Performance Evaluation of Dynamic Network Reconfiguration using Detour-UD Routing

Tsutomu YOSHINAGA, Yasuhiko NISHIMURA
Graduate School of Information Systems, University of Electro-Communications, Chofu-shi, Tokyo, 182-8585, Japan.
E-mail: yosinaga@is.uec.ac.jp

Abstract

Fault-tolerance is an emerging issue for massively parallel computers. This paper describes the performance impact of dynamic network reconfiguration protocols using a fault-tolerant, adaptive deadlock-recovery routing algorithm, Detour-UD, for k-ary n-cubes. We propose a scheme to specify unroutable packets by managing drain-flags in routing tables. We also propose two selective drainage protocols. One protocol drains the unroutable packets specified by the drain-flags after the reconfiguration process. The other protocol drains deadlocked packets to reduce the network load during the reconfiguration process. Our simulation results show that the first protocol helps reduce the number of drainage packets, and the second one keeps the network throughput during the reconfiguration process.

Keywords: dynamic network reconfiguration, fault-tolerance, adaptive routing, deadlock recovery, k-ary n-cube

1. Introduction

Fault-tolerance is becoming important for massively parallel processors (MPPs) as the system scale is enlarged. Flexible fault-recovery routing algorithms based on network reconfiguration have been recently proposed [1, 6, 8, 13]. These algorithms support dynamic network reconfiguration that does not totally stop user communication even during the reconfiguration process. Hence, performance degradation can be suppressed compared with static reconfiguration, which drops all in-flight packets once and retransmits them later.

The reconfiguration process is initiated from a node which detects a fault. Then, control packets, including those with the faulty information, are propagated to every non-faulty node in the network to adjust the routing function. When a router receives the control packet, it switches its operation mode for the network reconfiguration. Generally, the adjustment of the routing function on each node is asynchronous, and completion of the reconfiguration process requires global or local synchronization with other nodes. Another way to handle the asynchronous reconfiguration process is maintaining versions of the routing table [8]. However, in all cases, permanent deadlocks which are caused by the faults must be resolved by the reconfiguration protocol. Deadlock-freedom can be guaranteed by removing the set of unroutable packets from the network, even when the network does not have special channel resources for the reconfiguration process. In this case, the dynamic network reconfiguration scheme should also reduce the number of dropped packets to minimize retransmission overhead. Fortunately, direct networks such as k-ary n-cubes can treat the unroutable packets by draining them at the nodes where their header flits exist. Then, the drained packets are reinserted into the network toward their destinations after the reconfiguration process. Consequently, they often suffer considerable latency. Moreover, continuing the transmission of user’s packets during the network reconfiguration process may increase the number of unroutable packets by forming congestion around a fault. Therefore, careful treatment is required.

This paper considers the efficiency of network reconfiguration protocols. We propose a scheme to specify unroutable packets during the reconfiguration process by computing drain-flags, which are the information attached to each entry of the routing table. In order to reduce the number of drainage packets, we set the drain-flags only for the unavailable deadlock-recovery paths caused by the dynamic network reconfiguration. We also examine two additional protocols to reduce the network load during reconfiguration. One protocol is simple injection throttling and the other is optional drainage of deadlocked packets. Our simulation results suggest that a dynamic selection of these protocols based on the network load is attractive for reducing the number of drainage packets and keeping graceful network performance.

The rest of the paper is organized as follows. Section 2 explains the network reconfiguration process and the requirement of packet drainage. Section 3 presents four protocols of network reconfiguration for the Detour-UD routing algorithm [13] that we recently proposed. Section 4 gives
the simulation results of network throughput and average packet latency. Section 5 discusses the impact of the network reconfiguration protocols on performance. We conclude this paper in Section 6.

2. Network Reconfiguration

This section describes a network reconfiguration process based on Up*/Down* routing [9], which is one of the most popular routing algorithms for irregular networks.

2.1. Related Work

The main task of network reconfiguration is supplying a new routing function that adjusts to a change of the network topology. Figure 1 shows an example of network reconfiguration on a ternary 2-cube.

In Up*/Down* routing, we define a channel direction for each physical channel using a spanning tree of the network. Figure 1(b) shows a spanning tree based on the breadth-first search (BFS) algorithm. We specify the root of the tree as node number 00. In this figure, arrows show the up-directions, and the opposite directions are defined as down. A routable path on this tree is restricted to take zero or more up hops on the channels followed by zero or more down hops. This restriction of the path selection does not introduce cyclic channel dependency. Here, we suppose a channel between node 00 and 10 breaks down. Then, node 10 loses an up channel toward the root node; therefore, the spanning tree needs to be reconfigured as Figure 1(c). Because we can always assign acyclic paths between any pair of a source and destination, Up*/Down* routing is deadlock-free on any connected network topology.

Schroeder et al. proposed the automatic reconfiguration of a network that drops all in-flight packets once and then recomputes the routing tables for the new spanning tree [9, 10]. Recently, Casado et al. proposed a dynamic network reconfiguration algorithm that does not need to drop whole packets [1]. Unfortunately, it requires a fairly complex procedure for the networks which do not support multiple virtual channels (VCs). Pinkston et al. proposed a double VC scheme that prepares two sets of VCs per physical channel. In the double VC scheme, only one set of VCs is available at a time, and the available VC set is switched during the network reconfiguration. This is useful for breaking inconvenient channel dependency, which may be configured at an unstable state of the routing tables during the reconfiguration process. However, it is costly because only one of the two set of VCs is usable in the normal state.

The Detour-UD routing algorithm, which we focus on in this paper, also requires two sets of VCs per physical channel [13]. It does not need to switch the available VCs between non-faulty and faulty networks. One VC, called the FA-VC, is used for fully adaptive routing and the other VC, called the UD-VC, is used for deadlock recovery based on Up*/Down* routing. Packets are first freely routed using FA-VCs. When deadlock is detected, a blocked packet escapes from the deadlock and is delivered to its destination node using UD-VCs. By preparing deadlock-recovery up/down paths for any packet, deadlock can be safely broken all the time and FA-VCs are available without any restriction, including non-minimal paths.

2.2. Packet Drain

This section describes why unroutable packets appear by the network reconfiguration. Let us assume a second channel fault happens between node 10 and 20 from the state of Figure 1(c), which had the first channel fault between node 00 and 10. Then, its BFS spanning tree is reconfigured like the one shown in Figure 2(b). This reconfiguration process varies an up/down path from node 11 to destination node 20. In the spanning tree in Figure 1(c), the up path 11-10-20 exists. On the other hand, an up/down path is supplied as 11-01-00-20 in the spanning tree in Figure 2(b). If node 11 sends a packet to destination node 20 with the assumption of taking the former path via node 10 during the reconfiguration process, and the reconfiguration process completes when node 10 receives the header flit of the packet, this packet loses the succeeding path because the channel between node 10 and 20 is faulty. Node 10 has only two up channels in Figure 2(b), so it cannot supply an up path for the down packet from node 11. A solution is ejecting such a packet once at node 10, then reinjecting it into the network after the reconfiguration. To do so, a new routing function safely supplies a delivery path for any packet. Since ejecting a packet from the network at an intermediate node breaks inconvenient channel dependency, deadlock freedom is guaranteed. The Detour-UD routing algorithm allows reinjected
packages to restart adaptive routing utilizing FA-VCs to avoid the overhead of deadlock recovery. In this paper, we identify this kind of ejection of unroutable packets as selective drainage.

In the network region apart from the faults, network reconfiguration does not change the up/down paths. For example, an up path from node 22 to node 00 via node 02 is common in both Figure 1(c) and Figure 2(b). Therefore, a packet which takes a changeless path can continue transmission even during the reconfiguration process.

In summary, it is possible to reduce the number of drainage packets by effectively identifying the unroutable packets in the reconfiguration process. Note that we do not treat packets damaged or lost by faults during their transmission. Recovery for these packets requires not only fault-tolerant routing algorithms but also support in the messaging layer.

2.3. Identification of Unroutable Packets

This paper proposes a scheme to specify unroutable packets by setting a drain-flag for each entry of the routing table. Assume each packet header includes the destination node number, and the routing table contains binary information that sets the directions of the output channel candidates for each destination. As we explained in Section 2.2, the network reconfiguration process may introduce unroutable packets that lose succeeding paths for their delivery. In Up*/Down* routing, such unroutable packets try to advance in the up direction followed by the down direction after the network reconfiguration. It happens only at a node with an output channel whose direction is changed from down to up. Therefore, we set drain-flags for the destinations that unroutable packets are destined for and request such an output channel with the direction change from down to up.

This method is a sufficient condition to identify unroutable packets, but not a necessary condition. Let us represent an up/down path from a source node $S$ to a destination node $D$ as $S \rightarrow u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_i \rightarrow d_1 \rightarrow d_2 \rightarrow \cdots \rightarrow d_m \rightarrow D$, where $u_i, 1 \leq i \leq n$ are up direction channels and $d_j, 1 \leq j \leq m$ are down direction channels. If the direction of the channel $d_1$ is changed from down to up, it does not matter under the restriction of Up*/Down* routing. A necessary and sufficient condition for the drain-flag can be found at the first position where the channel direction is changed from down to up after taking a down path; however, this condition requires channel dependency analysis. This is expensive, so we use the sufficient condition to set drain-flags.

Similar techniques to identify unroutable packets are applicable for other routing algorithms. L-turn routing [5], which can be implemented with a single VC per physical channel for any network topology, is a good example instead of Up*/Down* routing for deadlock recovery. The L-turn routing algorithm classifies channels into left and right to avoid cyclic channel dependency, and it prohibits left turns after taking a right channel. By integrating such a deadlock-avoidance based routing algorithm with true fully adaptive routing, we can easily define a new fault-tolerant routing algorithm. In the case of the L-turn routing algorithm, drain flags should be set where packets request an output channel whose direction is changed from right to left. The idea to identify unroutable packets is common in the sense that the required task is finding prohibited turns by changes of the channel direction in the network reconfiguration process.

In all cases, packets which are injected into the network after the reconfiguration process do not need to be drained. This can be controlled by including a version number of the routing table in each packet header.

3. Protocols

Our goal is maintaining reasonable communication performance by limiting the number of drainage packets, even after dynamic faults appear in a network. We examined the following four protocols, which differ in new packet injection and drainage policy. The Detour-UD routing algorithm is used for user communication, and flooding is used to propagate the control packets which are required for network reconfiguration. Delivery of the control packets can be guaranteed by repeating transmission between neighbors when there is at least one available virtual channel.
Table 1. Network reconfiguration protocols.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Injection</th>
<th>UD-VCs</th>
<th>FA-VCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>x</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>ud-sel</td>
<td>o</td>
<td>△</td>
<td>x</td>
</tr>
<tr>
<td>ud-sel/throttle</td>
<td>△</td>
<td>△</td>
<td>x</td>
</tr>
<tr>
<td>ud-sel/fa-sel</td>
<td>o</td>
<td>△</td>
<td>△</td>
</tr>
</tbody>
</table>

FA-VCs, and UD-VCs are only used in cases when no FA-VC is available. This can be done by maintaining UD-VCs as nonblocking. To do so, users’ deadlock-recovery packets in UD-VCs are always drained when they are directly blocked by faults. Hence, new deadlock recovery is postponed until completion of the network reconfiguration to keep the number of drainage packets low. Unless we later explain otherwise, a packet denotes a user packet.

**static**  Packet injection is stopped during network reconfiguration and all transmitting packets are drained.

**ud-sel**  This protocol continues injection and transmission of packets during network reconfiguration. In-flight packets using UD-VCs, whose deadlock recovery is initiated before the network reconfiguration starts, are drained only when drain-flags are set.

**ud-sel/throttle**  Packet injection is throttled during network reconfiguration to reduce the network load. The packet drainage policy is the same as ud-sel.

**ud-sel/fa-sel**  Injection and transmission of packets is unconstrained. Deadlock-recovery packets that are transmitting using UD-VCs are drained when drain-flags are set (this is the same as the rule in ud-sel). Additionally, deadlocked packets on FA-VCs are drained instead of initiating deadlock recovery during the network reconfiguration.

Table 1 summarizes the difference among these protocols.

**4. Experiments**

**4.1. Simulation**

In order to evaluate the network reconfiguration protocols, we made a network simulator utilizing the ISIS library [14]. Our simulation conditions are as follows:

**Network:** 16-ary 2-cube (256 nodes).

**Routing algorithm:** Detour-UD.

**Virtual channel:** one UD-VC and three FA-VCs per physical channel, and each has 8-flit FIFO.

**Traffic pattern:** random uniform and bit-reversal.

**Offered traffic load:** 0.2 and 0.25 (flit/node/cycle).

**Packet length:** 16 flits.

**Hop delay:** 5 clock cycles for no collision cases.

**Injection queue:** generated packets are stored in this queue up to a 24-packet length on each source node until injection into the network.

**Switching:** wormhole routing.

**Deadlock:** a time-out function detects deadlock when a packet is blocked and all its output channel candidates are stalled for longer than 32 or 64 clock cycles.

Sections 4.2 and 4.3 show the communication performance while assuming a single node fault for the random uniform and bit-reversal traffic, respectively. We assume that node 33 becomes faulty at 40-K cycles from the beginning; however, the position is not so important since the network topology is homogeneous. We also show the experimental result for a series of two faults in Section 4.4. In either case, we suppose that the network reconfiguration process takes 5-K cycles to update a routing table with the assistance of support software on each node. Hence, dispersion of the start and end time of the reconfiguration process is simplified.

**4.2. Random Traffic**

**4.2.1 Light Load Case**

Figure 3 shows the accepted throughput and average packet latency for a random traffic pattern when we offer a normalized packet generation rate of 0.2 (flit/cycle/node). We plotted the data per 500 cycles. From Figure 3(a), we notice that the static protocol ceases communication after the appearance of the fault. Its throughput goes up after the 45-K cycle when the network reconfiguration process completes and the waiting packets burst into the network. The ud-sel protocol also raises latency by ceasing communication once, but the increase is smaller than that of the static protocol. Its throughput gradually decreases because the deadlock on the FA-VCs is not recovered during the network reconfiguration. This protocol stops communication at around the 45-K cycle and then shows a similar rise of throughput. The ud-sel/throttle protocol continues communication with limiting packet injection, resulting in low throughput during the network reconfiguration. The ud-sel/fa-sel protocol maintains flat throughput by selectively draining packets from both UD-VCs and FA-VCs.

Figure 3(b) shows the average packet latency for each protocol. The static protocol incurs high latency after network reconfiguration because the waiting time for injection is added. The ud-sel protocol also raises latency by ceasing communication once, but the increase is smaller than that of the static protocol. The ud-sel/throttle and the ud-sel/fa-sel protocols maintain low latency since communication does not stop even during the network reconfiguration.

Table 2(a) shows statistical results of the simulation from the beginning to the 100-K cycle for random traffic. We
found that the difference of throughput is not very large from a long-term viewpoint. On the other hand, latency incurs large overhead depending on the period of the communication stop. Therefore, the static protocol shows the longest latency, followed by ud-sel. We also confirm that the ud-sel protocol reduces the number of drainage packets by utilizing drain-flags to identify unroutable packets. The static protocol requires the most drainage packets. In contrast, ud-sel and ud-sel/throttle protocols drain zero packets. The ud-sel/fa-sel protocol increases the number of drainage packets in order to reduce network load during the network reconfiguration. This is a trade-off between communication performance and retransmission overhead.

### 4.2.2 Heavy Load Case

Figure 4 shows the accepted throughput and average packet latency for a random traffic pattern when we offer a normalized packet generation rate of 0.25 (flit/cycle/node). This traffic pattern is one of the popular non-uniform communication patterns for evaluating network performance [3]. It is hard for this pattern to be saturated with right-offered traffic compared to the random pattern.

This situation happened because of the higher network load. Figure 4(b) shows that ud-sel incurs higher latency than in Figure 3(b). The ud-sel protocol takes longer than the static protocol to return to the low latency state because the virtual network of FA-VCs is heavily crowded when the network reconfiguration is finished. Ud-sel/throttle and ud-sel/fa-sel avoid network saturation by reducing the offered load during the network reconfiguration.

Table 2(b) shows statistical data for 100-K cycles in the case of the offered traffic rate 0.25 (flit/cycle/node). All the protocols show larger values than Table 2(a) for all categories. This is caused by the heavier offered traffic, although the tendency among protocols is similar. Based on these results, we can say that ud-sel/throttle can be a good protocol for a random traffic pattern since it achieves low latency with a reduced number of drainage packets.

### 4.3. Bit-Reversal Traffic

#### 4.3.1 Light Load Case

This traffic pattern sends packets from a source node address \((b_{n-1}, b_{n-2}, ..., b_0)\) to a destination node \((b_0, b_1, ..., b_{n-1})\). Figure 5 shows the accepted throughput and average packet latency for the bit-reversal traffic when we offer a normalized packet generation rate of 0.2 (flit/cycle/node). This traffic pattern is one of the popular non-uniform communication patterns for evaluating network performance [3]. It is hard for this pattern to be saturated with right-offered traffic compared to the random pattern.

We notice from Figure 5(a) that dynamic reconfiguration protocols, except for the static protocol, do not cease communication. The ud-sel protocol without throttling or

### Table 2. Experimental results for random traffic.

<table>
<thead>
<tr>
<th>(a) offered traffic rate: 0.2.</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>Thr. Latency</td>
<td>#drain</td>
<td>Recov.</td>
</tr>
<tr>
<td>static</td>
<td>0.194</td>
<td>247</td>
<td>154</td>
</tr>
<tr>
<td>ud-sel</td>
<td>0.199</td>
<td>186</td>
<td>0</td>
</tr>
<tr>
<td>ud-sel/throttle</td>
<td>0.195</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>ud-sel/fa-sel</td>
<td>0.199</td>
<td>101</td>
<td>34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(b) offered traffic rate: 0.25.</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>Thr. Latency</td>
<td>#drain</td>
<td>Recov.</td>
</tr>
<tr>
<td>static</td>
<td>0.240</td>
<td>279</td>
<td>258</td>
</tr>
<tr>
<td>ud-sel</td>
<td>0.242</td>
<td>374</td>
<td>3</td>
</tr>
<tr>
<td>ud-sel/throttle</td>
<td>0.243</td>
<td>136</td>
<td>3</td>
</tr>
<tr>
<td>ud-sel/fa-sel</td>
<td>0.249</td>
<td>140</td>
<td>64</td>
</tr>
</tbody>
</table>

thr.: throughput (flit/node/cycle), recov.: rates of deadlock-recovery / accepted packets.
the fa-sel maintains throughput during network reconfiguration. Figure 5(b) illustrates that only the static protocol incurs latency overhead by stopping communication during network reconfiguration.

Table 3(a) shows statistical data of the bit-reversal traffic for 100-K cycles with offered traffic rate 0.2 (flit/cycle/node). Only the static protocol demonstrates high latency and a large number of drainage packets. There is no large difference among the other dynamic protocols. Although ud-sel/fa-sel drains 7 packets, this number is smaller than that of the random traffic because of less congestion.

### 4.3.2 Heavy Load Case

Figure 6 shows the accepted throughput and average packet latency for the bit-reversal traffic when we offer a normalized packet generation rate of 0.25 (flit/cycle/node). Table 3(b) shows statistical data for 100-K cycles.

With this offered rate, the ud-sel protocol quickly ceases communication after the network reconfiguration starts. This communication stop occurs by forming temporary deadlocks on the virtual network of FA-VCs. These deadlocks delay performance recovery after network reconfiguration. The static protocol also suffers deadlock-recovery overhead after reconfiguration. This could happen when many waiting packets are injected into the network. We can confirm the reasons from Table 3(b). The static protocol drains 498 packets, which is the largest number in our experiments. The deadlock-recovery rates of static and ud-sel are relatively high compared with the other two protocols.

The ud-sel/throttle and ud-sel/fa-sel protocols maintain high throughput by reducing the network load. A difference between these two protocols is that the number of drainage packets in ud-sel/fa-sel is relatively high. This reflects the higher latency of ud-sel compared to ud-sel/fa-sel by retransmission.

### 4.4. Two Faults

Figure 7 shows the simulation results for a series of two channel faults. The first channel fault is assumed to occur...
between node 33 and 43 at a time of 4-K cycles, and the second channel fault occurs between node 66 and 76 at a time of 8-K cycles without the repair of the first fault. A random traffic pattern with a normalized packet generation rate 0.2 (flit/cycle/node) was used.

The communication behavior for the first channel fault in Figure 7(a) is quite similar to the result of the single node fault case, which is shown in Figure 3(a). The second channel fault causes a larger impact on communication performance, especially for the ud-sel/throttle protocol. We can see that the throughput of ud-sel/throttle is gradually degraded during the network reconfiguration and its latency increases similarly to that of ud-sel after the second fault. These results imply that the throttling control was not enough to maintain the communication performance against the two faults. The ud-sel/fa-sel protocol keeps high throughput and low latency by draining deadlocked packets from FA-VCs for both the first and the second faults.

Table 3. Experimental results for bit-reversal traffic.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Thr. (cycles)</th>
<th>Latency (cycles)</th>
<th>#drain</th>
<th>Recov. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>0.205</td>
<td>258</td>
<td>201</td>
<td>1.11</td>
</tr>
<tr>
<td>ud-sel</td>
<td>0.212</td>
<td>105</td>
<td>1</td>
<td>0.02</td>
</tr>
<tr>
<td>ud-sel/throttle</td>
<td>0.207</td>
<td>104</td>
<td>1</td>
<td>0.02</td>
</tr>
<tr>
<td>ud-sel/fa-sel</td>
<td>0.212</td>
<td>105</td>
<td>7</td>
<td>0.01</td>
</tr>
</tbody>
</table>

(b) offered traffic rate: 0.25.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Thr. (cycles)</th>
<th>Latency (cycles)</th>
<th>#drain</th>
<th>Recov. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>0.242</td>
<td>648</td>
<td>498</td>
<td>8.40</td>
</tr>
<tr>
<td>ud-sel</td>
<td>0.239</td>
<td>706</td>
<td>2</td>
<td>9.70</td>
</tr>
<tr>
<td>ud-sel/throttle</td>
<td>0.254</td>
<td>310</td>
<td>2</td>
<td>4.13</td>
</tr>
<tr>
<td>ud-sel/fa-sel</td>
<td>0.259</td>
<td>359</td>
<td>90</td>
<td>5.24</td>
</tr>
</tbody>
</table>

thr.: throughput (flit/node/cycle), recov.: rates of deadlock-recovery / accepted packets.

5. Discussion

As we showed in Section 4, the communication protocols for network reconfiguration have considerable impact on performance. Although we showed experimental results using the Detour-UD routing algorithm, we also examined the case when Up*/Down* routing is replaced by L-turn routing for deadlock recovery. However, the change of the deadlock-recovery algorithm scarcely affected communication performance. The main reason is that we utilize a deadlock detection mechanism, which works to limit the number of deadlock-recovery messages. This mechanism helps to avoid false deadlock detection, and to reduce deadlock-recovery overhead. Therefore, our routing algorithms deliver the majority of the packets by fully adaptive routing based on the regularity of k-ary n-cubes utilizing FA-VCs, including non-minimal paths [13]. Namely, the rates of deadlock recovery are controlled to be small. Many routing algorithms are defined by utilizing multiple virtual channels (VCs) for combining adaptive and escape routing [4, 8, 11]. In such routing algorithms, limiting the utilization rate of escape VC resources is critical because messages are not allowed to move from escape VCs to adaptive VCs on irregular networks for deadlock prevention. This restriction is also required for fault-tolerant routing on k-ary n-cubes if we allow faults at any position, because the topological regularity is lost by the faults. So, we believe our technique to select drainage packets independently on each set of VCs is also useful for other adaptive and fault-tolerant routing algorithms.

In this paper, we could show simulation results for two communication patterns with two offered rates on 16-ary 2-cubes. For the other size of k-ary n-cubes, we found that the ud-sel protocol suits a smaller size of networks.
such as the 8-ary 2-cube. This is because it is hard for the small networks to be saturated compared to larger networks, and deadlock is rare. Therefore, reducing the number of drainage packets by ud-sel is useful, and neither throttling nor packet drainage from adaptive VCs is necessary. On the other hand, larger networks such as a 32-ary 2-cube and a 16-ary 3-cube could be saturated with small offered traffic if global communication is performed [12]. In this case, ud-sel/throttling and ud-sel/fa-sel are valuable to reduce network load during network reconfiguration.

Traffic patterns also affect network saturation throughput. Based on our experiments with other types of traffic, we can say that controlling the offered rate in smaller than saturation throughput is important. Otherwise, the deadlock frequency is raised and communication performance is degraded. Injection throttling as well as reducing the network load by draining blocked packets is helpful in congested networks.

6. Conclusion

We have studied the performance impact of network reconfiguration protocols using the fault-tolerant and adaptive routing algorithm Detour-UD. We proposed selective packet drainage by identifying unroutable packets using drain-flags. We also showed the effect of throttling and draining blocked packets during the network reconfiguration. Based on our experimental results, we can conclude the following:

1. The static protocol incurs the highest latency after network reconfiguration because all in-flight packets are drained and retransmitted. Dynamic protocols resolve this problem; however, there is a trade-off between maintaining communication performance and the number of drainage packets.

2. Selective packet drainage is useful to reduce the number of retransmission packets. The packets to be drained can be identified during the network reconfiguration process by finding the unroutable packets based on the restriction of underlying routing functions.

3. During network reconfiguration, temporary deadlock is configured especially in cases of high communication load and serious faults. Drainage of such deadlocked packets utilizing the deadlock-detection function is effective to maintain communication performance.

Our future work is establishing a more efficient self-tuning technique that automatically chooses an adequate communication protocol based on the communication load on the network and the significance of the faults.

Acknowledgments

We thank Prof. Amano of Keio University and his research group for opening the ISIS library to the public. We used it to implement our network simulator.

This research is supported in part by the Grants-in-Aid for Scientific Research of the Japan Society for the Promotion of Science (JSPS) No.15500033 and 17360178.

References


