Specification and Switch Design for a Hypercube Routing Network

RODRIC M. RABBAH AND KIRAN PUTTASWAMY
Georgia Institute of Technology

1. FUNCTIONAL SPECIFICATION

We define a hypercube network of N processors or *switches*, where each processor has a unique id $pid \in \{0, \ldots, N\}$ which is encoded as a binary string of length $n = log\ N$. Any two switches in the network are connected by a duplex communication channel if and only if their processor ids differ in exactly one position. The number of incoming and outgoing communication channels per switch is $D = n$ and this is the *fan-in/out* degree of each processor.

Each processor in the network is initially assigned a packet $k = \{destination\ pid, data\}$ destined for another processor in the network. Each processor shall have at most one packet, and each packet shall have a unique destination (i.e., no two packets have the same destination). The process of transmitting the packets to their destinations is known as *routing*. A single switch (i) may simultaneously receive up to n packets – one along each of its incoming channels, (ii) simultaneously decides to route or retain each of the packets, and (iii) if any of the packets are to be routed, they are dispatched along the proper outgoing channels. Only one packet may be routed along a particular outgoing channel at a time, and hence if two packets incident on a processor are to be routed along the same channel, one is routed and the other retained in the output buffer until the channel is free.

Each processor in the network implements an *oblivious* routing strategy in which the route of any packet is independent of the route chosen for any other packet. Specifically, a packet is routed along the d^{th} outgoing channel if the packet's destination processor id differs from the current processor id in the d^{th} bit position and d is minimal (i.e., it is the smallest index for which the two bit strings differ).

Rodric M. Rabbah and Kiran Puttaswamy
School of Electrical and Computer Engineering, Georgia Institute of Technology, USA {rabbah, kiranp}@ee.gatech.edu
This work is supported by DARPA Contract No. F30602-02-2-0124.

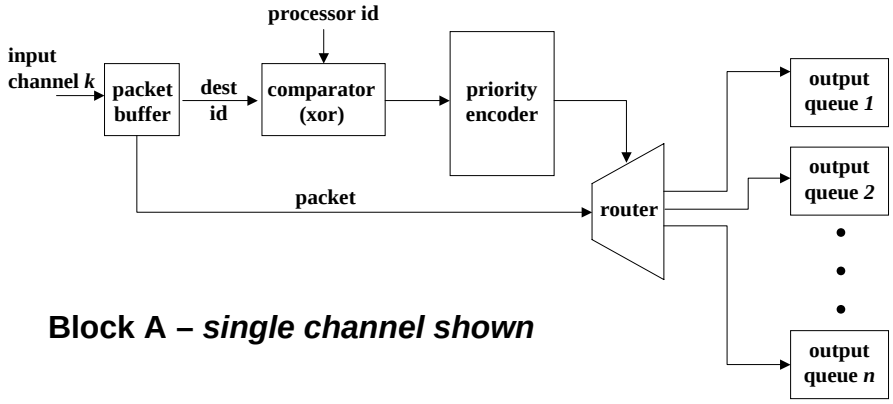

Figure 1. High-level circuit for a deterministic switch.

2. DESIGN SPECIFICATIONS - DETERMINISTIC ROUTING

A D-dimensional hypercube network of P processors which implements an oblivious, *deterministic*, bit-fixing routing algorithm consists of P single switches where each switch is as described below.
We shall detail the design of a single switch (SS) with $D = n$ in/out-communication channels. The hypercube routing network consists of N (properly) interconnected single switches; we shall refer to a routing network as a multi-switch network or MS for short.
In Figure 1 we illustrate a high-level circuit for a single channel within a switch. A single switch is comprised of D such circuits. Each circuit consists of

— *Input queue* to store (at most) one packet at time.

— *Output queue* of depth w to store as many as w outgoing packets.

— *Comparator* component to determine if a packet needs to be routed.

— *Priority Encoder* to determine which output queue the packet is to be assigned to.

— *Router* to assign a packet to the appropriate outgoing queue.

REMARK 1. *If a packet is not routed by a switch then that packet is said to have reached its destination – that is, the packet's destination processor id matches the processor id of the switch which is currently processing it. A packet which has reached its destination is removed from the network.*

REMARK 2. *Since a packet is always routed if it is not at its destination, the input queue need not accommodate more than one packet at a time. On the other hand, packets to be routed may be queued along an outgoing channel since only one packet may be routed at a time – other packets to be routed along the same channel are forced to wait until some time in the future when the channel is available for transmission.*

REMARK 3. *Let a packet with a destination processor id α be on a processor with id γ. A packet is routed if the bitwise xor of α and γ is non-zero. This is known as routing via bit-fixing.*

REMARK 4. *The priority encoder determines which of D output channels the packet is to be routed to; recall that a packet that has reached its destination is removed from the network. The encoder simply determines the smallest index i such that $\alpha(i) \neq \gamma(i)$.*

REMARK 5. *The router assigns the packet to the i^{th} output queue as determined by the priority encoder.*

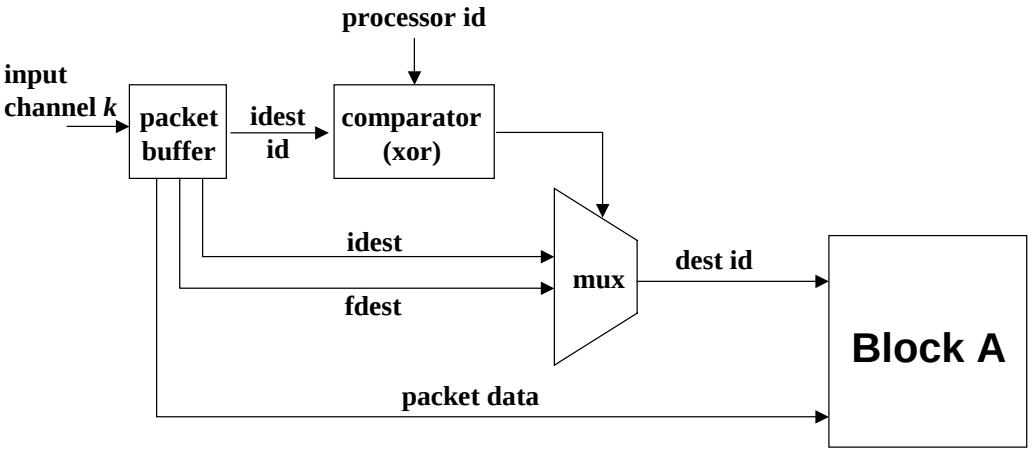

Figure 2. High-level circuit for the first randomized routing variant.

3. DESIGN SPECIFICATIONS - RANDOMIZED ROUTING

We define a randomized routing of a packet originating at pid α and destined for γ as follows. The packet is first routed to a randomly chosen intermediate destination η, and subsequently routed from η to its intended destination γ.
There are two methods for implementing randomized routing of the sort above.

3.1 First Variant to Randomized Routing

The first randomized routing variant involves the generation of an intermediate destination η at the originating processor α; the routing then proceeds deterministically. When the packet reaches its randomly chosen destination, the processor must recognize that the packet ought to be routed to its intended destination γ, and hence initiates routing the packet accordingly (also in a deterministic fashion). The switch illustrated earlier in Figure 1 is modified slightly to implement the new desired functionality. The augmented switch is illustrated in Figure 2.

REMARK 6. *Whereas a packet previously only contained its destination pid and data, it now also contains the randomly chosen intermediate destination.*

REMARK 7. *We consider the randomized routing as two distinct routing phases. Within each phase, the destination of the packet shall be known as the immediate destination. Hence, in phase one of the routing whereby the packet is routed from α to η, the immediate destination is η. By contrast, in phase two of the routing from η to γ, the immediate destination is γ.*

REMARK 8. *We shall encode the immediate destination of a packet as bits $1\ldots n$ and bits $n+1\ldots 2n$ will be reserved for the intended or final destination of a packet. The remaining bits are used to encode the packet's data.*

REMARK 9. *In order to detect that a packet has reached its intermediate destination, a destination exchange component compares the immediate destination with its pid. When they are equal – signifying that the packet has reached its immediate destination – bits $n+1\ldots 2n$ overwrite bits $1\ldots n$. The remainder of the circuit components function as described in the previous section.*

3.2 Second Variant to Randomized Routing

The other approach to randomized routing is to randomly chose to route a packet along a particular outgoing channel at every routing step. That is, let a packet *k* originate at a switch with pid α, and let $\alpha_{\bar{i}}$ be the switch whose pid differs from

α in the i^{th} bit position. The first routing decision is whether to route the packet to $\alpha_{\bar{i}}$ where $i = 1$. The subsequent routing decision is whether to route the packet along the second dimension, namely $i = 2$. The first routing phase whereby the packet is routed to an intermediate destination is said to complete when $i = D$. The second routing phase transmits the packet to its intended destination.

The decision to transmit a packet in the first phase of the routing is based on a random event. For example, with probability p the packet is not routed along the i^{th} communication channel when the i^{th} routing decision is made for a packet. Therefore, the packet is retained for an additional step and the $i+1$ dimension is considered during the subsequent round. This implies that the switch must provide additional storage to retain such packets. Providing such storage complicates the circuit design as well as the queuing discipline for selecting which packets to process during a routing step. Thus, instead of randomly making one decision for a packet, a random decision is made for each of the D dimensions. A *dimension selector* arbitrates which decision is to be committed during a specific routing step. Constraining the selector's decision is a *mask* carried by the packet which conveys to the switch which of the dimensions it may consider (i.e., which dimensions have not already been processed during previous routing steps). Once a decision is made, a new mask is generated and written to the designated packet location. Finally, the packet is assigned to the appropriate output queue.

Should the switch randomly decide not to route a packet along *any* of the allowed and remaining dimensions, the packet is forcefully transmitted to a one of the neighboring switches (chosen at random). This is to avoid the problem noted earlier which may arise in terms of internally storing a packet because the switch decides not to transmit it anywhere during the first phase. While such occurrences occur with very low probability, they may occur and hence the need for the extra circuitry.

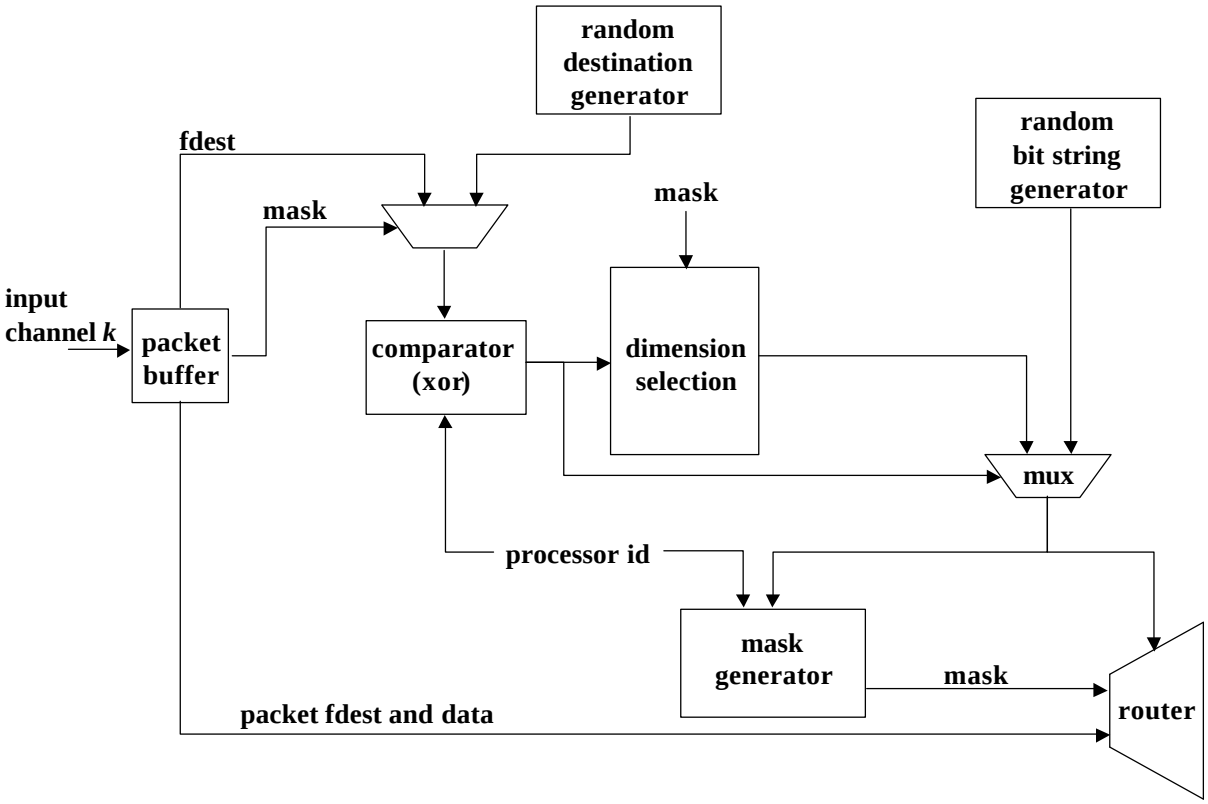

Figure 3. High-level circuit for the second randomized routing variant.

To summarize, during phase one of the second randomized routing variant, a switch performs the following tasks – note we only define the tasks per input channel as each incoming channel carries out the same tasks in parallel.
Let $T_k = 1, \ldots, n$ for a packet k originating out of a processor with pid α.

01.if a packet s exists in the input buffer
02. if T_s is not empty
03. $i \leftarrow$ smallest index $j \in T_s$
04. $T_s \leftarrow T_s - j$
05. $route_i \leftarrow$ random $a \in \{0,1\}$
06. if $route_i = 1$ then
07. assign s to output buffer i
08. end if
09. end if
10.end if

The circuit associated with the tasks above is illustrated in Figure 3. Note that the set T for a packet is encoded as a *mask*. Also note that task 05 is carried out concurrently for all $i \in T$. Should all randomly chosen decisions agree not to route a packet, a second component labeled the *alternate destination selector* forces the packet to one of the D neighboring

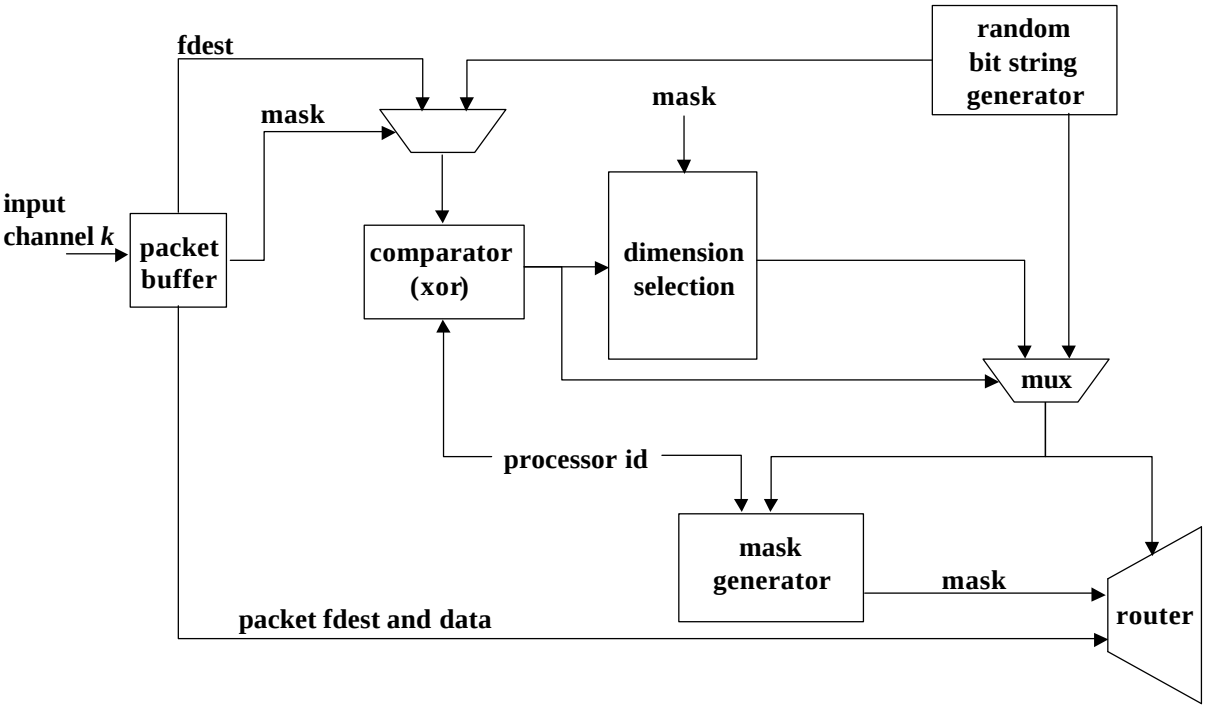

Figure 4. High-level circuit for the second randomized routing variant.

switches (with equal probability) and the mask updated to encode the empty set T. Hence, once the packet is transmitted, subsequent routing decisions transmit the packet to its final destination.

REMARK 10. *The packet mask shall contain an additional bit to indicate that all dimensions have already been fixed and hence a switch should route the packet to its final destination. This bit is set accordingly by the mask generator.*

REMARK 11. *Note that in Figure 3 there are two random bit sources. It may be possible to reuse one of the two to provide the functionality of the other. This is illustrated in Figure 4.*

4. EVALUATION FRAMEWORK

The following oblivious bit fixing routing algorithms were implemented and simulated:

(1) Deterministic routing (Det-R)

(2) Synchronized randomized routing in two phases (Rand-Sync)
   — The first phase sends packets to randomly chosen destinations. The random destination is pre-computed before the first phase begins – this is the case for the next two variants as well.
   — The second phase, which begins after the first phase completes for all packets, sends the packets from their randomly chosen destinations to their intended final destinations.

(3) Transient randomized routing (Rand-Trans)
   — Packets which complete their first phase immediately begin the second phase - note that in this algorithm variant, the packets are simply placed in the output queue and processed on a first come first served basis.

(4) Transient randomized routing with out of order processing (Rand-Trans-OOO)
   — Same as the third variant above with the exception that packets await transmittal in an output queue are prioritized such that those in the first phase of the routing are always transmitted before packets in the second phase.

(5) Transient distributed randomized routing (DRand-Trans-OOO)
   — Unlike the previous variants, here the random destination is chosen incrementally at each routing step.

For each of the above variants, networks with $N = 2^2 \; -- \; 2^{18}$ switches were simulated. For each network configuration, the following packet distributions were simulated:

— One packet per switch – each switch is the final destination of exactly one packet.

— As many as D packets per switch where $D = log_2(N)$ and no switch is the final destination of more than D packets.

Each of the algorithm variants was simulated 100 times, and below the average results across all trials are reported.

5. EXPERIMENTAL RESULTS

The following are the main observations of the results:

— Rand-Sync is the worst performing of the four randomized routing variants. This is intuitively obvious since the second phase does not start until the all packets complete the first phase.

— Rand-Trans-OOO affords no significant improvements over Rand-Trans. The performance difference - measured in terms of routing steps - was only a few percent.

— The number of packets reprocessed by any particular switch during a routing step is the highest of the DRand-Trans-OOO variant. This is reasonable since at any given instance during the first routing phase, a decision may be made not to route a packet along a particular dimension - hence the packet must be reprocessed and a decision made for the next dimension. The number of reprocessed packets per switch is roughly $log(N)$. By contrast, a packet is reprocessed at most once when the intermediate destination is pre-computed prior to the first phase.

— With the exception of the *transpose* permutation, deterministic routing outperforms the randomized routing variants 20-30% on average.

— In the case of the transpose permutation the performance benefits are $4-8\times$ depending on the randomized routing variant. In this context, Rand-Trans-OOO and DRand-Trans-OOO perform comparably well and better than Rand-Trans – as noted earlier however, OOO processing is only marginally better.

The performance advantage of deterministic routing versus randomized routing in the majority of the packet distributions is attributed to the fact that in the case of randomized routing, there are two separate routing phases; the number of steps in both routing phases will most likely exceed the number of steps for Det-R.

All the data collected can be found in two electronic files: `all-across.xls` and `all-per.xls`. The format of each file is as follows. The first described the network configuration `Sn.p` where `n` is the dimensionality of the network and `p` is the number of packets per switch. Hence, `S4.3` represents a network with 2^4 switches and 48 total packets (3 per switch). The second column describes the data to be presented, which may be one of four categories:

(1) Steps Speedup: the improvement in the number of routing steps relative to the baseline deterministic routing; a number lower than one implies degradation and a number greater than one implies improvement.

(2) Average Congestion: the average number of routing steps a packet was delayed because of collisions with other packets along its route.

(3) Percent XPackets: the number of packets reaching their destination without encountering any congestion, as a percentage.

(4) Max OQD: the maximum depth of the output buffer in the network.

Columns three through six represent the data for Det-R, Rand-Trans, Rand-Trans-OOO, and DRand-Trans-OOO respectively. The first file groups the data by the number of packets per switch. The other file groups the data according to the number of switches in the network.