Intelligent congestion control for NoC architecture in Gem5 simulator

Authors:

Mushtaq Ahmed Shaikh, R V College of Engineering, Bangalore, India

Smriti Srivastava, R V College of Engineering, Bangalore, India

Shivaneetha G, R V College of Engineering, Bangalore, India

Dr. Minal Moharir, R V College of Engineering, Bangalore, India

Outline

- About the Author
- Introduction
- Literature Survey
- Design Methodology
- Experimental Results and Analysis
- Conclusion
- References

Introduction

- On-chip communication facilitates the integration of multiple modules onto a single chip called the System-on-Chip (SoC).
- Network on Chip (NoC) became the dominant and cost-effective method for data transmission in an SoC.
- The NoC is a structured architecture for network-based communication subsystems on an integrated chip. It consists of a topology with several nodes arranged in a pattern.
- However, NoC architecture has latency and power consumption concerns when a data packet is transmitted to distant nodes.
- Moreover, in a heavy-traffic network, which causes congestion at busy nodes, the network needs to learn the traffic information dynamically to avoid possible congestion.
- Machine Learning (ML) algorithms proves to be helpful by enabling the network to understand the traffic and provide an optimal path.

Introduction

- This work presents a Reinforcement Learning(RL) based congestion-aware Q-routing algorithm.
- The routing method employs a Q-learning (Quality-learning), a value-based RL algorithm that uses a Q-table to maintain Q-values.
- The Q values are estimated to observe the results when a specific action is performed.
- The proposed work uses a Q-learning-based adaptive congestion-aware routing algorithm called Q-Routing that is designed to keep track of local and global congestion.
- The Q-values that are available in the Q-table guide a node to forward a data packet along the most optimal route.
- This proposed methodology aims to improve the network's performance by reducing the average packet latency and power and energy consumption.

Literature review

Author	Year	Work Done
D. Radha, Mushtaq Shaikh Vamsi Silla, Minal Moharir	**2021**	The paper proposed a workload allocation methodology in an NoC-based many-core processor. The application-aware allocation works by making the packets communicate with a single quadrant, reducing the hop count for the transmission. It is compared with the random allocation using SPEC CPU 2006 benchmark suite on the GEM5 simulation tool.
Smriti Srivastava, Adithi Viswanath, Krithika Venkatesh, Minal Moharir	**2022**	The author proposed an adaptive routing algorithm for a multicasting network in a wireless Network-on-chip. The placement of the wireless hubs is made such that combined together, they can reach every node optimally. The methodology is based on the TDMA (Time Division Multiple Access) protocols which showed lesser power consumption than the traditional FDMA (Frequency Division Multiple Access).
F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg and J. Plosila	**2020**	The paper tests a Q-learning-based congestion-aware routing (QCA) in an Omnet++-based NoC simulator. This algorithm alleviates congestion by a reliable estimation of the traffic status of the network. Still, the latency is high compared to the traditional routing algorithms when there is no congestion. Thus, an efficient routing policy is invoked for corresponding network states.

Literature review

Author	Year	Work Done
Wang, K. Louri, A	**2020**	This paper implements the deep RL method, CURE, to reduce the end-to-end delay and improve energy efficiency. Each router is applied with fault-secure adaptive error correction to enhance the model's reliability. The bypass design and router power gating method extend chip lifespan and reduce power consumption.
Zheng, H. and Louri, A	**2020**	This paper proposed an error tolerance framework of quality control and a data approximation method called Lightweight lossy compression to reduce the packet size, network latency, and power consumption. The process identifies resilient error variables during transmission and determines the error based on quality. The developed method reduces the latency but increases the packet loss in the network.
Lee, S.C. and Han, T.H	**2020**	The authors applied a Q-learning-based thermal and traffic-aware adaptive routing protocol based on RL on the NoC routing path. The method selects a deadlock-free direction based on topology information and Runtime Thermal Management (RTM). The information obtained from Q-learning-based decisions is applied to improve balanced inter-layer traffic information performance.

Methodology

Reinforcement Learning (RL)

- The RL Model is based on the Markov decision process (MDP) and consists of the following elements:
    1. Present state of the agent St.
    2. Set of actions for the agent At.
    3. A state transition function Pa(St, St+1), which is the probability that the process transitions from state St to state St+1.
    4. An immediate reward Ra(St, St+1) is provided by the decision maker as the process moves from state St to state St+1 due to action At.
- Here, the state St and the action set At can be finite in some cases and infinite in cases like real numbers. Markov decision process contains a policy function π that determines the mapping from state to action, i.e., π(S) = St à At.

Methodology

Q-Learning

- Q-learning algorithm is inherited from the Reinforcement Learning theory that uses Q values to determine the system's state and action.
- The algorithm implements a Q table which consists of four fields: Current Space, Next Space, Action Space, and Q values. The Markov decision process (MDP) for the algorithm consists of the following elements:
    1. State (St) – In a 2-d mesh topology, the routers, or the nodes act as the system's state.
    2. Action (At) – At any instance, when a packet is at a node y, then the node has 2 possible paths to forward the packet towards the destination node. This selection defines the action of the system.
    3. Transition (T) – The movement of the packet from the current router to the next router simulates the transition.
    4. Reward (R) – Based on the state St and the Action At the system offers a reward to a node.
    5. Learning rate (α) – The learning rate determines the amount of information that is fed back to the router. It ranges from 0 to 1. If α is 1, then the old Q-value is completely replaced by the latest Q-value. When the α is 0, then Q-value remains unchanged.
    6. Discount factor (γ) – The value ranges from 0 to 1. For $\gamma = 0$, the system offers an immediate reward to the current state. For $\gamma = 1$, the current state receives a future reward.

Methodology

Q-Table

- To implement the proposed Q routing algorithm, a novel Q-table is designed that provides Q-values at every instance of the network.
- For an NxM 2-d mesh topology, where N – number of rows and M – number of columns, there are NxM nodes, each having a Q table. Each Q-table has 4 fields – current node, destination node, output port, and Q-value as shown in Table 1.

Current Node (y)	Destination Node (d)	Output Port	Q-value
4	0	South, West	…
4	1	South	…
4	2	South, East	…
4	3	West	…
4	4	Local	…
4	5	East	…
4	6	West, North	…
4	7	North	…
4	8	East, North	…

TABLE I. EXAMPLE OF Q-TABLE FOR NODE 4

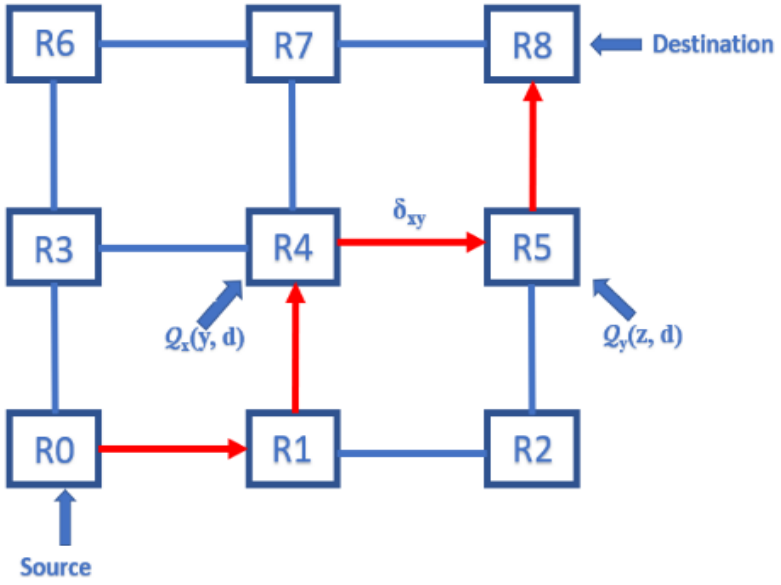


Fig 1. Q-Routing Algorithm

Methodology

Congestion-Aware Q-Routing algorithm

- Q-routing aims to find the most optimal routing path for a packet in a 2-d mesh topology. The algorithm is based on Q-learning.
- Each node is represented with an initial state using the values present in the Q-table. The system's action is to transfer the packet from a node to its neighboring nodes with minimum latency.
- When a source node x sends a packet towards destination node d through a node y, node y sends its minimum Q-value Qy(z, d) back to node x. When node x receives the best estimate Q-value from node y, it updates its Q-value, i.e., Qx(y, d).

$$Qy(z, d) = \min Qy(n, d)$$

- where n ∈ N(y) – set of y's neighbors.

$$Qx(y, d) = Qy(z, d) + qy + \delta$$

- Where qy is the queuing delay for the node y and δ is the link latency from node x to node y.

$$Qx(y, d)new = Qx(y, d)old + \gamma*(\alpha* Qy(z, d) + qy + \delta xy - Qx(y, d)old)$$

Congestion-Aware Q-Routing algorithm

- Algorithm 1 depicts Q-routing, which is designed and implemented in the gem5 simulation tool.
- The Q-update algorithm updates the Q-values of previous node after each iteration as shown in Algorithm 2.

Algorithm 1 Proposed Q-routing algorithm

1: A packet from source node x is transferred to destination node d through node y.
2: At node y, find the previous node using incoming port value available in route info.
3: Minimum Q-value is obtained from its neighbouring nodes using std::minimum function.
(a) Get destination node and compute the possible paths for the packet
(b) **if** current_node == destination_node **then**
(c) absorb the packet
(d) **else if** possible output port == 1 **then**
(e) Q-value = Q[current node][destination node][output port]
(f) **else**
(g) Find latency of both the output ports
(h) Q-value(1) = Q[current node][destination node][port 1]
(i) Q-value(2) = Q[current node][destination node][port 2]
(j) **if** Q-value(1) < Q-value(2) **then**
(k) Q-value = Q-value(1)
(l) **else**
(m) Q-value = Q-value(2)
(n) **end if**
(o) **end if**
4: The packet is transmitted from node y to the neighbouring node with least latency.
5: Node x receives the Q-value used by node y and updates it Q-values using Q-update algorithm.

Algorithm 2 Q-table updation algorithm

y = get_current_node()
2: dest = get_destination_node()
p = get_previous_node()
4: queuing_delay qy = destination_queuing_delay + source_queuing_delay
if Q-value($path_1$) < Q-value($path_2$) **then**
6: lat δ = get_link_latency(y, next_node ($path_1$))
Q_{new}[p][dest][$path_1$] = Q_{old}[p][dest][path_1] + 0.5(0.7*Q[y][dest] + qy+ δ - Q_{old}[p][dest][$path_1$])
8: **else if** Q-value($path_1$) > Q-value($path_2$) **then**
lat δ = get_link_latency(y, next_node ($path_2$))
10: Q_{new}[p][dest][$path_2$] = Q_{old}[p][dest][$path_2$] + 0.5(0.7*Q[y][dest] + qy+ δ + lat - Q_{old}[p][dest][$path_2$])
end if

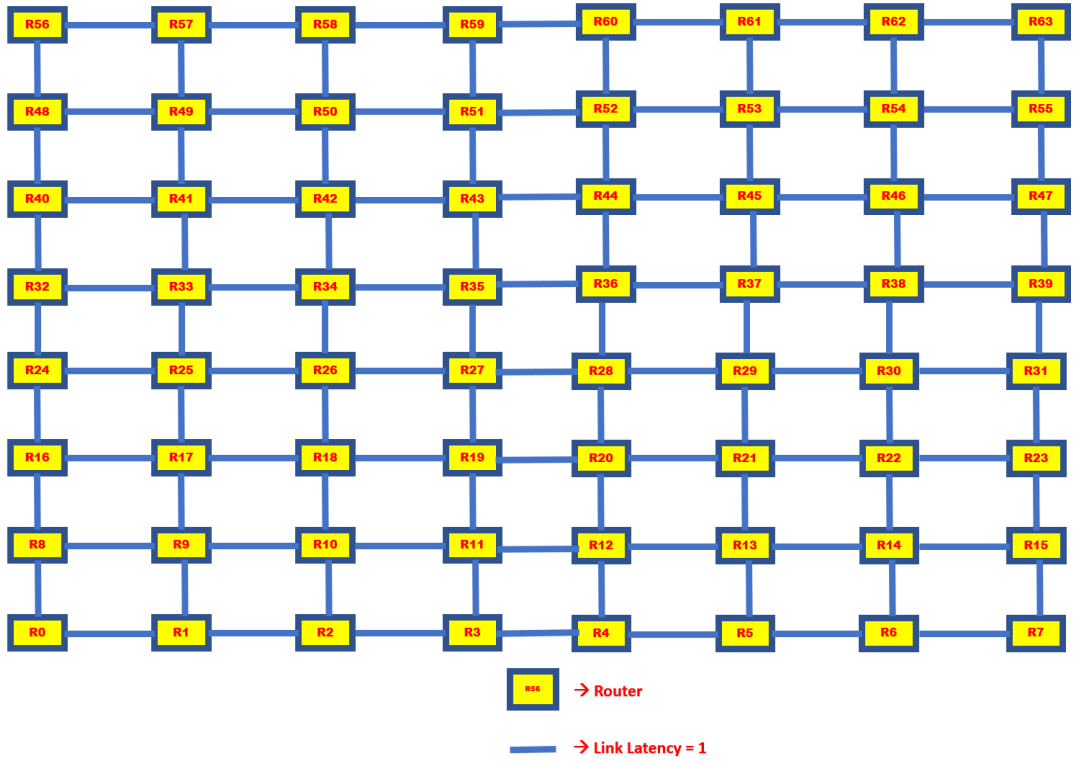
Experimental Results and Analysis

- The proposed work is carried out on gem5 simulation software in a System called Emulation mode (SE).
- In this work, ALPHA ISA, along with the Ruby memory model, is used on an 8x8 2-d Mesh Topology inside the Garnet network to evaluate the performance of the Q-routing algorithm compared to the default deterministic routing algorithms available in Gem5.
- For an 8x8 2-d Mesh XY NoC topology, the performance of the various routing algorithms is computed using three metrics – average packet latency, average power, and total energy.
- The work compares 3 routing algorithms based on these metrics – XY, Odd-Even, and Congestion-Aware Q-routing algorithms.
- Further, link latency between multiple nodes is randomly varied within the network since, by default, all the links have a latency of 1.
- Each simulation is carried out for 10,000 system ticks, and the generated metrics are used for comparison.

Experimental Results and Analysis

Uniform Link Latency

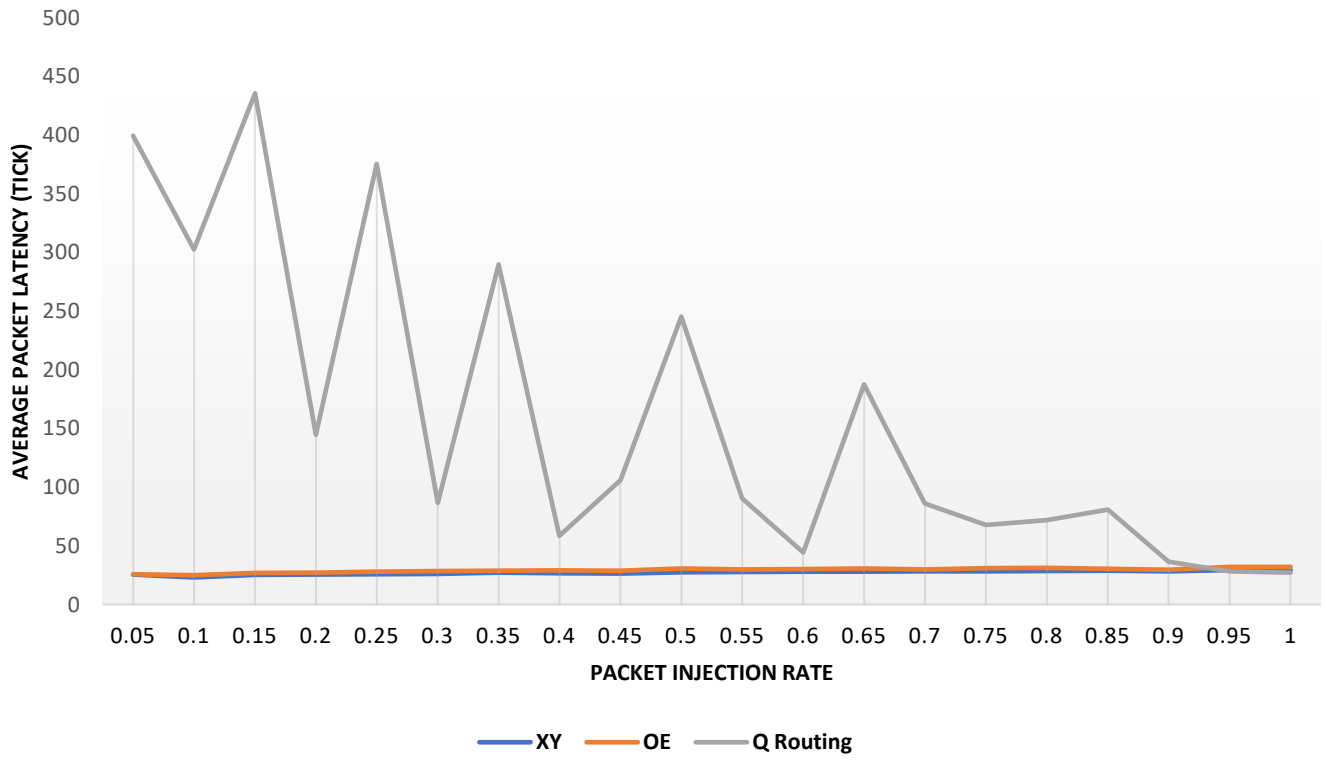
- The simulation is carried out on a Mesh XY network with uniform link latency.
- Be default, Gem5 offers a link latency of 1 between each nodes in the Mesh XY network



Experimental Results and Analysis

Uniform Link Latency

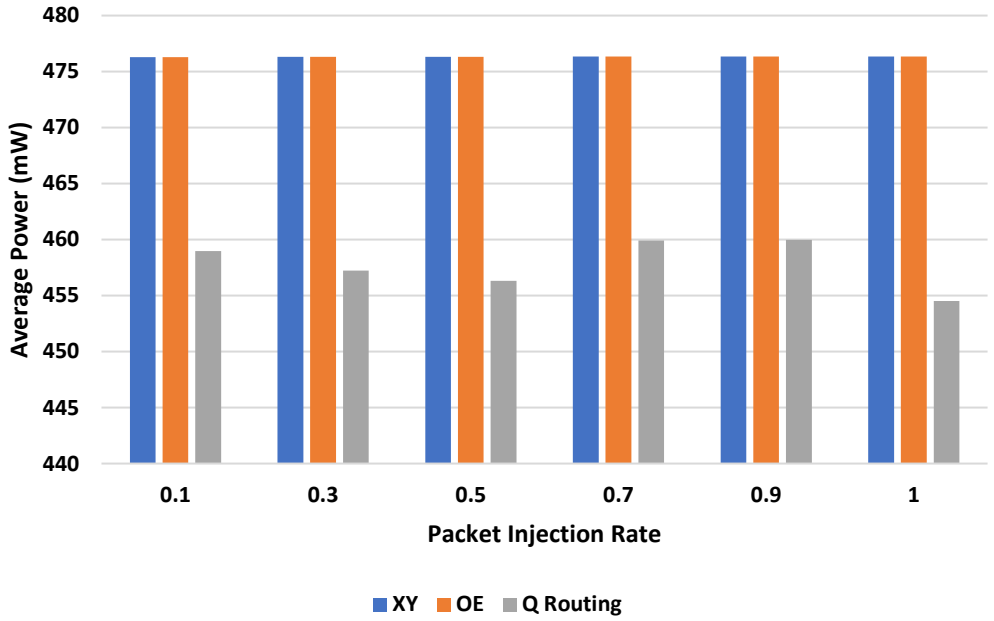
- The simulation is carried out on a Mesh XY network with uniform link latency.
- Performance gain of 7.38 % and 15.19% for XY and Odd-Even algorithm, respectively for injection rate of 0.95

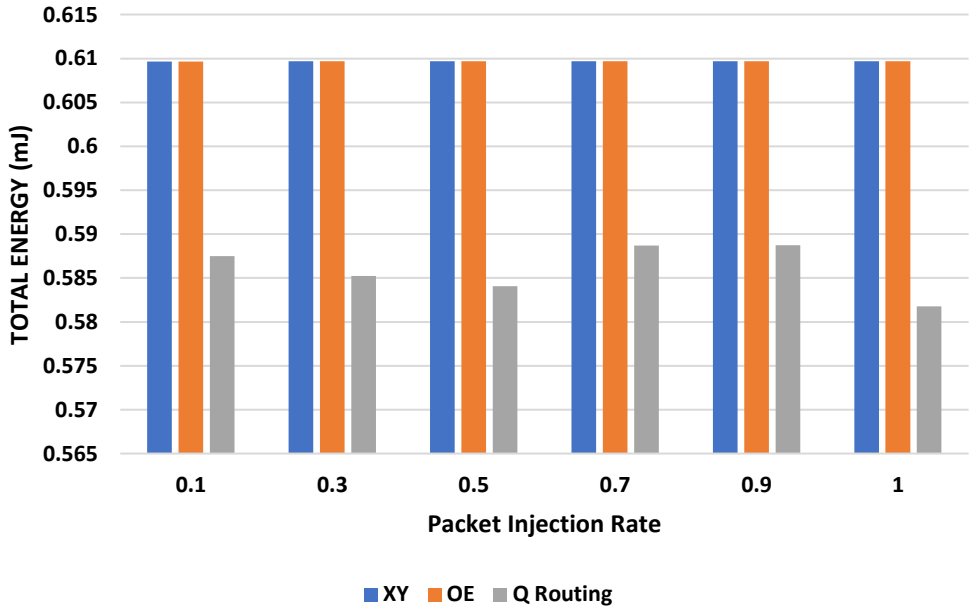


Experimental Results and Analysis

Uniform Link Latency

- Q-routing performs better in resource consumption by a factor of 3.4 % for an IR of 0.9.

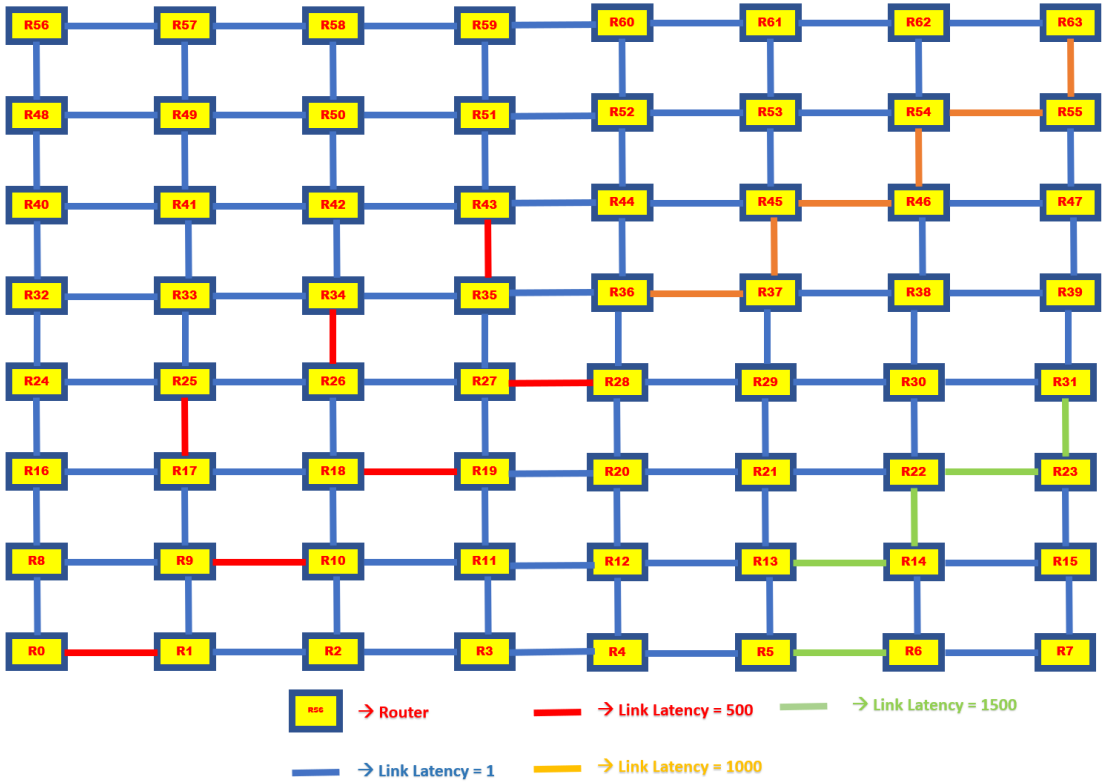




Experimental Results and Analysis

Random Link Latency

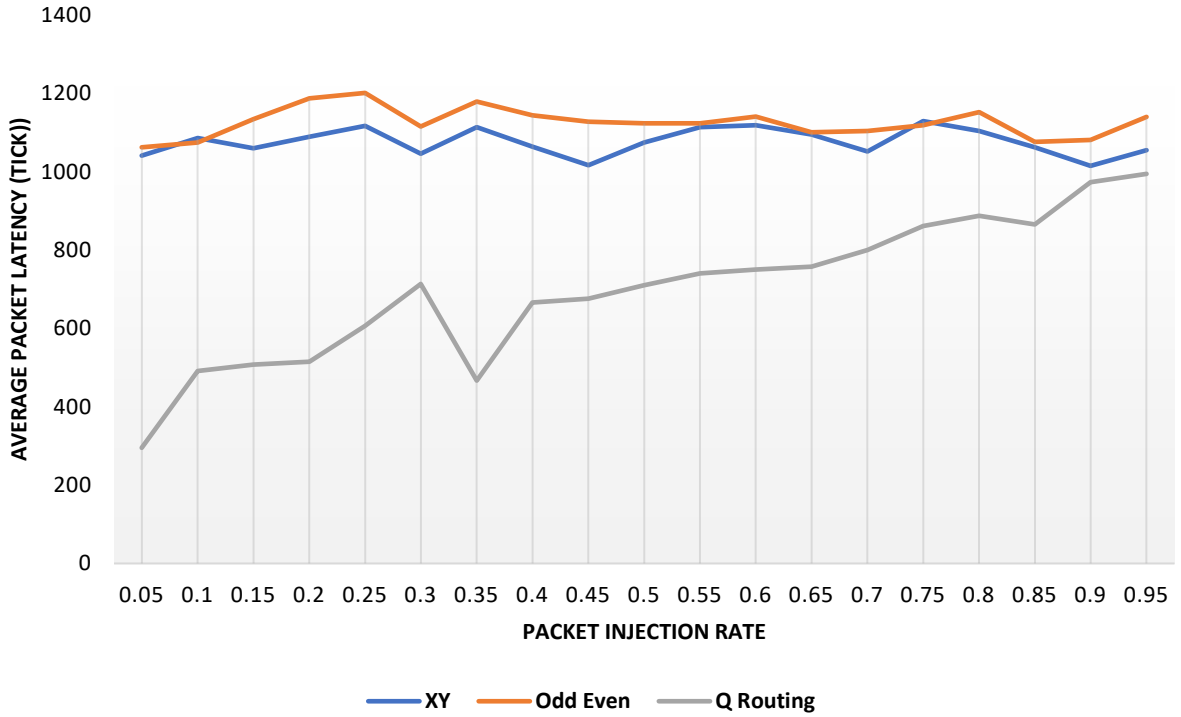
- The garnet2.0 network facilitates the variation of link latencies between the nodes in the topology.
- To simulate the network real-world scenarios of congestion on the links, the link delays during the creation of the 2-d Mesh XY topology are varied.



Experimental Results and Analysis

Random Link Latency

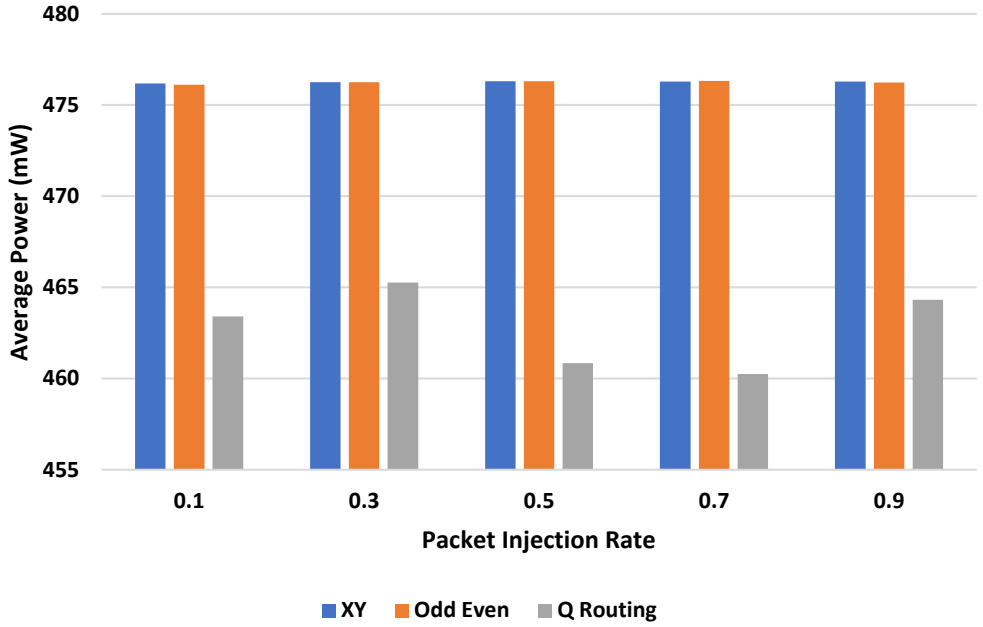
- When the network inherently has higher congestion, the Q-routing algorithm performs better than the deterministic routing algorithms.
- With increasing Injection Rate (IR) of the packets, the average packet latency of the Q-routing algorithm increases and saturates at an IR of 0.9. Q-routing shows a performance gain of 5.73 % and 12.73 %.

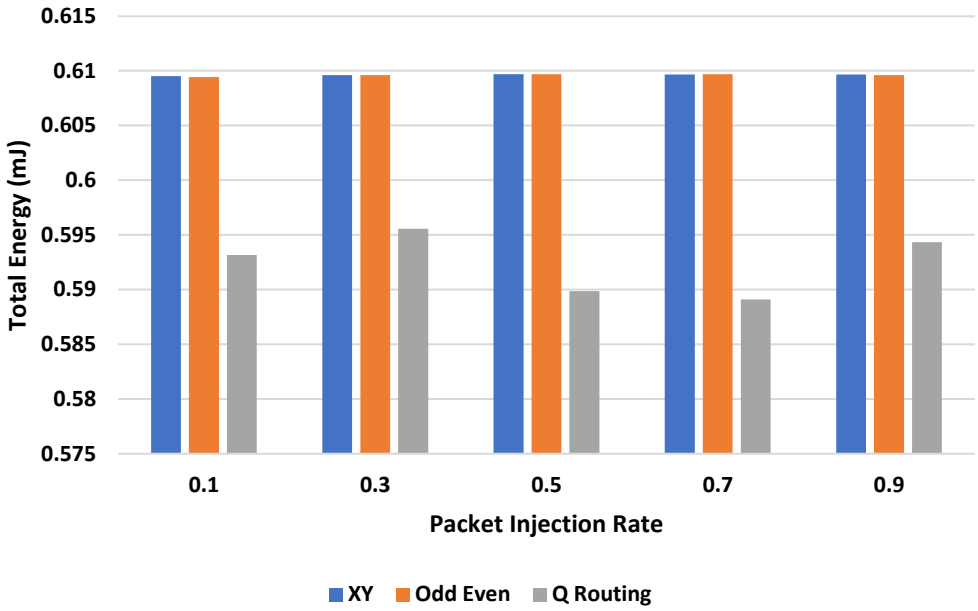


Experimental Results and Analysis

Random Link Latency

- The resource utilization on average shows an improvement of 4.7 % for a higher injection rate of 0.9.

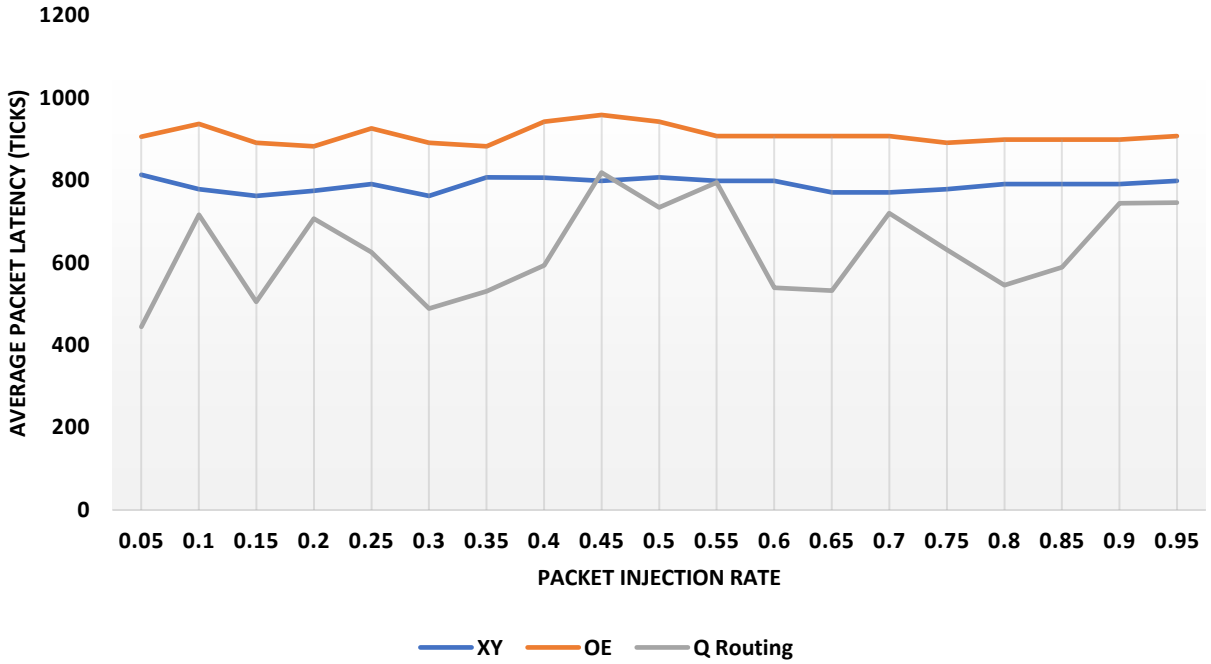




Experimental Results and Analysis

Transpose Synthetic Traffic Pattern

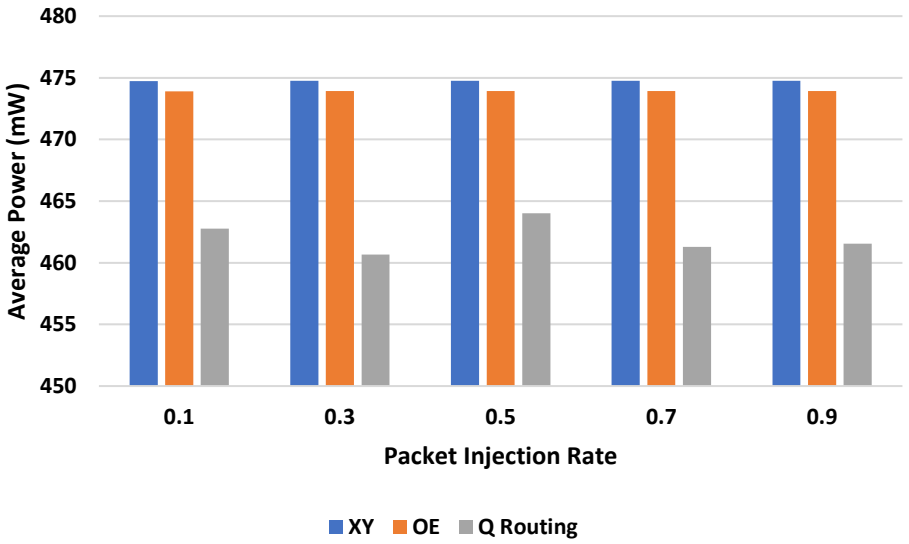
- The performance comparison of the Q-routing algorithm is extended to other types of synthetic traffic patterns in the network such as Transpose.
- In this traffic pattern, each node communicates solely with another node that is diametrically opposite to it.
- The results show that the average packet latency of Q-routing is better than XY and Odd-Even by an average factor of 19.9% and 30.54 % respectively-

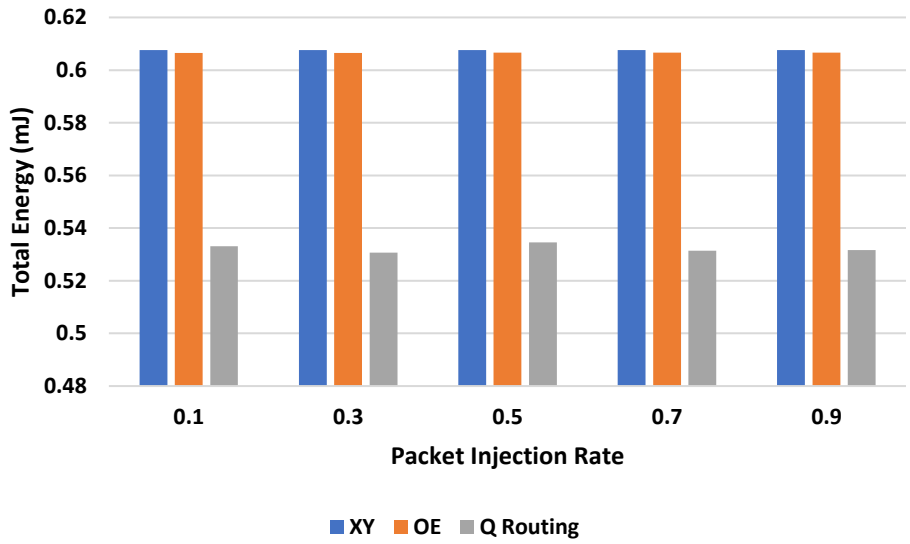


Experimental Results and Analysis

Transpose Synthetic Traffic Pattern

- The average power and total energy consumption of Q routing show an improvement of 3.15 % and 13.11 % compared to deterministic algorithms

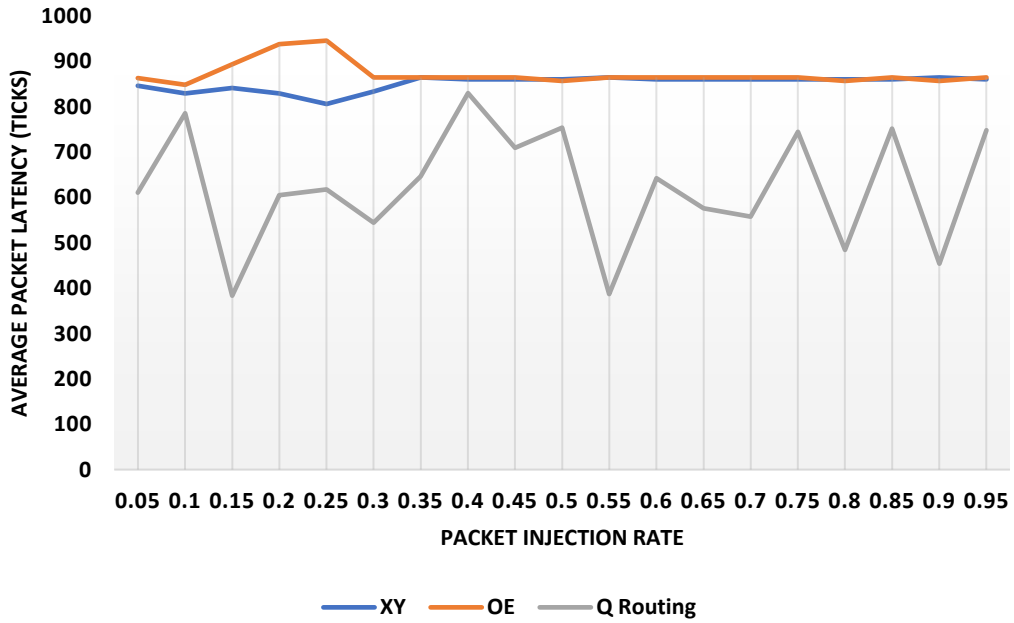




Experimental Results and Analysis

Bit Reverse Traffic Pattern

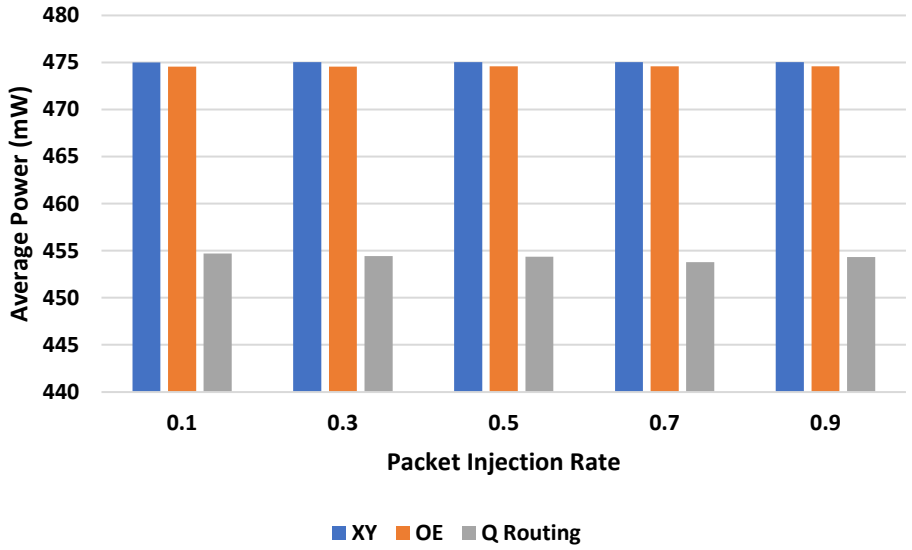
- In this traffic pattern, the binary representation of each node is used, and the packet is transferred to the destination node, whose binary representation is reversed compared to the source node.
- This traffic pattern was run on the 8x8 mesh-XY topology for 10,000 simulation ticks for varying packet injection rates.
- The average packet latency of Q-routing is better than XY and OddEven by an average factor of 26.88 % and 28.58 %, respectively.

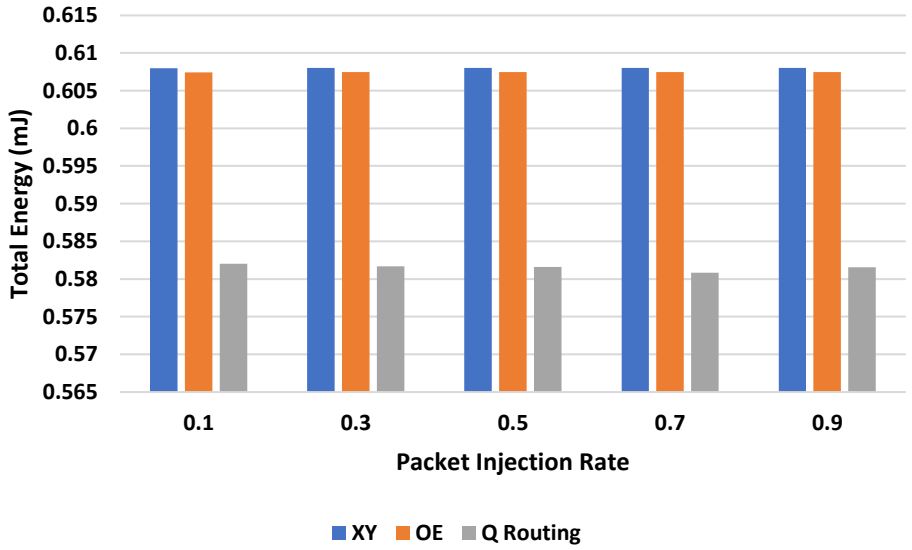


Experimental Results and Analysis

Bit Reverse Traffic Pattern

- The average power and total energy consumption of Q-routing algorithm show an improvement of 4.21 % and 4.09 % compared to deterministic algorithms





Conclusion

- This paper proposes a congestion-aware Q-routing algorithm, which reduces the average packet latency and power consumption in an NoC. The algorithm uses Q-values available in the Q-table at each node to realize the traffic conditions of the network.
- The simulation done on the gem5 simulator with uniform link latency in the network exhibits that Q-routing performs better in a high-load environment than traditional XY and Odd-Even Routing methods, with a performance gain of 5.73% and 12.73%, respectively.
- The results for varied link latencies that were randomly assigned to create a practical congestion-probable scenario showed that the proposed method outperformed both the XY and Odd-Even routing algorithm with a respective performance gain of 7.38% and 15.19%.
- Additionally, these results conform to different synthetic traffic patterns.

References

[1] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood, “The gem5 simulator”, SIGARCH Comput. Archit. News 39, 2, 2011, pp 1–7.

[2] Wang, K. and Louri, A., “Cure: A high-performance, low-power, and reliable network-on-chip design using reinforcement learning”, IEEE Transactions on Parallel and Distributed Systems, 31(9), 2020, pp.2125-2138.

[3] Chen, Y. and Louri, A., “An approximate communication framework for network-on-chips”, IEEE Transactions on Parallel and Distributed Systems, 31(6), 2020, pp.1434-1446.

[4] Zheng, H. and Louri, A., “Agile: A learning-enabled power and performance-efficient network-on-chip design”, IEEE Transactions on Emerging Topics in Computing, 2020.

[5] Lee, S.C. and Han, T.H., “Q-function-based traffic-and thermal-aware adaptive routing for 3D network-on-chip”. Electronics, 9(3), 2020, p.392.

[6] W. Choi et al., “On-Chip Communication Network for Efficient Training of Deep Convolutional Networks on Heterogeneous Manycore Systems," in IEEE Transactions on Computers, vol. 67, no. 5, 2018 pp. 672-686.

[7] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila and P. Liljeberg, "Optimized Q-learning model for distributing traffic in on-Chip Networks," 2012 IEEE 3rd International Conference on Networked Embedded Systems for Every Application (NESEA), 2012, pp. 1-8.

[8] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg and J. Plosila, "Q-learning based congestion-aware routing algorithm for on-chip network," 2011 IEEE 2nd International Conference on Networked Embedded Systems for Enterprise Applications, 2011, pp. 1-7

References

[9] Justin A. Boyan and Michael L. Littman., “Packet routing in dynamically changing networks: a reinforcement learning approach”, In Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS'93). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993 671–678.

[10] Yin, J., Che, S., Oskin, M., & Loh, G.H., “Toward More Efficient NoC Arbitration: A Deep Reinforcement Learning Approach.”, 2018.

[11] Smriti Srivastava and Dr. Minal Moharir, “An Overview of Communication Methods on WiNoC”, International Journal of Advanced Science and Technology, 29(3),2020, 12876 - 12883.

[12] D. Radha, Mushtaq Shaikh, Vamsi Silla, Minal Moharir, “Application Aware Workload Allocation to optimize the performance in Network on Chip based Manycore Processor”, Webology Journal, Volume 18, No. 6, 2021, Pages: 1639-1656.

[13] Radha Doraisamy, Minal Moharir, Rajakumar Arul, “Congestion aware and game based odd even adaptive routing in network on chip many-core architecture”, Indonesian Journal of Electrical Engineering and Computer Science Vol. 28, No. 2, November 2022, pp. 962-972.

[14] Smriti Srivastava, Adithi Viswanath, Krithika Venkatesh, and Minal Moharir, “TDMA-Based Adaptive Multicasting in Wireless NoC”, 6th International Conference on Inventive Systems and Control [ICISC 2022], organized by the JCT College of Engineering and Technology on 6–7 January 2022, Springer Lecture Notes in Networks and Systems, Volume 436.

[15] S. S. Shetty, M. Moharir, S. K and S. F. Ahmad, "Low Latency & High Throughput Wireless-NoC Architecture for Manycore Processors," 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS), 2018, pp. 1-5.

THANK YOU!!