Reliable Flow Control Coding in Low-Buffer Grid Networks

Shima Bayatian[1]✉, Gholamreza Latif Shabgahi [2]
(1)Computer Engineering Faculty , Islamic Azad University , Arak branch, Arak, Iran.
(2)Computer & Control Dept. Electrical Faculty, Power & Water University of Technology (PWUT), Tehran, Iran.
m_bayatian@yahoo.com; latif_shabgahi@yahoo.co.uk

Received: 2011/05/22; Accepted: 2011/07/15 Pages: 61-69

Abstract

We consider a grid network where nodes contain small buffers. A packet that faces a crowded buffer in its route will get extra latency and may be dropped. In this paper, we propose a novel flow control protocol called RFCC for grid networks. RFCC tries to reroute delayed packets and utilizes network coding to introduce a configurable amount of redundant information in the network, thereby increasing reliability in the face of packet loss. RFCC contains a number of mechanisms to adapt to the traffic model on a grid interconnection network in a multiprocessor system. Our simulation experiments show that RFCC improves reliability with comparable traffic overhead compared to the case in which RFCC is not used.

Keywords: *Grid network, Flow control, Network coding, Routing, Reliability*

1. Introduction

We consider a grid switching network (Figure. 1) where nodes contain small buffers. A packet that faces a crowded buffer in its route will get extra latency and may be dropped. This leads to low reliability in data transfer and we are going to improve reliability by reducing packet loss. We define reliability in a network as ratio of dropped packets over all packets. For example, if reliability is 0.9, it means 90 percent of packets are dropped in the network.

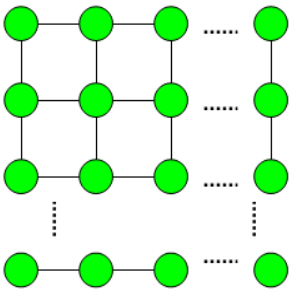

Figure 1. Grid Network Model

If packets are uniformly spread among paths, then congestion and packet loss will be minimized. The problem is that a node can not be aware of the real-time status of buffers in a given path since it needs buffer status information to be continuously broadcasted in the network. In fact, we are going to design a mechanism by which node

forwards packet through a relatively-short non-crowded path only using the information of its neighbors.

There are a number of routing algorithms (such as Dijkstra SPF algorithm [1], and Planar Adaptive Routing algorithm [2]) that discover the network topology to find a route. Then, each node identifies the neighbor to which it has to forward the packet. In this way, packets travel a single path from the source to the destination. As a complement, network coding provides a way in which nodes may transfer packets combined as an encoded frame. In this way, reliability is increased if nodes incorporate a packet into multiple encoded frames.

In this paper, we propose a novel flow control protocol called RFCC for grid networks. RFCC tries to reroute delayed packets and utilizes network coding to introduce a configurable amount of redundant information in the network, thereby increasing reliability in the face of packet loss. RFCC contains a number of mechanisms to adapt to the traffic model on a grid interconnection network in a multiprocessor system.

The rest of this paper is organized as follows. We review related works in Section II. We propose RFCC in Section III. Section IV contains our simulation results, and section V concludes the paper.

2. Related Work

In this section, we review existing researches related to our work.

2.1 Routing in Grid

Routing may be static or dynamic. In static routing, the path between any two nodes is fixed off-line. Once the network is brought up, no changes are made. Such a method is easy to implement and gives fairly good performance in situations where traffic and topology do not change much. The basic static routing algorithm in Grid is the Dijkstra algorithm that finds the shortest path [1].

On the other hand, in dynamic routing the path from one node to another is decided dynamically, taking into account the changes in traffic and network topology. A study of some of the existing algorithms revealed that the major issues confronting a dynamic routing algorithm are [3-6]:

1) Selecting the best path quickly and efficiently.
2) Updating the routing tables with minimum update messages.
3) Counting to infinity problem.
4) Keeping the network "loop free'.

2.2 Flow Control

Authors in [7] consider network control for wireless networks with finite buffers. They investigate the performance of joint flow control, routing, and scheduling algorithms which achieve high network utility and deterministically bounded backlogs inside the network. Their algorithms guarantee that buffers inside the network never overflow.

In [8], authors make a case for a new approach to designing on-chip interconnection networks that eliminates the need for buffers for routing or flow control.

Authors in [9] present elastic buffers, an efficient flow-control scheme that uses the storage already present in pipelined channels in place of explicit input virtual channel buffers. With this approach, the channels themselves act as distributed FIFO buffers.

Authors in [10] propose a new solution combining quality of service routing protocol and flow control mechanism. This routing protocol selects the routes with more resources in an intelligent manner and returns the best route offering a higher transmission rate, a less delay and a more stability.

2.3 Network Coding

Network coding is first introduced in [11]. Basic concepts of network coding are discussed in [12-13].

A distributed scheme called practical network coding is proposed in [14]. It obviates the need for centralized knowledge of the network topology, the encoding/decoding functions, and synchronous data transfer.

Authors in [15] review related works on network coding for distributed storage in wireless networks. Authors in [16] propose an efficient error-recovery scheme that carefully couples network coding and multi-path routing.

Partial network coding (PNC) generalizes the existing network coding paradigm, an elegant solution for ubiquitous data distribution and collection [17]. PNC allows efficient storage replacement for continuous data, which is a deficiency of the conventional network coding. The performance of PNC is quite close to network coding, except for a sub-linear overhead on storage and communications.

Growth-Code [18] is a data encoding and distribution technique. This code is designed to increase the amount of information that can be recovered at a sink node at any point in time, such that the information that can be retrieved from a failing network is increased. The code grows with time.

Sense Code is a collection protocol for sensor network to employ network coding [19]. SenseCode provides a way to gracefully introduce a configurable amount of redundant information in the network, thereby increasing reliability in the face of packet loss.

In [20], priority random linear codes are proposed in a generic network model that encompasses both P2P and sensor networks. A feature of priority random linear codes is the ability to partially recover more important subsets of the original data with higher priorities, when it is not feasible to recover all of them due to node dynamics.

3. The Novel Flow Control Protocol

In this section, we propose a flow control protocol that uses network coding to increase data and path redundancy in grid network. Our proposed protocol is called RFCC (Reliable Flow Control Coding).

3.1 Node Addressing

We assume that node in grid is addressed as (x, y) where x represents row and y represents column of the node. Figure. 2 shows a view of addresses in a network with R rows and L columns.

$$\begin{matrix} (1,1) & (1,2) & (1,3) & ... & (1,L) \\ (2,1) & (2,2) & (2,2) & ... & (2,L) \\ (3,1) & (3,2) & (3,3) & ... & (3,L) \\ ... & ... & ... & ... & ... \\ (R,1) & (R,2) & (R,3) & ... & (R,L) \end{matrix}$$

Figure2. Node addresses in a grid

3.2 Clustering

Neighbor nodes are grouped into clusters. We define a configurable system p on which network rows and columns are divided p has the following properties.

Number of cluster's rows = $\frac{R}{p}$

Number of cluster's columns = $\frac{L}{p}$

Number of cluster's nodes = $\frac{R}{p}.\frac{L}{p}$

Figure. 3 shows a view of clustering in RFCC. The method is to place as many as R/p neighbor rows in the same cluster. Therefore, we take $\lfloor R/p \rfloor$ (the floor of R/p) as number of rows in each cluster. In the same way, we take $\lfloor L/p \rfloor$ as number of columns in each cluster.

$$\begin{bmatrix} (1,1) & (1,2) & ... & (1,\lfloor L/p \rfloor) \\ (2,1) & (2,2) & ... & (2,\lfloor L/p \rfloor) \\ ... & ... & ... & ... \\ (\lfloor R/p \rfloor,1) & (\lfloor R/p \rfloor,2) & ... & (\lfloor R/p \rfloor,\lfloor L/p \rfloor) \end{bmatrix} \begin{bmatrix} (1,\lfloor L/p \rfloor+1) & (1,\lfloor L/p \rfloor+2) & ... & (1,2\lfloor L/p \rfloor) \\ (2,\lfloor L/p \rfloor+1) & (2,\lfloor L/p \rfloor+2) & ... & (2,2\lfloor L/p \rfloor) \\ ... & ... & ... & ... \\ (\lfloor R/p \rfloor,\lfloor L/p \rfloor+1) & (\lfloor R/p \rfloor,\lfloor L/p \rfloor+2) & ... & (\lfloor R/p \rfloor,2\lfloor L/p \rfloor) \end{bmatrix} ...$$

$$\begin{bmatrix} (\lfloor R/p \rfloor+1,1) & (\lfloor R/p \rfloor+1,2) & ... & (\lfloor R/p \rfloor+1,\lfloor L/p \rfloor) \\ (\lfloor R/p \rfloor+2,1) & (\lfloor R/p \rfloor+2,2) & ... & (\lfloor R/p \rfloor+2,\lfloor L/p \rfloor) \\ ... & ... & ... & ... \\ (2\lfloor R/p \rfloor,1) & (2\lfloor R/p \rfloor,2) & ... & (2\lfloor R/p \rfloor,\lfloor L/p \rfloor) \end{bmatrix} ...$$

$$... \quad ... \quad ... \quad ...$$

Figure3. Cluster Scheme in RFCC

Figure. 4 shows how clusters are addressed in RFCC according to their location on the grid.

$$\begin{matrix} (1,1) & (1,2) & (1,3) & ... & (1,L/p) \\ (2,1) & (2,2) & (2,2) & ... & (2,L/p) \\ (3,1) & (3,2) & (3,3) & ... & (3,L/p) \\ ... & ... & ... & ... & ... \\ (R/p,1) & (R/p,2) & (R/p,3) & ... & (R/p,L/p) \end{matrix}$$

Figure4. Cluster addresses

For a node with address (x,y), we can find its cluster head's address (x_c, y_c) as follows.

$x_c = \frac{x}{R/p}$ And $y_c = \frac{y}{L/p}$

3.3 Coding System

We use the RLC coding scheme in RFCC [21]. Given N packets $x_1,...x_N$, RLC creates encoded frame c_i according to the following formula as a linear combination of the N packets.

$$c_i = \sum_{j=1}^{N} \beta_j . x_j$$

Where $\beta_1,...,\beta_N$ are coding coefficients randomly selected from a Galva field. These coefficients are constant for all encoded frames in RFCC. This removes the requirement to send them from the source node to the destination node.

3.4 Coding Process

A node that wants to encode a packet sends it to its clusterhead.
The clusterhead encodes the packets received from the cluster nodes into the least number of encoded frames.
The maximum number of packets which can be combined into one encoded frame is a system parameter and is denoted by N_c.

The destination address of an encoded frame is set as the address of the destination cluster.

3.5 Routing

If a node has a packet to send to a destination, it runs the routing algorithm to find a path to the destination. The path is embedded into the packet and then the packet is forwarded to the next node in the path.

When a node receives a packet, it forwards the packet to the next hop according to the embedded path.

When a packet is in node n_i, if the next hop n_{i+1} is failed or its buffer is full, n_i follows these steps.

1. n_i runs the routing algorithm to find a path to the destination disjoint from n_{i+1}. The new path is embedded into the packet and then the packet is forwarded to the next node in the new path.
2. n_i Sends a copy of the packet to its cluster head for coding.

When a clusterhead creates an encoded frame, it sets the source address as its address and it sets the destination address as the address of the clusterhead of the destination cluster. Then, it runs the routing algorithm to find a path to the destination clusterhead. The path is embedded into the frame and then it is forwarded to the next node in the path.

3.6 Packet Recovery

If a main packet correctly reaches the destination node, nothing will be done to recover it. Otherwise, the destination waits for some time. If the destination does not receive the packet in this time, it requests the clusterhead to recover the packet. Then, the clusterhead delivers the packet to the destination after recovery. RFCC distributes

the decoding work among nodes in the cluster.

We assume that the requested packet x_k is embedded in encoded frame c_i along with as many as N-1 packets $\{x_j \mid j \neq k, j = 1,2,...,N\}$. To recover x_k, the cluster head has to obtain the other packets inside c_i. According to the following steps, the clusterhead makes use of the other nodes to recover x_k.

1. The clusterhead sends an empty frame $c_i^{'}$ to a node in the cluster governing packet $\{x_j \mid j \neq k, x_j \notin c_i^{'}\}$.
2. When a node receives $c_i^{'}$, for each packet $\{x_j \mid x_j \in c_i, x_j \notin c_i^{'}\}$ which it governs, the node adds the packet to $c_i^{'}$ as below:
   $c_i^{'} = c_i^{'} + \beta_j x_j$
3. If all the N-1 packets $\{x_j \mid j \neq k, j = 1,2,...,N\}$ are added to $c_i^{'}$, then the node sends $c_i^{'}$ to the cluster head. Otherwise, it sends $c_i^{'}$ to a node in the cluster governing packet $\{x_j \mid j \neq k, x_j \notin c_i^{'}\}$.
4. Steps 2 and 3 are repeated until all the N-1 packets $\{x_j \mid j \neq k, j = 1,2,...,N\}$ are added to $c_i^{'}$.
5. When the cluster head receives $c_i^{'}$, it recovers x_k according to the following.
   $c_i = c_i^{'} + \beta_k x_k$

3.7 Packet Drop

Each packet has an Expiration Time. If the Expiration Time is passed, the node removes the packet from its buffer. Each packet also has a TTL field. If the packet passes a node, the TTL field of the packet is reduced by one. If the TTL field reaches zero at a node, the node removes the packet from its buffer. These two fields prevent the packet from staying too long in node buffers.

3.8 Packet Retransmission

If the clusterhead is not able to recover a packet, it informs the destination node. Then, the destination node requests the source to retransmit the packet. Retransmission is the final solution to recover the packet.

4. Simulation

We implemented the RFCC protocol in the NS2 network simulator [22]. In this section, we evaluate the performance and the overhead of RFCC when using the Dijkstra routing algorithm [1].

To evaluate RFCC, we define a simulation scenario presented in Table 1. In this scenario, we use different values for packet generation rate in different executions whereas the other parameters are fixed to evaluate the performance of RFCC under different traffic loads.

Table1. Simulation Scenario

Parameter	Value
Number of Nodes	400
Network Topology	Grid 20x20
TTL	20
Network Traffic	A Pareto-On/Off flow between every node pair
Average Bit Rate per Traffic Flow	10 Kbps
The Idle time Period per Traffic flow	5 seconds
Packet Expiration Duration	3 seconds
Routing Algorithm	Dijkstra
Node's Buffer Capacity	20 packets
Node Switching Delay	30 ms
Link Bandwidth	1 Mbps
Network Traffic Load	variable from 0.7 to 1.0
p	8
N_c	5
Simulation Duration	1 hour

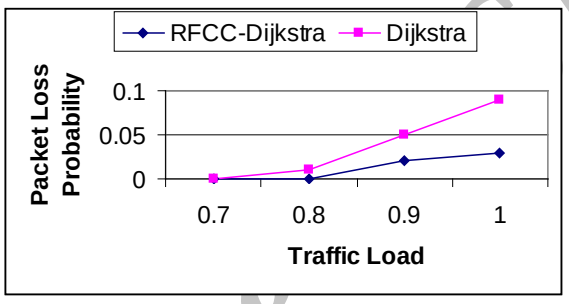

Figure5. Packet loss ratio versus traffic load

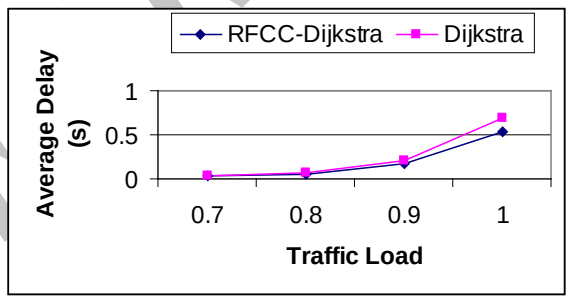

Figure6. Average packet delay versus traffic load

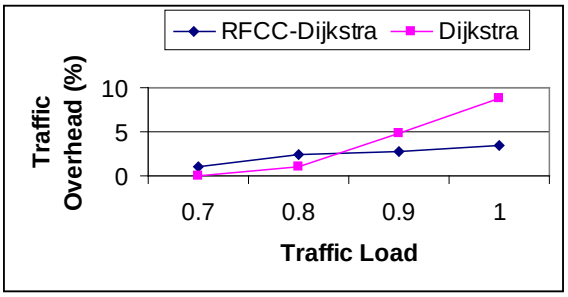

Figure7. Traffic overhead versus traffic load

4.1 Simulation Results

Figure. 5 shows RFCC's ability in correctly delivering packets to destinations in different traffic conditions. With increase in traffic load, node buffers are filled and then some packets deviate from their original path and may be dropped. RFCC in this experiment reduces packet loss in average by 2.5 times. When the network does not use RFCC, the reliability goes down to 0.91 whereas the reliability is bounded by 0.97 when the network uses RFCC.

Figure. 6 shows average packet delay in our experiment. Delay in our network model is the result of increase in network traffic and congestion in buffers. When there is no flow control protocol, packets have to wait behind long queues and get high delays. When RFCC is used, bottom-line packets are rerouted and get smaller delays. In this experiment, RFCC reduces average packet delay as much as 1.2 times.

Figure. 7 shows RFCC reduces packet retransmission from sources to destinations. Thus, it reduces the traffic overhead in the network. In the 100% traffic load, RFCC reduces the traffic overhead in the network as much as 2.51 times.

5. Conclusions

We propose a novel flow control protocol called RFCC in grid networks. A packet is simply forwarded in RFCC if it faces neither a full-buffer node nor a faulty node. When the packet faces such a node, it is forwarded on another route toward the destination and a copy of it is sent toward the destination embedded in an encoded frame. The network is clustered. In this way, network coding adapts to the traffic model of a multiprocessor interconnection network. In our simulation experiments, we have

- RFCC reduces packet loss in average by 2.5 times.
- Reliability is bounded by 0.97 when the network uses RFCC.
- RFCC reduces average packet delay as much as 1.2 times.

In the 100% traffic load, RFCC reduces the traffic overhead in the network as much as 2.51 times.

6. References

[1] R. Bhandari, "Survivable Networks: Algorithms for Diverse Routing," in Kluwer, 1999.
[2] A. A. Chien, and J. Kim, "Planar-Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors", J. ACM 42(1), pp.91-123, 1995.
[3] B. Awerbuch and S. Even,"Reliable Broadcast Protocol in Unreliable Networks," Networks, Vol. 16, 1986, pp. 381-396.
[4] J. J. Garcia,"A Minimum Hop Routing Algorithm Based on Distributed Information," Computer Networks and ISDN Systems, Vol. 16, 1989, pp. 367-382.
[5] E. N. Gilbert,"A Solvable Routing Problem," Networks, Vol. 19, pp. 587-594, 1989.
[6] I. M. Jaffe, A. E. Baratz and A. Segall,"Subtle Design Issues in the Implementation of Distributed, Dynamic Routing Algorithms," Computer Networks and ISDN Systems, Vol. 12, pp. 147-158, 1986.
[7] L. Le, E. Modiano, and N. Shroff, "Optimal Control of Wireless Networks with Finite Buffers", In Proceedings of INFOCOM', pp.2034~2042, 2010.
[8] T. Moscibroda, and O. Mutlu, "A Case for Bufferless Routing in On-Chip Networks", Proceedings of the 36th annual international symposium on Computer architecture, 2009.
[9] G. Michelogiannakis, J. Balfour, and W. Dally, "Elastic-buffer Flow Control for on-chip networks", In Proceedings of HPCA'09, pp.151-162, 2009.
[10] M. Belkadi, M. Lalam, A. Mzoughi, N. Tamani, M. Daoui, and R. Aoudjit, "Intelligent Routing and Flow Control In MANETs", journal of Computing and Information Technology 18(3), 2010.

[11] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow", IEEE Trans. Inform. Theory, vol.46, no.4, pp.1204–1216, July 2000.
[12] C. Fragouli, and E. Soljanin, "Network Coding Fundamentals", Foundations and Trends in Networking, Vol.2, No.1, pp.1–133, 2007.
[13] C. Fragouli, J. Widmer, and J. Boudec, "Network Coding: An Instant Primer", ACM SIGCOMM Computer Communication Review, vol.36, no.1, pp.63–68, 2006.
[14] P. A. Chou, Y. Wu, and K. Jain, "Practical Network Coding", Proc. of 51st Allerton Conf. Communication, Control and Computing, Oct. 2003.
[15] A.G. Dimakis, and K. Ramchandran, Network Coding for Distributed Storage in Wireless Networks, Chapter in edited volume, Networked Sensing Information and Control, Signals and Communication Series, V. Saligrama (Ed.) Springer Verlag, 2007.
[16] Z. Guo, B. Wang, and J.-H. Cui, "Efficient Error Recovery Using Network Coding in Underwater Sensor Networks", Proc. of IFIP Networking'07, pp.227-238, 2007.
[17] D. Wang, Q. Zhang, and J. Liu, "Partial Network Coding: Theory and Application in Continuous Sensor Data Collection", Proc. of. IEEE IWQoS'06, June 2006.
[18] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth Codes: Maximizing Sensor Network Data Persistence", Proc. of ACM SIGCOMM'06, pp.255-266, 2006.
[19] L. Keller, E. Atsan, K. Argyraki, and C. Fragouli, "SenseCode: Network coding for reliable sensor networks", EPFL Technical Report, October 2009.
[20] Y. Lin, B. Li, and B. Liang, "Differentiated Data Persistence with Priority Random Linear Codes", Proc. of 27th IEEE Int'l Conf. Distributed Computing Systems (ICDCS), 2007.
[21] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting", Proc. of ISIT'03, July 2003.
[22] Information Sciences Institute, NS-2 network simulator, Software Packet, 2003, http://www.isi.edu/nsnam/ns/.