An Adaptive Software-based Deadlock Recovery Technique

M. Mirza-Aghatabar[2], A. Tavakkol[1,2], H. Sarbazi-Azad[1,2], A. Nayebi[2]
[1]*IPM School of Computer Science, Tehran, Iran*
[2]*Department of Computer Eng. of Sharif Uni. of Technology, Tehran, Iran*
aghatabr@ce.sharif.edu, tavakkol@ce.sharif.edu, azad@ipm.ir, nayebi@ce.sharif.edu

Abstract

Deadlock management has a direct effect on making a reliable connection between processing nodes in parallel computers. Networks using wormhole switching are the most vulnerable networks to deadlock occurrence due to chained blocking nature of this switching method. Different hardware based techniques for deadlock recovery were proposed in the literature which have considerable design complexity, while deadlock occurrence in a network is rare. A software based technique can reduce this cost while preserving performance. The only software based technique proposed in the literature is static and independent of network workload and working conditions. In this paper we present an adaptive software based technique for deadlock recovery, and validate its performance in the presence of different traffic patterns including uniform, hot spot, local and first matrix transpose (FMT) patterns in 8-ary 3-cube network (torus). Simulation results exhibit about 21%, 20%, 98% and 20% performance improvement under local, FMT, hotspot and uniform traffic patterns, respectively.

1. Introduction

Nowadays, lots of products, such as cell phones and portable computers are implemented on a silicon chip [2]. The number of transistors per chip will increase beyond billions, due to technology scaling, at the end of this decade [1]. Therefore, new methods should be employed to manage such huge number of transistors on a chip. The **N**etwork-**o**n-**C**hip (NoC) interconnecting different IPs and parts in a **S**ystem-**o**n-**C**hip (SoC) is the main potential implementation approach to implement such a huge chip. NoCs can make SoCs structured, reusable, scalable, and high performance [3]. Many topologies with different capabilities have been proposed for NoCs including the mesh [6], torus [5], Octagon [9], SPIN [10], BFT [11].

The routing algorithm is one of the most important issues for NoC designers which determines the path from the source to destination node. Routing algorithms can be classified according to their adaptivity level. A routing algorithm can be either *deterministic* or *adaptive*. Adaptive routing algorithms use the information about network traffic and channel status to avoid congested or faulty regions of the network [6]. Among different routing algorithms, the FAR[1] algorithm has the best throughput with the cost of using more virtual channels to avoid deadlock occurrence; as an exemplification, FAR routing for the torus network requires at least 3 virtual channels per physical channel to be deadlock-free [6].

Switching method is another important issue for NoC designers and defines the way in which messages visit and use intermediate routers, buffers, and switches. Different switching methods are introduced in the literature to improve the performance of the network. Wormhole [10] and virtual cut-through [11] have become the most widely used switching techniques for multi-computers and distributed shared-memory multiprocessors, due to their lower buffering requirements and better usage of network resources. Moreover, these switching methods provide high data transmission rate as a result of their pipeline nature, but this property makes wormhole susceptible to deadlock occurrence and increment of blocking time. Deadlock is a state of the network where no messages can advance along its path in the network because each message requires a channel occupied by another message [6].

There are two main approaches to resolve deadlock problem [6]: *Deadlock Avoidance* techniques where resources are granted to packets only if the resulting global state is safe (deadlock free); and *Deadlock Recovery* techniques where resources are granted to packets without any check and therefore, some detection mechanism must be provided. If deadlock is

[1] Fully Adaptive Routing

detected, some resources are de-allocated and granted to other packets in order to break cyclic dependency.

Deadlock avoidance based routing algorithms do not use resources efficiently compared to the deadlock recovery based ones. Also it was shown that deadlocks rarely occur when sufficient routing freedom is provided, but they are more likely to occur when the network is close to or beyond saturation [12]. Thus, the cost paid to avoid deadlocks is usually more than achieved performance especially when the network is not close to saturation region (note that this is usually the case because a network should approach saturation region). To illustrate, authors in [21-23] have shown that deadlock recovery based routing algorithms can be used to eliminate the extra hardware cost and gain a considerable cost-performance. In fact, deadlock recovery based routing algorithms have the highest adaptivity for routing and can use all available virtual channels associated to each physical channel to route messages, and thus, in some papers they are called TFAR[2].

In this paper we will introduce a new Adaptive soFtware-Based deAdlock Recovery (AFBAR) technique for TFAR in wormhole switched networks. In section 2, related works will be reviewed. Section 3 provides details of the AFBAR technique. Results of performance evaluation will be proposed in section 4. Finally in Section 5, some concluding remarks are given.

2. Related Works

Deadlock recovery routing algorithms can be classified into two groups: hardware-based and software-based algorithms. First we describe two hardware-based algorithms and then a software-based routing algorithm will be described.

One deadlock recovery-based algorithm, *Compressionless Routing (CR)* algorithm, was proposed in [14]. Let d_{SD} be the distance between source and destination nodes. The source node can determine whether that the header flit has reached the destination node or not: When the number of transmitted flits in the source node is equal to d_{SD}, it means that the header flit has reached the destination, otherwise maybe deadlock has occurred. In CR, the source node keeps track of the injected message and detects if it has reached the destination node or not. If the message length is shorter than d_{SD}, then stuffing flits will be appended at the end of the message in the source node to determine that the header flit has reached to destination or not. If it has reached, no deadlocks can happen. If not, and the message is blocked for some time, the source node breaks down the partial message path, kills the deadlocked message, and then tries sending it again later. So the CR uses a regressive technique.

Another deadlock recovery scheme, called DISHA, was proposed in [15]. This method was proposed to tackle the disadvantages of CR, i.e. (1) increasing the message length to more than its actual size which decreases effective channel utilization -when the packet size is small, compared to the network diameter, or the buffers at routers are deep, the overhead due to padding is large; (2) Killing deadlocked messages and re-injecting them increases message latency. DISHA provides a central buffer for each node that is not dedicated to any specific channel and calls it *Deadlock Buffer (DB)*. The collection of these DBs in the nodes creates a new floating network which can be used to route the deadlocked messages. Each input virtual channel is equipped with a counter. When a message is blocked for more than a threshold time, it is assumed the message is in a potential deadlock situation. Now the router copies the deadlocked message into its DB and routs it via in the floating network containing the DBs of intermediate routers to the destination node. When a router detects a deadlocked message in its DB, preempts the needed output channel to route that message and after routing the deadlocked message, the channel can be used by non-deadlocked messages. It means deadlocked messages in DBs, have a higher priority in preempting the output channels for routing. As DISHA does not kill the deadlocked messages, it is a progressive technique. Progressive deadlock recovery-based techniques usually increase performance in comparison with deadlock avoidance-based techniques, because they require less dedicated resources to handle deadlocks [15].

Martinez et al., in [13], introduced a software-based routing technique for deadlock recovery routing algorithms. They proposed a new progressive approach to handle deadlocks. It uses message injection limitation mechanism [16] used to prevent performance degradation near the saturation point and especially reduce the probability of deadlock occurrence under TFAR. Moreover, in this method a counter is associated with each output physical channel. This counter is incremented in each cycle and is reset when a flit is transmitted across the physical channel. When the value of counter exceeds a predetermined threshold, an inactivity flag of that physical channel is set. Every time a message is routed, if all the feasible virtual output channels are busy, then the inactivity flags associated with the corresponding

[2] True Fully Adaptive Routing

physical output channels are checked. If all of these flags are set, then there is no activity through any of the feasible physical output channels, and the message is presumed to be involved in a deadlock. Thus the message is taken out from the network by the current Processing Element (PE). If the software messaging layer detects a message with the destination address different from that of the current node, it must re-inject that message to the network at a later time.

The proposed technique in [13]: (1) requires a very small amount of hardware due to no dedicated buffer resources to handle deadlocks, (2) eliminates performance degradation at saturation point with message injection limitation, (3) reduces the frequency of deadlock, and (4) uses a new deadlock detection technique which considerably reduces the probability of false deadlock detection. The performance of mentioned technique is better than CR, and is about that of the progressive deadlock recovery techniques (like DISHA***),*** assuming that both of them use the same injection limitation and deadlock detection mechanisms and that the additional router complexity required in DISHA does not impact clock frequency [13].

Although the last technique has the mentioned advantages, but using a fixed and static threshold value for deadlock detection for each node and each output physical channel is not fair, since under non-uniform traffic patterns the blocking time of packets and utilization of each physical channel depend on the position of each node in the network topology. The fixed value of threshold may change with regard to traffic rate around a node for each physical channel. As an exemplification, under hotspot traffic pattern, traffic rate around the hot node is high, so the blocking probability and blocking duration is high. But it does not mean all of these blocked messages are engaged in a deadlock; consequently, with a fixed value of threshold, many blocked messages will be detected as deadlock messages. In fact the best threshold value for each node depends on the traffic rate around its channels. Thus, a node with high traffic around itself should have more threshold value. In this paper we have implemented the message injection limitation technique but we have used a function to dynamically determine the threshold value for each physical channel in each node in order to detect deadlocks. Our results depict a lower number of detected deadlocks under different traffic patterns, especially under hotspot traffic pattern and exhibit a better performance in spite of using a lower number of virtual channels.

3. AFBAR technique

Different applications, during execution, exert different traffic patterns to the network. As a consequence, a predetermined and fixed value of threshold for each node (each physical channel of node) can degrade the performance of the network by increasing the number of mistakenly detected deadlocks. When the traffic rate increases over the channels of a typical node, the blocking time of messages will increase, so the threshold value for these physical channels should increase to avoid mistakes in deadlock detection. Figure 1 shows the deadlock detection reduction using AFBAR in comparison with the described technique of [13]. The implementation details of AFBAR will be described later in this section. To simplify the explanation process we assume that the network a is k-ary n-cube.

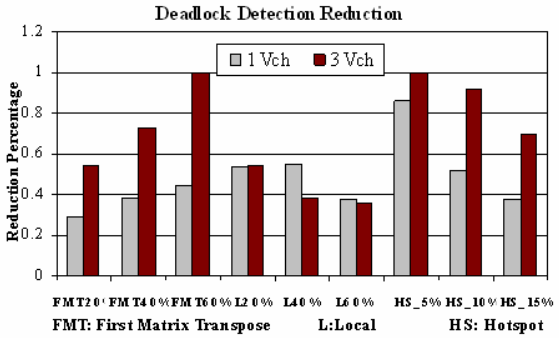

Figure 1: Deadlock Detection Reduction

AFBAR is An **A**daptive so**F**tware-**B**ased de**A**dlock **R**ecovery technique which uses local traffic information of each node to dynamically set a threshold value for each physical channel for deadlock detection during the execution. Like [13] when an intermediate router detects a message to be engaged in deadlock, sends the message to the current PE and the software messaging layer re-injects the message into the network later. Unlike [13] which uses a static re-injection time, AFBAR dynamically tunes this value according to local information and average utilization of the channels.

AFBAR continuously monitors each physical channel of nodes to determine the traffic rate around nodes. Then, it updates the threshold value for each physical channel adaptively after a period of time (PT). Here this period is set to be eight times of message length. We define a channel utilization parameter for channel monitoring in a period of time as:

$$U_{total} = \sum_{i=0}^{2n} U_i \qquad (1)$$

$$\text{Utilization Ratio}_i = \frac{U_i}{U_{total}} \quad (2)$$

where n is the network dimension of the k-ary n-cube and U_i is the number of transmitted flits per physical channel.

AFBAR dedicates another counter (instead of the counter used for deadlock detection) for each physical channel that is incremented by one when a flit is transmitted on the corresponding physical channel. Then, at the end of PT, using equations (1) and (2) channel utilization for each physical channel is calculated. It is clear that the channel with highest utilization rate sees the most traffic rate around and its threshold value must be increased.

Generally, when a node has n bidirectional physical channels and the traffic is distributed uniformly in the network, then the traffic rate of each physical channel is a portion of the total traffic of the mentioned node. This ratio can be estimated as follows:

$$R_0 = \frac{1}{n} \quad (3)$$

Thus, in a k-ary 2-cube topology where each node has 4 bidirectional channels, we have: $R_0 = 0.25$

Now, using the deviation from the uniform ratio (R_0), AFBAR updates the next threshold value to tune it according to the traffic rate of each physical channel and when the deviation is negligible or zero, the threshold (like [13]) is set to four times of message length. AFBAR uses the following equations to update the threshold value for each physical channel:

$$R_i' = \frac{CurrentRatio_i - R_0}{R_0} \Rightarrow R_i' = \begin{cases} R_i' & , \quad -0.5 < R_i' < 1 \\ 1 & , \quad 1 < R_i' \\ -0.5 & , \quad R_i' < -0.5 \end{cases} \quad (4)$$

$$T_i = \begin{cases} 4 \times messageLength \times (1 + 2 \times R_i') & , \quad 0 < R_i' \\ 4 \times messageLength \times (1 + R_i') & , \quad R_i' < 0 \end{cases} \quad (5)$$

As mentioned before, another way that leads to more adaptivity is the use of dynamically tuned re-injection time. In [13] this factor has been assume to be constant (200 cycles), but we will dynamically change it using equations (6, 7, 8).

$$T_{re-inject} = \alpha \times messageLen\ gth \quad (6)$$

$$\overline{U} = \frac{U_{total}}{n \times V} \quad (7)$$

$$\alpha = \begin{cases} 1 & \overline{U} < 0.1 \\ 2 & 0.1 < \overline{U} < 0.2 \\ 6 & 0.2 < \overline{U} < 0.9 \\ 12 & 0.4 < \overline{U} \end{cases} \quad (8)$$

Where V is the number of virtual channels per physical channel.

4. Experimental Results

In this section, we will evaluate AFBAR under different traffic loads and compare its performance with software-based deadlock recovery (called STFAR) and FAR algorithms. The message length is 32 flits and as aforementioned our network is 8-ary 3-cube. Figure 2 shows the performance comparison of these algorithms under different traffic loads including: *Local*, *Uniform*, *HotSpot* and *First Matrix Transpose* (FMT). The best performance under all traffic patterns is achieved by AFBAR with 3 virtual channels. Although the performance of FAR is better than AFBAR with 1 virtual channel, but the main constraint of FAR is that it needs at least 3 virtual channels to be deadlock free. This requirement consumes more power in comparison with AFBAR using 1 virtual channel.

As can be seen if Figure 2, the performance gap between STFAR and AFBAR with 3 virtual channels under uniform traffic pattern is small, hence we can use a static value for T_i like [21]. However, the uniform traffic load is not realistic, and different applications usually exert non-uniform loads.

Table 1. Comparison of the number of detected deadlocks

Traffic Patterns	Rate	Num. of Detected Deadlock			
		1Virtuall Channel		3Virtuall Channel	
		STFAR	AFBAR	STFAR	AFBAR
Uniform	---	161	82	0	0
FMT	*20%*	226	161	0	0
	40%	144	89	0	0
	60%	121	67	2	0
Local	*20%*	903	419	618	282
	40%	2407	1095	1596	984
	60%	5709	3547	3124	2002
Hotspot	*5%*	208	29	5	0
	10%	669	324	414	35
	15%	810	505	541	162

Another important metric that is directly related to the performance is the number of detected deadlocks. Table 1 shows the number of detected deadlocks using AFBAR and STFAR routing algorithms. It is clear that the number of detected deadlocks with AFBAR is lower than STFAR which can lead to a better performance.

There are two important points about the local traffic load:

1) Under local traffic load (with a locality diameter of 1) lots of messages are sent to the neighbors, and such high level of locality in the traffic increases the probability of deadlock creation. It is clear from Table 1 that the higher number of detected deadlocks is

resulted for this traffic pattern which can lead to performance degradation.

2) The short distance between the source and destination nodes under local traffic model results in a lower network latency and thus performance improvement. Figure 3 reveals this fact.

Table 2. Performance Improvement

Traffic Patterns	Performance Improvement	
Number of Virtual Channels	1 Virtual Channel	3 Virtual Channels
Uniform	20%	4%
FMT	20.5%	10%
Local	21%	4.8%
Hot Spot	20.5%	97.7%

Table 2 shows the performance improvement of the proposed routing algorithm (AFBAR) in comparison with STFAR under different traffic loads. The best performance improvement is achieved under hotspot traffic loads, due to the pipeline nature of wormhole switching and high blocking time under this type of traffic loads as mentioned before.

5. Conclusions

Deadlock management is essential for providing a reliable communication path between processing elements in interconnection networks of parallel computers. Among different switching methods for interconnection networks, wormhole switching is the most vulnerable to deadlock occurrence due to its pipeline nature. So some mechanisms should be used to resolve deadlocks.

In this paper, we focused on the software-based deadlock recovery method (STFAR) which has been described in [13]. The deadlock detection mechanism used in [13] has a static nature to detect deadlocks at each node. Different traffic patterns generated by different applications require different treatment and thus using a static value in deadlock detection mechanism is not suitable. Our results showed the poor performance of STFAR under different traffic patterns and proposed an adaptive deadlock recovery method called AFBAR, which improved the performance under all traffic patterns. The performance improvement was ranged from 20-97% for local, FMT, HotSpot and Uniform traffic patterns.

6. References

[1] A. Allen, D. Edenfeld, W.H. Joyner, A.B. Kahng, M. Rodgers, and Y. Zorian, "2001 Technology Roadmap for Semiconductors," *IEEE Computer,* pp. 42-53, 2002.

[2] S. Kumar, A. Jantsch, J.P. Sonion, M. Forsell, M. Millberg, J. Oeberg, K. Tiensirja, and A. Hemani, "A Network on Chip Architecture and Design Methodology," *IEEE Design and Test of Computers,* no.4, vol.16, pp.6-15, 1999.

[3] L. Benini and G. de Micheli, "Networks-on-Chip: A new Paradigm for System on Chip Design," *Design Automation and Test in Europe, IEEE computer*, vol. 35, no. 1, pp. 70-78, 2002.

[4] W.J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Proc. of Design Automation Conference*, pp. 683-689, 2001.

[5] J. Duato, S. Yalamanchili, and L. Ni, "Interconnection Networks—An Engineering Approach," Morgan Kaufmann, 2002.

[6] F. Karim, A. Nguyen, and S. Dey, "An Interconnect Architecture for Networking Systems on Chip," *IEEE Micro*, vol. 22, no. 5, pp 36–45, 2002.

[7] Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino, "Spin: A scalable, Packet Switched, on Chip Micro-Network," *DATE 03 Embedded Software Forum*, pp. 70–73, 2003.

[8] Guerrier and A. Greiner, "A Generic Architecture for on Chip Packet-Switched Interconnections," *IEEE Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pp. 250-256, 2000.

[9] D. Rahmati, A. E. Kiasari, S. Hessabi and H. Sarbazi-Azad, "A Performance and Power Analysis of WK-Recursive and Mesh Networks for Network-on-Chips," *Proceedings of the 24th International Conference on Computer Design (ICCD)*, 2006.

[10] W. J. Dally and C. L. Seitz, "Deadlock-free messagerouting in multiprocessor interconnection networks," *IEEE Trans. on Computers*, vol. C-36, no. 5, pp. 547-553, 1987.

[11] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," Computer Networks. Vol. 3, pp. 267-286, 1979.

[12] T.M. Pinkston and S. Warnakulasuriya, "On deadlocks in interconnection networks," *Proc. of the 24th International Symposium on Computer Architecture*, 1997.

[13] J.M. Martinez, P. Lopez, J. Duato, and T.M. Pinkston, "Software-based deadlock recovery techniques for true fully adaptive routing in wormhole networks," *Proc. of International Conference on Parallel Processing*, 1997.

[14] J.H. Kim, Z. Liu, and A.A. Chien, "Compressionless routing: A framework for adaptive and fault-tolerant routing," *Proc. of the 21st International Symposium on Computer Architecture*, pp. 289-300, 1994.

[15] Anjan K.V. and T.M. Pinkston, "An efficient fully adaptive deadlock recovery scheme: Disha," *Proc. of 22nd International Symposium on Computer Architecture*, pp. 201-210, 1995.

[16] J. Duato, "Improving the efficiency of virtual channels with time-dependent selection functions," *Future Generation Computer Systems*, no. 10, pp. 45-58, 1994.

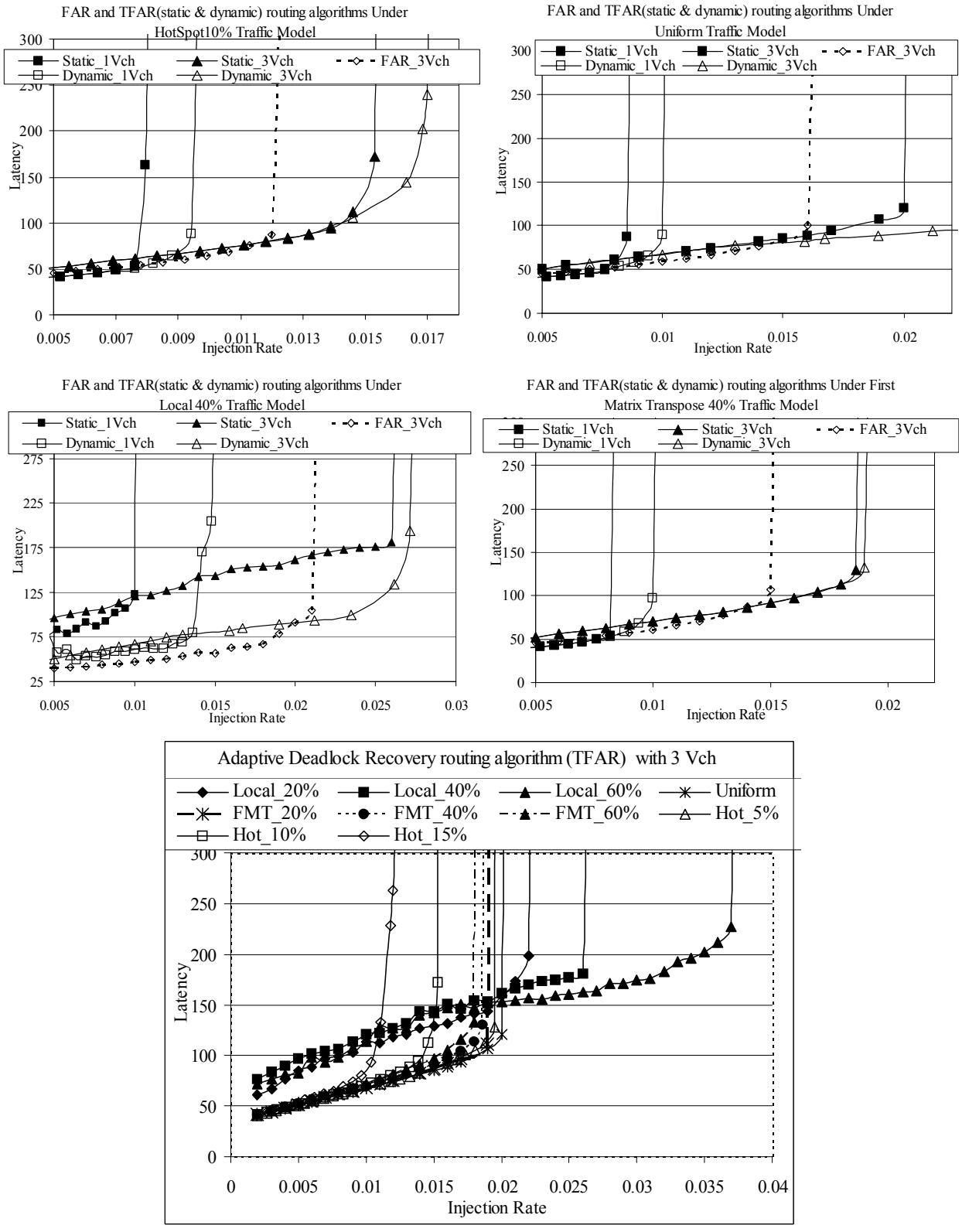

Figure 3. Performance comparison of TFAR under different traffic models