.9944	0.9285	0.9977	0.9362	0.7676
alg 4:	1	0.9948	0.9296	0.998	0.9367	0.7605
alg 5:	1	0.9965	0.9329	0.9988	0.8945	0.6297
alg 6:	1	0.9955	0.9461	0.9983	0.9478	0.7852
alg 7:	1	0.9966	0.9474	0.9989	0.9498	0.7772

TABLE II
20 4x4 SWITCH AVERAGE SUCCESS RATES

i/o %'s:	70% load			90% load		
	5 percent	15 percent	25 percent	5 percent	15 percent	25 percent
alg 1:	0.2504	0.08336	0.04998	0.2499	0.08333	0.05009
alg 2:	0.999	0.9184	0.5992	0.9937	0.7452	0.4237
alg 3:	0.9989	0.92	0.6347	0.9931	0.7893	0.5036
alg 4:	0.9989	0.9209	0.6332	0.9939	0.7903	0.5006
alg 5:	0.9997	0.9527	0.6129	0.9983	0.7563	0.4259
alg 6:	0.9997	0.9521	0.6495	0.9974	0.8228	0.5142
alg 7:	0.9997	0.9535	0.6478	0.9982	0.8245	0.5115

TABLE III
40 4x4 SWITCH AVERAGE SUCCESS RATES

i/o %'s:	70% load			90% load		
	5 percent	15 percent	25 percent	5 percent	15 percent	25 percent
alg 1:	0.1249	0.0417	0.02497	0.125	0.0416	0.02505
alg 2:	0.9968	0.8093	0.4556	0.9833	0.5716	0.3129
alg 3:	0.9967	0.8305	0.4845	0.9846	0.6599	0.3702
alg 4:	0.9968	0.831	0.4827	0.9859	0.6588	0.3686
alg 5:	0.9981	0.8427	0.4558	0.9926	0.5756	0.3083
alg 6:	0.9981	0.8641	0.4931	0.9923	0.6883	0.3766
Alg 7:	0.9982	0.8648	0.4914	0.9935	0.6874	0.375

TABLE IV
10 8x8 DEFLECTION COUNT PERCENTS FOR SUCCESSFUL PACKETS

		5 percent		15 percent		25 percent	
		alg 4:	alg 7:	alg 4:	alg 7:	alg 4:	alg 7:
70%	0	0.21	0.22	0.22	0.23	0.2	0.2
	1	0.33	0.31	0.29	0.29	0.24	0.24
	2	0.22	0.23	0.21	0.2	0.21	0.21
	3	0.092	0.088	0.1	0.1	0.12	0.12
	4	0.054	0.058	0.062	0.06	0.075	0.074
	5+	0.088	0.092	0.12	0.12	0.15	0.15
90%	0	0.13	0.14	0.15	0.15	0.21	0.21
	1	0.21	0.21	0.19	0.18	0.22	0.22
	2	0.17	0.18	0.16	0.16	0.19	0.19
	3	0.1	0.1	0.11	0.11	0.12	0.12
	4	0.078	0.076	0.083	0.082	0.078	0.078
	5	0.3	0.3	0.31	0.32	0.18	0.18

TABLE V
20 4x4 DEFLECTION COUNT PERCENTS FOR SUCCESSFUL PACKETS

		5 percent		15 percent		25 percent	
		alg 4:	alg 7:	alg 4:	alg 7:	alg 4:	alg 7:
70%	0	0.14	0.14	0.094	0.092	0.14	0.14
	1	0.2	0.19	0.17	0.17	0.18	0.18
	2	0.28	0.27	0.19	0.18	0.19	0.19
	3	0.15	0.15	0.16	0.15	0.15	0.15
	4	0.078	0.087	0.1	0.1	0.1	0.1
	5+	0.15	0.16	0.28	0.29	0.24	0.25
90%	0	0.089	0.095	0.091	0.089	0.17	0.16
	1	0.12	0.13	0.14	0.14	0.19	0.19
	2	0.18	0.18	0.15	0.15	0.19	0.19
	3	0.13	0.13	0.13	0.13	0.14	0.14
	4	0.09	0.091	0.097	0.098	0.096	0.098
	5+	0.39	0.38	0.38	0.39	0.22	0.22

TABLE VI
40 4x4 DEFLECTION COUNT PERCENTS FOR SUCCESSFUL PACKETS

		5 percent		15 percent		25 percent	
		alg 4:	alg 7:	alg 4:	alg 7:	alg 4:	alg 7:
70%	0	0.056	0.058	0.057	0.055	0.1	0.1
	1	0.12	0.12	0.1	0.098	0.13	0.13
	2	0.22	0.21	0.15	0.14	0.17	0.17
	3	0.19	0.2	0.15	0.15	0.16	0.16
	4	0.11	0.12	0.12	0.12	0.12	0.12
	5+	0.29	0.3	0.42	0.44	0.31	0.32
90%	0	0.037	0.039	0.066	0.064	0.13	0.13
	1	0.069	0.076	0.1	0.099	0.15	0.15
	2	0.13	0.13	0.14	0.13	0.17	0.17
	3	0.13	0.13	0.14	0.13	0.16	0.16
	4	0.094	0.096	0.11	0.11	0.12	0.12
	5+	0.55	0.53	0.45	0.46	0.27	0.27

VI. CONCLUSIONS

The results show that under reasonably high stress conditions, the deflection routing scheme using randomly changing priorities with the probability table yield high success rates for packet delivery. The difference in the time spent for successful packets in the system is realistically very small between the shortest hop count algorithms and the probability table algorithms. From our test results, this algorithm is the superior one. We also saw a significant boost in the success rates as the switch size increases, so with larger switches this method of routing will have higher chance of success. Our results indicate that deflection routing based networks with these routing strategies, coupled with erasure coding, seem quite feasible.

REFERENCES

- [1] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE JSAC*, vol. 20, no. 8, pp. 1528-1540, Oct. 2002.
- [2] G. Castanon, L. Tancevski, and L. Tamil, "Routing in all-optical packet switched irregular mesh networks," *IEEE GLOBECOM*, vol. 1B, pp. 1017-1022, 1999.
- [3] T. Chich, J. Cohen, and P. Fraigniaud, "Unslotted deflection routing: a practical and efficient protocol for multihop optical networks," *IEEE/ACM TON*, vol. 9, no. 1, pp. 47-59, Feb. 2001.
- [4] V. O. K. Li and A. K. Choudhry, "An approximate analysis of the performance of deflection routing in regular networks," *IEEE JSAC*, vol. 11, no. 8, pp. 1302-1316, Oct. 1993.
- [5] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE-IT*, vol. 47, no. 2, pp. 569-584, Feb. 2001.