**TTQR: A Traffic- and Thermal-Aware Q-Routing for 3D Network-on-Chip**

**Context**: 3D NoC systems face challenges such as high-power density and uneven thermal distribution due to their die-stacked architecture. These issues can lead to decreased reliability and performance.

**Objective**: The paper aims to develop a routing algorithm that considers both traffic congestion and thermal conditions to enhance the overall performance of 3D NoCs.

The main contribution of this paper is the proposal of a 3D NoC adaptive routing method based on the Q-learning mechanism, called the traffic- and thermal-aware Q-routing algorithm (TTQR). TTQR enhances the traditional Q-routing algorithm by optimizing its overhead area through the cancellation of dedicated links and the simplification of the Q-table. Additionally, it shifts from single-objective optimization to a multi-objective optimization approach.

Each router in the system features two tables: the Q1-table, which provides the buffer status of neighboring nodes, and the Q2-table, which offers global thermal information updated based on average temperature data from the headers of received packets. The TTQR method is capable of estimating and predicting the network's congestion and temperature conditions, utilizing this information to make informed routing decisions that prioritize less congested paths or regions with lower temperatures.

**Q-Learning**

Q-learning is a method within the realm of reinforcement learning. In this methodology, a learning agent explores an online environment model to devise an efficient control strategy for specific tasks. A Q-value represents the expected benefit gained from taking action ‘a’ at a particular moment in state ‘s’. The primary objective of Q-learning is to construct a Q-table, where the rows correspond to states and the columns correspond to actions. The Q-values in this table are continuously updated based on the rewards received from each action, serving as a foundation for determining subsequent actions.

**Q-Routing:**

The main idea of Q-routing is to store a Q-table at each router to evaluate the quality of the alternative paths. The Q-table stores an estimate of the time taken from each output port to the destination node.

1 **Packet Transmission and Q-table Update**: Whenever a router sends a packet to one of its neighboring routers, it utilizes a Q-table to determine the best possible action (i.e., which neighboring router to send the packet to). After the transmission, the Q-table is updated using a basic update mechanism. This mechanism typically involves adjusting the Q-value associated with the state-action pair based on the reward received after the action.

2 Learning **Packet Generation**: After successfully transmitting a packet to the downstream node (the next router), the downstream router collects information about the local and global traffic conditions. This information may include metrics such as congestion levels or temperature data, which are vital for optimizing future routing decisions.

3 Return **of the Learning Packet**: The downstream router then generates a learning packet containing the gathered traffic information. This learning packet is sent back to the upstream router (the one that initiated the transmission) via a dedicated link.

4 Incorporating **Global Information**: As the upstream router receives the learning packet, it updates its Q-table by incorporating the new global information into its Q-values. This allows the router to refine its understanding of the network conditions and improve its routing decisions over time.

5 Gradual **Information Accumulation**: This process of sending packets and returning learning packets allows each router to gradually accumulate more comprehensive global information about the network. As the routers learn from their interactions and share traffic data, they can adapt their routing strategies to better navigate congestion and optimize overall network performance.

**Traffic- and Thermal-Aware Q-Routing Algorithm (TTQR)**

* Our proposed method is able to balance the network traffic burden by selecting relatively idle output ports or routing directions pointing to low-temperature regions.
* The proposed technique employs both traffic and temperature as information for Qrouting and is therefore called traffic- and thermal-aware Q-routing

**Step 1: Collecting Deadlock-Free Channel Ports**

* **Routing Function**: The first step involves gathering a set of available channel ports that can be used to route packets without causing deadlocks. Deadlocks occur when packets are waiting on each other to move, preventing any progress.
* **Determining the Set Size**: The number of ports included in this set is based on the relative positions of the current node (the router sending the packet) and the destination node (the router receiving the packet). This ensures that only valid and accessible routing options are considered.

**Step 2: Selection Function Based on Q-Learning**

* **Q-Learning Mechanism**: The second step introduces a selection function that utilizes Q-learning to choose the best path from the available deadlock-free channels. This involves using two Q-tables:
  + **Q1-table**: This table maintains local traffic status information for neighboring routers.
  + **Q2-table**: This table holds global thermal information about the network, helping to identify areas of potential congestion or overheating.
* **Update Rules**: The update mechanisms for these Q-tables are defined, allowing routers to adjust their Q-values based on the traffic and thermal conditions they encounter during packet transmission.

**Routing Function:**

In the Traffic- and Thermal-Aware Q-Routing (TTQR) algorithm, the **Routing Function** is responsible for identifying deadlock-free paths to route packets while avoiding congested or overheated areas. Here's a breakdown of the process:

**Step 1: Identifying Throttled Nodes and Deadlock-Free Directions**

* **Throttled Nodes**: Throttled nodes are those with high congestion or excessive temperature, making them inefficient for routing. The algorithm first identifies such nodes in the **minimum area** surrounding the current node.
* **Deadlock-Free Routing**: The function aims to select routing directions that prevent deadlock. If the available paths are limited due to throttled nodes, the flexibility of selecting alternative paths diminishes. However, TTQR provides alternative deadlock-free paths using the **odd-even turn model** for horizontal routing and a **downward routing scheme** for inter-layer routing (between layers of a 3D network).

**Step 2: Intra-Layer and Inter-Layer Routing**

* **Horizontal (Intra-Layer) Routing**: The odd-even turn model is used for lateral (within the same layer) routing to prevent deadlocks in horizontal paths.
* **Vertical (Inter-Layer) Routing**: Vertical routing uses a **downward scheme** to avoid routing conflicts between layers. This reduces the chances of deadlock by ensuring that there are no circular dependencies between horizontal and vertical directions.

**Step 3: Flowchart Logic (Figure 1)**

* The algorithm starts by checking if the **destination node (Nd)** is throttled. If it is not, the algorithm compares the x and y coordinates of the **current node (Ns)** with Nd.
* If the coordinates are the same, the algorithm considers vertical routing. Otherwise, it divides routing into horizontal directions and downward routing.
* **Path Selection**: If there are no throttled nodes in the minimum region, the routing function ensures **path diversity** by selecting multiple routable directions. If throttled nodes are detected, the routing switches to downward paths to avoid those areas.

**Step 4: Preventing Circular Dependencies**

* To avoid deadlocks, TTQR prohibits **upward routing followed by lateral routing**, which could create circular dependencies between intra-layer and inter-layer routing.
* **Upward routing** is only allowed when the x and y coordinates of the current node and destination node are the same, and the z-coordinate of the destination is smaller than that of the current node.

**Key Points:**

* **Adaptive Routing**: TTQR adapts the routing strategy depending on traffic and thermal conditions to optimize performance, especially when the current node lies above a throttled layer.
* **Reduced Complexity**: The algorithm only searches the **minimum region** for possible paths, reducing computational overhead (O(4N) complexity).

In summary, TTQR's routing function dynamically selects deadlock-free paths by considering both congestion and thermal states, balancing traffic across the 3D NoC while avoiding hotspots and ensuring efficiency.

**Selection Function:**

In the **Selection Function** of the TTQR algorithm, the goal is to select the best routing direction based on the traffic and temperature conditions in the network. This decision is made using the information stored in **Q-tables**.

**Key Points:**

1. **Purpose**: The selection function aims to choose a routing direction that leads to a **non-congested port** or a region with **lower temperature**, optimizing both network traffic and thermal conditions.
2. **Q-Tables**:
   * **Previous Work**: In earlier algorithms, such as those described in [12,33], each router stores path estimates for all nodes in the network, leading to large Q-tables where each router contains detailed path estimates for every node.
   * **Optimization in TTQR**: To reduce the complexity and memory requirements, TTQR optimizes the Q-tables. Instead of using a table with the total number of nodes (which grows significantly in a 3D NoC), the **row index** of the Q-table is simplified to represent the **four directions** within the layer (e.g., North, South, East, West). This reduces the row size of the table from the number of nodes in the NoC to just 4 directions.
3. **Mesh Topology**:
   * In a 3D NoC with an **n × m × l mesh topology** (where **n** and **m** are the number of rows and columns in a layer, and **l** is the number of layers), the Q-table would initially have a size of **n × m × l**.
   * TTQR reduces this complexity by only considering the four directional paths available within a layer, simplifying the Q-table’s row size to just 4 entries.
4. **Two Q-Tables per Router**:
   * Each router stores **two Q-tables**, one for **local traffic information** and another for **global thermal information**.
   * Both tables contain 4 rows (for the 4 directions) and 4 Q-values that estimate the optimal direction for routing based on current network conditions.

The **Selection Function** in TTQR selects the best routing path based on a simplified Q-table structure. By reducing the table size to just 4 directions (instead of accounting for every node in the network), TTQR minimizes memory usage and hardware overhead while still making efficient routing decisions that balance traffic and thermal conditions.

Top of Form

**Q1-Table for Optimizing Latency : Bottom of Form**

The **Q1-Table** in the TTQR algorithm is used to **optimize latency** by evaluating the traffic status in different directions surrounding the current router.

**Key Concepts:**

1. **Q1-Value**:
   * Represents the estimated traffic status in a specific direction. A higher Q1-value implies that the direction is less congested and thus more favorable for routing.

**Update Mechanism**:

* + The Q1-values are updated over time as new traffic information is received. The update mechanism combines **old Q1-values** with newly estimated values from neighboring nodes to provide the most up-to-date estimate for routing decisions.

**Q2-Table for Optimizing Temperature :**

The **Q2-table** in the TTQR algorithm is used to optimize temperature management by estimating the average temperature along different routing paths. This helps prevent network-on-chip (NoC) overheating and ensures that data packets are routed through cooler regions of the network, reducing the risk of thermal hotspots.

**Key Concepts:**

1. **Q2-Table**:
   * The **Q2-value** represents the **average temperature** of the routers along a specific direction (North, East, South, West) from the current router to the destination.
   * Initially, all Q2-values are set to a default value (e.g., **outside temperature**). As packets travel through the NoC, these values are updated to reflect the actual thermal conditions.
2. **Temperature Update Mechanism**:
   * Unlike traditional methods that require **learning packets** to update Q2-values, TTQR eliminates the need for additional packets and dedicated links. Instead, the **temperature information** is carried within the headers of regular data packets.
3.  When a router receives a packet, it extracts the **average temperature** (Avg\_Temp) from the packet header, which represents the temperature of routers the packet has passed through.
4.  The Q2-value for the direction from which the packet arrived is updated using **Equation**
5. After updating its own Q2-table, the router must **modify the Avg\_Temp** in the packet before forwarding it.

The **Q2-table** helps optimize the routing decisions based on **thermal conditions** by dynamically updating temperature estimates for each direction. By embedding thermal data within packet headers, TTQR avoids the overhead of dedicated learning packets, allowing routers to make more informed routing decisions without additional communication cost.

**Summary of TTQR**

The Traffic- and Thermal-aware Q-routing algorithm (TTQR) is designed to optimize routing in a 3D NoC by considering both traffic congestion and thermal conditions. Here’s a summary of its operation:

1. **Routing Decision**: When a packet reaches router r, the router checks if there is a throttled node in the minimal region. If so, the packet is routed to the next layer; otherwise, it selects a set of horizontal output channels using a parity turn routing mechanism.
2. **Selection Mechanism**: The router compares two potential output channels based on their Q1-values (representing traffic status) and Q2-values (representing temperature). If only one output channel is available, the router selects it directly. If there are two, the router chooses the direction with the highest ratio of Q1 to Q2.

**Simulation Results and Discussion:**

We evaluated the average latency, throughput, temperature, and traffic load distribution of our proposed method and compared them with the TAAR routing algorithm.

**Analysis of Network Performance:**

Analyse the performance of NoC by measuring the average delay and throughput of packets under different injection rates.

* Compared with TAAR, the throughput of TTQR is improved by 25.3–50.0%.
* The network latency of TTQR shows improvements of 21%, 40.9%, and 128.6% over TAAR (Thermal-aware Adaptive Routing), depending on the traffic pattern.
* TTQR outperforms the TAAR routing algorithm by an average of 63.6% and 41.4% in average latency and throughput, respectively.

**Analysis of Statistical Traffic Load Distribution (STLD):**

**TAAR (Thermal-aware Adaptive Routing)**:

* The traffic load is concentrated in the bottom layers due to its inter-layer routing approach, which does not account for traffic load between layers.
* When TAAR encounters throttled areas and the destination is in a different layer, it prioritizes downward routing, leading to traffic congestion in lower layers.

**TTQR (Traffic- and Thermal-aware Q-routing)**:

* TTQR has a more balanced distribution of traffic load across layers, preventing bottlenecks at the bottom layers.
* The algorithm directs traffic toward peripheral paths when routers overheat, avoiding overloading specific areas, and ensures smoother downward routing as a secondary option.

**Analysis of Temperature Distribution:**

The Analysis of Temperature Distribution compares the temperature patterns in the network under various synthetic traffic patterns for TTQR, Q1, Q2, and TAAR.

* TTQR (Traffic- and Thermal-aware Q-routing):
  + TTQR provides a more uniform temperature distribution across network layers compared to TAAR.
  + This balance is achieved because TTQR considers both local throttling points and global thermal data when making routing decisions. It directs traffic to cooler areas on the perimeter links before resorting to downward routing, helping to prevent hot spots.
  + While the average temperature is slightly higher than TAAR (with a negligible difference of around 0.2%), TTQR effectively reduces the formation of throttling points by monitoring real-time temperature conditions through its Q-Thermal mechanism.
* TAAR (Thermal-aware Adaptive Routing):
  + TAAR's approach of routing packets downward upon encountering throttling nodes results in concentrated heat in lower layers. This leads to less uniform temperature distribution.
  + Unlike TTQR, TAAR lacks a mechanism to distribute traffic based on real-time thermal information, contributing to uneven temperature spread and potentially greater hot spots.

In conclusion, TTQR's routing strategy, which takes into account both traffic load and thermal conditions, offers a more balanced temperature distribution across the network, improving overall system stability without significantly increasing the average temperature.

To address thermal issues in 3D NoCs, numerous temperature management techniques have been developed. However, balancing traffic remains a challenge, as imbalanced traffic can lead to performance degradation. This paper proposes a **Traffic- and Thermal-aware Q-routing algorithm (TTQR)** tailored for 3D on-chip networks.

The key features of **TTQR** include:

* **Dual Q-tables**: Each router maintains two Q-tables, one for traffic congestion and one for temperature information. Routing decisions are made based on these values, allowing the selection of less congested and cooler paths.
* **Performance Improvement**: Experiments using synthetic traffic patterns demonstrate that TTQR significantly outperforms TAAR in terms of both latency and throughput. Specifically, TTQR improves latency by an average of 63.6% and throughput by 41.4% compared to TAAR.
* **Efficiency**: The Q-tables used in TTQR are compact, and no extra links are needed for learning packets, keeping the hardware overhead low.

These attributes make TTQR a practical and efficient solution for 3D NoCs, enhancing performance while maintaining low thermal impact and minimal resource usage.