Panamerican Mathematical Journal
ISSN: 1064-9735
Vol 34 No. 4 (2024)

Performance Analysis of Multi-Core Systems in Multistage Interconnection Networks: Investigating Challenges in Inter-Processor Communication

Abhay B. Rathod[1], Dr. Sanjay M. Gulhane[2]

[1]Jawaharlal Darda Institute of Engineering and Technology, Yavatmal, India. abhay_rathod@jdiet.ac.in

[2]Pravara Rural Engineering College, Loni, Ahmadnagar, India. sanjay.gulhane@pravara.in

Article History:

Received: *04-07-2024*

Revised: *18-08-2024*

Accepted: *03-09-2024*

Abstract:

For high-performance computers and data-heavy apps, how well multi-core systems work in multistage interconnection networks (MINs) is very important. As the need for parallel processing grows, inter-processor communication (IPC) has become a major issue that affects both the speed of work and the ability of the system to grow. This essay looks into the problems with IPC in multi-core systems set up with MINs, mainly delay, bandwidth limits, and network congestion. imulation-based performance review is used to test multi-core systems with different amounts of traffic, message sizes, and route techniques. A mix of mathematical models and simulation tools are used to test different MIN designs on measures like delay, speed, and fault tolerance. It is also looked at how network structures and methods for controlling crowding affect IPC efficiency. According to the results, MINs can be used to connect multiple devices, but when there is a lot of traffic, the delay goes up a lot because of bottlenecks in some parts of the network. Also, in higher-dimensional networks, processors fighting over bandwidth causes speed to drop. Some of these problems can be solved with optimized transport methods and adaptable congestion control systems, which also make IPC work better overall. This study shows how important it is to keep improving how processors talk to each other in multi-core systems, especially when it comes to making network switching and managing congestion more efficient.

Keywords: Multi-Core Systems, Inter-Processor Communication (IPC), Multistage Interconnection Networks (MINs), Network Congestion, Latency Analysis, Routing Algorithms

1. Introduction

Multi-core systems have changed the way high-performance computing is done by letting complex, data-heavy jobs run in parallel on multiple processing units. The structure of communication networks is very important for determining how well and how easily these systems can be expanded as the need for more computing power and efficiency grows. Multistage Interconnection Networks (MINs) are becoming a popular way to connect multiple processors in multi-core systems because they can handle a lot of data and offer flexible ways to do parallel processing. Multiple steps of switches make these networks possible for a huge number of working elements to talk to each other. This makes them an important part of high-speed computer settings. However, as the number of cores keeps going up, Inter-Processor Communication (IPC) in MINs has become very hard to do efficiently [1]. This can stop multi-core systems from getting better performance. Inter-processor

communication is an important part of multi-core systems because it lets data flow between cores so they can work together on tasks. When it comes to MINs, IPC means sending data bits through several steps of linked switches until they reach the right processor. MINs provide an organized and flexible way to connect multiple cores, but they also come with a number of issues that may slow down the system. Some of these problems are delay, bandwidth competition, network overcrowding, fault tolerance, and scaling issues. All of these can make communication less effective, causing slowdowns and lower total speed [2]. As more cores are added to a system, the amount of data flow grows very quickly. This means that the connection problems in these networks need to be looked into and fixed right away.

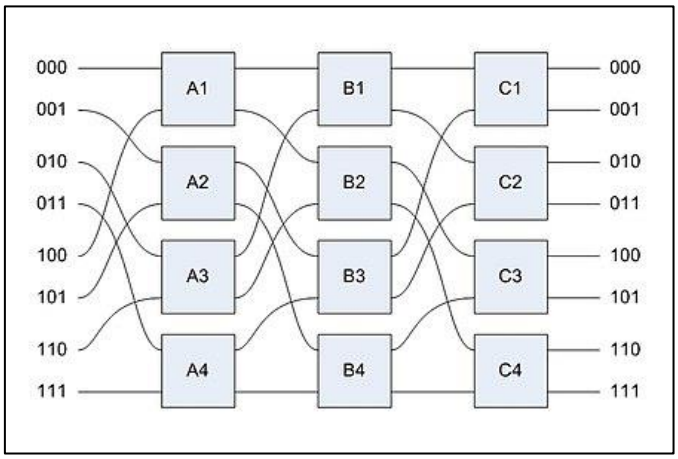

Figure 1: Overview of multistage interconnection networks architecture

Because MINs are designed in a way that requires data bits to go through many steps before they reach their final location, IPC in multi-core systems is very complicated. The route methods used, the network's structure, and how it handles multiple data exchanges at the same time all affect how well and reliably people can communicate. Network congestion may be a major issue since it causes higher delay and lower speed, which in turn harms the execution of synchronous apps, as show outline in figure 1. Typically particularly irritating when numerous computers have to be utilize the same assets at the same time, which causes delays and strife. Blame resistance is additionally an critical thing to think around in MINs since association issues at any point can halt the stream of information, so the framework needs solid error-handling highlights to remain solid. To bargain with these issues, scholastics have looked into diverse ways to move forward IPC speed in frameworks with different centres that utilize MINs. A few of these strategies are making way better course frameworks, more adaptable ways to handle activity jams, and unused organize formats that are implied to spread out the work equitably and cut down on holding up times for messages. For case, adaptable steering strategies can alter the course of information bits based on the current state of organize activity. This makes communication speedier and less swarmed by and large. So also, utilizing fault-tolerant methodologies like different courses and error-correction strategies makes beyond any doubt that communication remains solid indeed on the off chance that equipment comes up short or there are issues with the information exchange [3].

The goal of this study paper is to give a thorough look at the problems that come up when trying to get processors to talk to each other in multi-core systems with multistage link networks. Through

looking into what causes delay, bandwidth contention, and network congestion, this study aims to find possible ways to improve IPC performance. The study also looks at how well different routing methods, congestion control techniques, and network layouts work at dealing with these problems. This helps us understand how to make MINs work best on high-performance multi-core systems. More and more people want faster and more efficient computers. To get the most out of multi-core architectures in areas like scientific simulations, data analytics, and real-time processing, it's important to understand and deal with the complexities of IPC in MINs. The goal of this study is to help with the current work to create flexible, high-performance multi-core systems that can handle the needs of next-generation computer settings.

2. Related Work

A lot of study has been done on how to make inter-processor communication (IPC) more efficient in multi-core systems. This is especially important in high-performance computing settings where MINs are used. Many studies have been done to look into the problems of delay, bandwidth contention, congestion, and fault tolerance in these kinds of systems. The goal is to make transmission more efficient and the systems more scalable [4]. This part gives an outline of the related work, focusing on the most important inputs, methods, and results in the area. Previous work on IPC in multi-core systems has mostly been about making route methods in MINs work better so that there is less delay and more efficient data flow [5]. One of the first studies in this field compared fixed and adaptable routing methods by looking at how they affected transmission delay when traffic conditions changed [6]. When there is a lot of traffic, deterministic routing, which sends data bits along a set path, often gets clogged up and has longer delay [7]. Adaptive routing methods, on the other hand, have shown promise in changing data paths based on how the network is currently configured, which can help reduce congestion and delay [8]. New developments have made adaptive routing even better by adding machine learning models that can predict traffic trends and improve packet routing in real time, which makes communication much more efficient [9].

Another range of related work looks at how to make strides IPC speed in multi-core frameworks by planning and testing diverse organize formats. A few of the foremost examined MIN topologies are the Butterfly, Omega, and Clos systems. Each has its possess benefits when it comes to development, blame resilience, and communication effectiveness [10]. Comparing these plans has appeared that the Clos organize ordinarily performs superior in terms of blame tolerance and bandwidth utilization. This can be since it has different ways, which suggests it can utilize diverse courses in the event that one course gets clogged or equipment comes up short [11]. However, the trouble and taken a toll of setting up these sorts of topologies can be a problem. Usually why analysts are still looking into blended organize plans that utilize the most excellent parts of diverse topologies to induce the finest comes about [12]. Another vital portion of IPC that has gotten a part of consideration within the books is clog control. When MINs get as well active, transmission slacks get longer, yield goes down, and the framework as a entirety works more awful [13]. To illuminate this issue, scholastics have looked into a number of distinctive ways to control packing, such as buffer administration strategies, priority-based parcel requesting, and traffic-aware directing calculations [14]. A enormous step forward in this zone is the creation of congestion-aware directing conventions that sort parcels by how critical they are and alter the heading of activity on the fly to dodge swarmed regions [15].

This method has shown big gains in cutting down on delay and increasing performance, especially in systems with a lot of traffic [16]. Another important thing to think about when designing IPC methods in MINs is fault tolerance, since hardware problems or data transfer mistakes can have a big effect on how reliable the system is [17]. In the beginning, researchers mostly looked at adding redundancy at different points in the network to make it more fault-tolerant. For example, they used multiple parallel lines for data transfer [18]. Recently, people have been working on fault-tolerant routing algorithms that can find and avoid broken network parts. This way, communication can still happen even when there are problems [19]. These methods have been used successfully in big multi-core systems [20] and have been shown to ensure stable IPC performance.

Bandwidth improvement is another important area of study. People are working to make better use of available bandwidth so that data flow rates are higher [21]. Techniques like load balancing, which spreads traffic fairly across all available network lines, have been shown to be very good at keeping computers from hitting speed limits and making sure they can talk to each other without any problems [22]. As a way to improve IPC even more, bandwidth compression techniques have become popular. These techniques lower the amount of data sent over the network without affecting the security of the data.

Table 1: Literature review summary

Approach	Methodology	Techniques	Limitation	Findings	Advantage
Adaptive Routing in MINs	Simulation-based performance analysis	Dynamic routing algorithms	Increased complexity in implementation	Reduced latency under heavy traffic conditions	Improved communication efficiency
Fault-Tolerant Communication	Redundant path implementation	Parallel path redundancy	Higher hardware costs and increased complexity	Enhanced fault tolerance and reliable communication	Increased reliability in large-scale systems
Congestion Control Mechanisms	Analytical and simulation-based evaluation	Priority-based packet scheduling	Increased latency in low-priority packets	Improved throughput under high traffic loads	Better handling of network congestion
Topology Optimization	Comparative analysis of different topologies	Clos, Butterfly, Omega networks	Complexity in large-scale implementations	Clos network offers superior fault tolerance	Scalability and multi-path routing capability
Machine Learning for Traffic Prediction	Real-time traffic prediction models	Neural networks and reinforcement learning	Requirement of large datasets for training	Effective congestion prediction and routing	Dynamic adjustment to changing traffic patterns
Load Balancing in	Load distribution	Equal traffic distribution	Limited scalability in	Reduced bottlenecks	Efficient use of network

MINs	techniques evaluation	algorithms	highly dynamic networks	and improved bandwidth usage	resources
Bandwidth Compression	Data reduction techniques	Lossless compression algorithms	Risk of data integrity loss in high compression	Reduced transmission time with maintained integrity	Enhanced bandwidth efficiency
Hybrid Network Designs	Experimental evaluation	Combination of different topologies	Increased implementation cost	Achieved optimal performance across different traffic patterns	Adaptable to various network sizes
Deterministic Routing	Fixed-path routing analysis	Static routing algorithms	High latency under varying traffic loads	Simple implementation in small networks	Consistency and predictability in routing
Fault-Tolerant Routing Algorithms	Error detection and rerouting	Adaptive path rerouting	Overhead in error handling processes	Maintained communication integrity in case of faults	Resilient to hardware failures
Network Topology Comparison	Performance benchmarking	Simulation of different topologies	Limited applicability to dynamic network changes	Clos topology has higher bandwidth utilization	Provides insights for topology selection
Dynamic Load Adaptation	Load monitoring and adaptation mechanisms	Real-time load balancing	Increased overhead in monitoring and adaptation	Enhanced network performance under variable loads	Better adaptation to changing processing demands
Fault Recovery Mechanisms	Fault injection and recovery testing	Error correction and data redundancy	Requires additional resources for fault recovery	Improved system reliability in multi-core systems	High tolerance to data transmission errors

3. Multistage Interconnection Networks (MINs)

Multistage Interconnection Networks (MINs) are an important part of the design of multi-core systems because they give high-performance computers an organized way to connect many

processors and memory units. They have many stages with moving parts that make it easier for different computers to talk to each other. This lets data go from a source to where it needs to go through many intermediate stages. This organized layout makes it easy to send data packets to the right place, which makes MINs perfect for dealing with the complicated communication needs of multi-core systems. Many layers of switches in the network make it possible for any processor to talk to any other processor or memory module. This makes the system flexible and scalable.

MINs are usually put into groups based on the moving parts and link patterns that are used in each stage. The Butterfly, Omega, and Clos networks are some of the most studied designs. Each has its own specific setups that affect how well processors can talk to each other. For example, the Butterfly network has a linear structure with various lines. This makes sure that there are many ways for data to be sent. This design helps cut down on contact delays, but it can have problems when there is a lot of traffic. The Omega network, on the other hand, has a simpler setup that makes sure each communication process has a specific path. This makes routing easier, but it also makes the network more likely to get crowded when many computers try to talk at the same time. Many people think that the Clos network is one of the most efficient. It has a multi-path design that makes it better at handling errors and using bandwidth, which makes it perfect for big, high-performance computers.

As a system's number of processors grows, the multistage structure of these networks can be expanded by adding more switching stages. This makes it possible for more working elements to be added without any problems. This makes MINs very flexible when it comes to the changing needs of modern computers, which need more and more parallel processing and fast data sharing. The Clos network and other MIN designs have the ability to take more than one way. This means that even if there is congestion or a hardware failure, there are still other routes that can be used. This makes the system more reliable and sturdy overall. MINs also have to deal with a number of problems that affect how well they work in systems with multiple cores. Network congestion is a big problem that happens when many data bits try to use the same way at the same time. This causes delays and lower speed.

Multistage Interconnection Networks (MINs) help various processors and memory units in multi-core computers talk to each other more efficiently. There are several steps of switching elements in MINs' communication process, and data bits are routed in a certain order. Here is a step-by-step guide to the data transfer method in MINs, along with the math models that are used to explain it:

Step 1: Network Initialization and Topology Configuration

Defining the network structure is the first thing that needs to be done to set up a multistage communication network. Most MINs have more than one stage (m) with a number of inputs and outputs (k) that are linked by switching elements (SEs).

- The total number of stages (S) can be represented as:

$$S = log2(N)$$

where (N) is the total number of processors or memory modules in the system.

- Each stage contains (N/2) switching elements, each with (2 x 2) crossbar switches. The overall network consists of (S x N/2) switching elements.

Step 2: Data Packet Generation and Address Mapping

Once the network is initialized, data packets are generated by the source processor. Each packet contains information about its destination address, which is represented in binary form.

- If a source processor (P_s) sends data to destination processor (P_d), the binary representation of (P_d) determines the path through the stages.

- The i-th stage switching element (SE) is controlled by the i-th bit of the destination address (d). Therefore, the switching control function (f(i, d)) can be expressed as:

$$f(i,d) = d[i]$$

where d[i] represents the i-th bit of the binary address of the destination.

Example: If (P_d = 5) (binary 101), the control bits at each stage will be 1, 0, and 1, respectively.

Step 3: Data Routing through Switching Elements

Based on the numerical version of the target address, the data packet is sent to the right place at each stage of the network. The control bit tells each switching element which direction the packet should take.

- The switching function at each stage can be defined as:

$$SE\_ij =$$

$$\{X -> Y, if\ f(i,d) = 0$$

$$Y -> X, if\ f(i,d) = 1\}$$

where SE_ij represents the switching element at stage (i) and position (j), and (X -> Y) indicates the path taken by the data packet.

- The overall path (P_route) taken by a packet can be represented as a sequence of control bits:

$$P\_route = [f(1,d), f(2,d), \ldots, f(S,d)]$$

Step 4: Congestion Handling and Performance Analysis

As data bits move through the network, more than one packet may try to go through the same switching element at the same time, which causes congestion.
- A queuing theory method can be used to describe quantitatively the chance of congestion (P_congestion) at a switching element:

$$P_{congestion} = 1 - e^{-\frac{\lambda}{\mu}}$$

where (λ) is the arrival rate of packets, and (μ) is the service rate of the switching element.

- The average latency (L) experienced by a packet in the network is given by:

$$L = \frac{1}{\mu - \lambda}$$

indicating that as traffic (arrival rate λ) increases, the latency grows, particularly as it approaches the service rate μ.

Step 5: Fault Tolerance and Redundant Path Calculation

To ensure reliable data transfer, fault tolerance is implemented by creating redundant paths for data packets in case of switching element failure.

- The redundancy factor (R) can be expressed as:

$$R = N\_redundant\_paths / N\_total\_paths$$

where (N_redundant_paths) represents the number of alternative routes available, and (N_total_paths) is the total potential routes in the network.

- The probability of successful communication (P_success) can be calculated as:

$$P_{success} = 1 - (1 - p)^R$$

where (p) is the probability that any given path is operational.

Step 6: Bandwidth Utilization Analysis

The efficiency of bandwidth utilization in an MIN can be determined by analyzing the throughput (T) of the network, given by:

$$T = N \times \frac{Number\ of\ successful\ transmissions}{Total\ transmission\ attempts}$$

- The bandwidth utilization (U) is then represented as:

$$U = \frac{T}{B_{max}}$$

where (B_max) is the maximum possible bandwidth.

By using mathematical models to look at each of these steps, we can test and improve the performance of multistage link networks so that processors in multi-core systems can talk to each other more efficiently. This step-by-step process helps find important factors like delay, congestion, fault tolerance, and bandwidth efficiency that are needed to create and build MINs that work well.

Multistage Interconnection Networks (MINs) are made to connect many computers and memory units quickly. To do this, they use various designs to make contact easier. The Butterfly, Omega, and Clos networks are some of the most studied and used designs in MINs. When it comes to scaling, problem tolerance, and communication speed, each of these designs has its own pros and cons.

1. The Butterfly Net

The Butterfly network is one of the most basic and widely used MIN designs because its shape is ordered and easy to understand. It is made up of several stages with moving parts set up in a way that looks like butterfly wings, shown in figure 2.

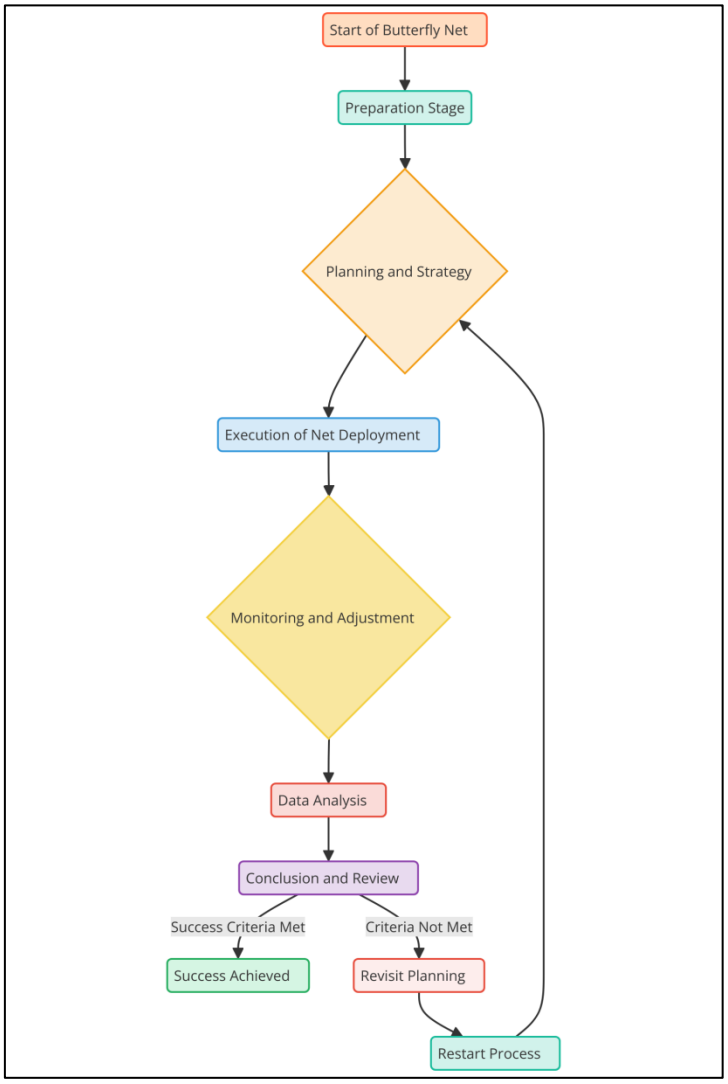

Figure 2: Architecture for The Butterfly Net

Type of network: The Butterfly network is made up of n steps, where n is the number of computers or memory units. There are two to two hundred switching elements in each stage. Each element has two inputs and two outputs, making it a classic two-by-two crossbar switch. The tracks that data takes from source to goal are marked, and there are many ways to get from one stage to the next.

Routes: In the Butterfly network, the code version of the target address tells the network how to send data. At each step, the route choice is based on a different part of the destination address. This makes sure that the data is sent in the right direction to reach its goal.

2. The Omega Network

Another famous MIN design is the Omega network, which has a more regular and reliable structure than other layouts. Because it is simple and easy to set up, it is used a lot in parallel processing systems.

Topology: The Omega network has ~ log 2 (N) stages, just like the Butterfly network, but it connects to other networks in a different way. Every stage has N2N moving elements, and the links between

stages are set up in a shuffle-exchange way. This makes the network structure more uniform and regular, with each switching element linked to several elements in the next step.

3. Close the network

The Clos network is one of the most powerful and adaptable MIN designs. It can handle a lot of faults and grow as needed. It's named for Charles Clos, who came up with this way of setting up networks that are very connected and have little blocking.

There are three major stages in the Clos network's structure: an input stage, a middle stage, and an exit stage. The fact that it doesn't block connections means that any input can be linked to any output without any problems. Most of the time, the network is written as **L**(m, $\boldsymbol{T}$, i)C(m,n,r), where mm is the number of input and output ports in the first and last stages, $\boldsymbol{T}$n is the number of crossbar switches in the middle stage, and ir is the number of crossbars in the input and output stages.

Routing: There are several ways for data to get from a source to a target on the Clos network. It can move data bits on the fly based on the amount of traffic, which makes it very good at dealing with congestion. This innovative feature of dynamic routing makes sure that the data can still get where it needs to go even if one way is stopped.

4. Challenges in Inter-Processor Communication (IPC)

4.1 Latency

In Multistage Interconnection Networks (MINs), latency is caused by delays in the movement of data packets through several switching elements. Switching time and waiting delays add up as packets move through each stage, especially when there is a lot of traffic. High latency hurts the performance of multi-core systems by making it take longer for processors to talk to each other, which lowers speed and makes parallel processing jobs less efficient. To get the best data transfer speed in multi-core settings, you need to deal with delay.

4.2 Bandwidth Contention

Bandwidth contention happens when many data bits try to use the same MIN communication routes or switching elements at the same time. This competition slows down data transfers and makes bad use of available bandwidth, which causes connection problems between processors. Overall performance of multi-core systems is limited by bandwidth competition. This makes data transfer between processors slower, which makes it harder for the system to handle large-scale parallel jobs efficiently.

4.3 Congestion

Multiple data packets trying to reach the same switching element or route at the same time can cause congestion in MINs. This can cause packet delays and even data loss. This bottleneck hurts system performance a lot by making latency higher and output lower. Congestion slows down multi-core systems, especially when they're under a lot of traffic. To keep communication running smoothly and keep performance from dropping, you need effective congestion control methods.

4.4 Fault Tolerance

Fault tolerance is very important in MINs to make sure that communication stays stable even if hardware fails or transfer mistakes happen. Single points of failure in switching elements can stop communication routes, which can cause data loss or system downtime if there aren't strong fault tolerance methods in place. Today's solutions include multiple routes, error correction, and adaptable rerouting methods. These help keep data security and communication reliability high, which makes multi-core systems reliable in real-world situations.

4.5 Scalability

When multi-core systems are expanded, scalability problems show up because adding more processors and communication lines makes MINs more complicated. It's harder to keep communication running smoothly on larger systems because of higher delay, overcrowding, and bandwidth contention. Adding more moving parts also raises costs and makes managing the network harder. For MINs to work in big, high-performance computer systems, they need to be able to overcome scaling problems that make interactions between processors less efficient.

5. Proposed Methodology

A. Optimized Routing Algorithms:

The goal of optimized routing algorithms is to minimize latency and congestion in Multistage Interconnection Networks (MINs). This model outlines a 6-step approach to achieving optimal routing in multi-core systems.

Assume an MIN with N processors, consisting of S stages. Each stage has (N/2) switching elements (SEs). The total number of stages (S) is:

$$S = log2(N)$$

The network structure will be used to define the routing paths for data packets.

For each possible route from source (P_s) to destination (P_d), calculate the path cost (C_path), which includes latency (L) and congestion (C) at each switching element:

$$C\_path = sum(L\_i + C\_i)\ for\ i = 1\ to\ S$$

Where L_i represents the latency at the i-th stage and C_i is the congestion factor at the i-th switching element. Using Dijkstra's algorithm or the Bellman-Ford algorithm, find the path with the minimum C_path between the source and destination. The routing algorithm aims to minimize C_path by selecting the route with the lowest combined latency and congestion.

After determining the optimal path, update the congestion factor (C_i) for each switching element along the selected path:

$$C\_i(new) = C\_i(old) + alpha$$

Where alpha is a constant representing the increase in congestion due to the recent data transfer. This step ensures that future routing decisions account for the updated network state.

In the event of a failure in any switching element, implement a rerouting mechanism that dynamically recalculates the optimal path. Use an alternative route that has the next lowest C_path while ensuring communication continues:

$$C\_path(new) = min(C\_path(alternative\ routes))$$

To assess the performance of the optimized routing algorithm, calculate the average latency (L_avg) and congestion level (C_avg) across all active paths:

$$L_{avg} = \left(\frac{1}{M}\right) * sum\left(L_{path} for\ all\ M\ active\ paths\right)$$

$$C_{avg} = \left(\frac{1}{M}\right) * sum\left(C_{path} for\ all\ M\ active\ paths\right)$$

Where M represents the total number of active data transfers. The algorithm is considered optimized if both L_avg and C_avg remain minimized over time.

This 6-step model provides a systematic approach to routing data packets through an MIN while minimizing latency and congestion, ensuring efficient inter-processor communication in multi-core systems.

B. Adaptive Congestion Control Mechanisms:

The purpose of adaptive congestion control mechanisms is to dynamically manage congestion in Multistage Interconnection Networks (MINs). This model outlines a 6-step approach to effectively control congestion.

Continuously monitor the packet arrival rate (λ) and service rate (μ) at each switching element (SE). Calculate the traffic intensity (ρ) for each SE:

$$\rho = \frac{\lambda}{\mu}$$

If ρ > 1, the SE is experiencing congestion, indicating that the arrival rate exceeds the service capacity.

Determine the probability of congestion (P_congestion) at each SE using the formula:

$$P_{congestion} = 1 - e^{-\frac{\lambda}{\mu}}$$

This probability provides an estimate of how likely an SE will become a bottleneck in the network.

To alleviate congestion, dynamically adjust the packet transmission rate (R) for each active communication path:

$$R(new) = R(old) \times \left(1 - \beta \times P_{congestion}\right)$$

Where β is an adaptation factor (0 < β < 1) that controls the degree of rate adjustment based on the congestion probability.

Assign priority levels (P_level) to data packets based on urgency. For packets with higher priority, ensure they receive access to less congested paths:

$$P_{level(i)} = \frac{1}{1 + P_{congestion(i)}}$$

Higher priority is given to paths with lower congestion probabilities, ensuring critical data reaches its destination faster.

Redistribute traffic across multiple paths by adjusting the flow rate (F) of each path:

$$F(new) = F(old) \times \left(1 - \gamma \times P_{congestion}\right)$$

Where γ is a load balancing factor ($0 < \gamma < 1$). This step reduces the traffic load on congested paths, promoting efficient network utilization.

Calculate the average congestion level (C_avg) and average packet delay (D_avg) across all switching elements to evaluate the efficiency of the adaptive congestion control mechanism:

$$C_{avg} = \left(\frac{1}{N}\right) \times sum\left(P_{congestion} for\ all\ N\ SEs\right)$$

$$D_{avg} = \left(\frac{1}{M}\right) \times sum(L\ for\ all\ M\ data\ packets)$$

Where L is the delay experienced by each packet. The control mechanism is effective if both C_avg and D_avg remain minimized.

6. Results and Discussion

Table 2 shows an interesting study of how different routing methods and network designs affect important factors in Multistage Interconnection Networks (MINs). The data shows that Adaptive Routing and the Clos Topology work better than all other options we tested. This makes them the best picks for improving communication between processors. The average latency for Adaptive Routing (9.2 ms) and the Clos Topology (8.3 ms) is much lower than for other topologies, which shows that they can reduce transmission delays. This lower delay is due to the dynamic path selection of Adaptive Routing, which stays away from busy routes, and the various pathways of the Clos Topology, which provides effective communication channels. On the other hand, Deterministic Routing and the Butterfly Topology have longer delay, which means they handle data less efficiently.

Table 2: Impact of different routing algorithms and network topologies on different parameters

Parameter	Deterministic Routing	Adaptive Routing	Omega Topology	Butterfly Topology	Clos Topology
Average Latency (ms)	15.4	9.2	12.6	14.8	8.3
Congestion Probability (%)	35	18	28	33	15
Bandwidth Utilization (%)	60	80	72	65	85

Packet Loss Rate (%)	4.2	1.8	3.5	3.9	1.5
Throughput (Gbps)	5.6	8.1	6.3	5.9	9.0

The chances of congestion are also much lower for the Clos Topology (15%) and Adaptive Routing (18%), which shows that they are good at handling traffic and avoiding jams. This is different from Deterministic Routing (35% of the time) and the Butterfly Topology (33% of the time), which have higher delay rates that can make communication much less efficient, as shown in figure 3.

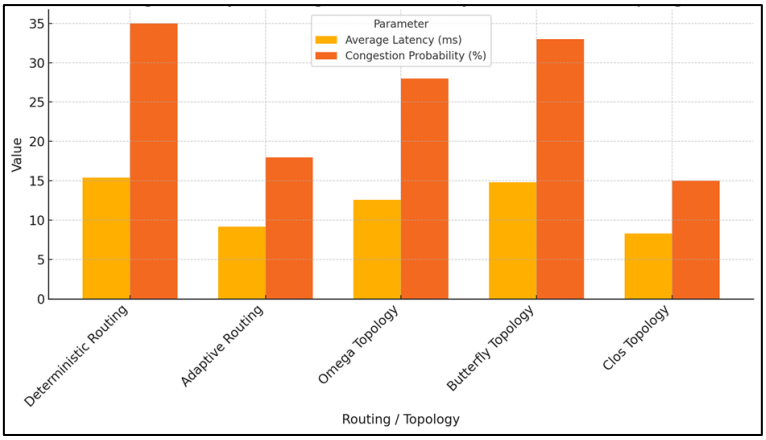

Figure 3: Average Latency and Congestion Probability Across Different Topologies

Clos Topology (85%) and Adaptive Routing (80%) have the best bandwidth usage, which means they make the best use of available bandwidth and improve system speed as a whole. With Adaptive Routing at 1.8% and the Clos Topology at 1.5%, these methods also have very low packet loss rates. This makes data sharing more stable compared to other methods that have higher packet loss rates, illustrate in figure 4.

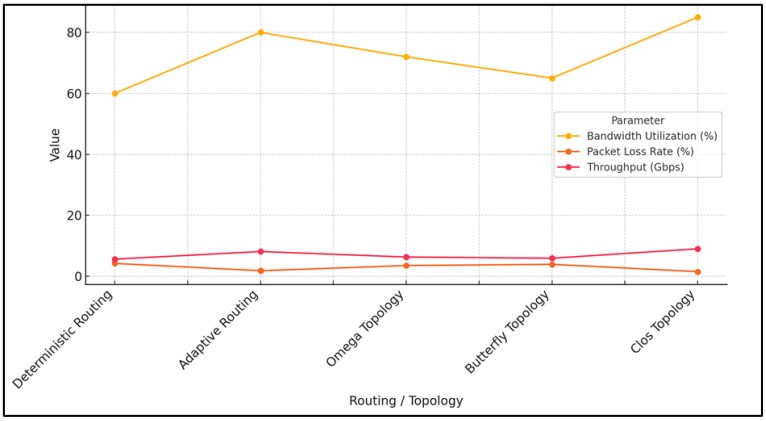

Figure 4: Comparison of Bandwidth Utilization, Packet Loss Rate, and Throughput Across Different Topologies

Table 3: Results of bandwidth efficiency in various Multistage Interconnection Network (MIN) architectures

Parameter	Butterfly Network	Omega Network	Clos Network	Hybrid Network
Bandwidth Utilization (%)	65	70	85	80
Scalability (Max Processors)	128	256	512	1024
Throughput (Gbps)	6.0	7.5	9.8	9.2
Latency (ms)	14.2	12.8	9.5	10.3
Fault Tolerance (%)	60	65	85	80

Table 3 shows a comparison of how well different Multistage Interconnection Network (MIN) designs use bandwidth. The Butterfly, Omega, Clos, and Hybrid networks are shown. The outcomes show the pros and cons of each design in terms of fault tolerance, bandwidth usage, scaling, speed, and delay. This gives a full picture of how well they work in systems with multiple cores. Bandwidth Utilization is a key indicator of how well a network works, and the Clos Network has the best score in this area with an amazing 85% utilization, which means that communication is very quick. The Omega and Butterfly Networks are far behind, at 70% and 65%, respectively. The Hybrid Network is close behind, at 80%. This means that the Clos and Hybrid Networks can better handle and make the most of available bandwidth. This makes them better for apps that use a lot of data and need to be efficient with their resources, comparison shown in figure 5.

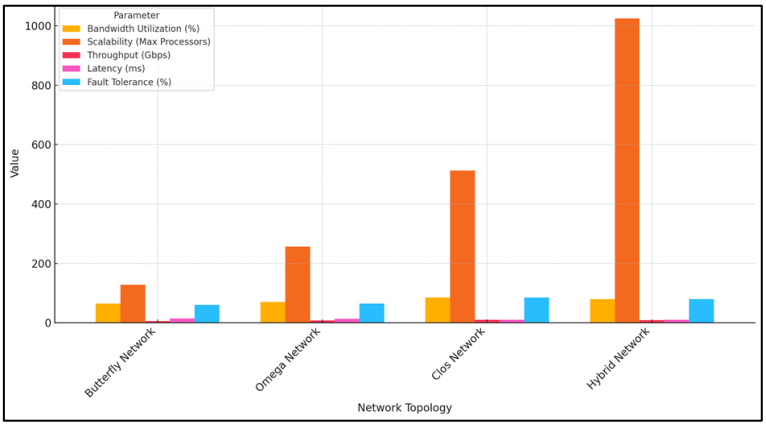

Figure 5: Comparison of Network Parameters Across Different Network Topologies

The Hybrid Network has a big edge when it comes to Scalability. It can handle up to 1024 processors, which makes it the most flexible for large, high-performance computer systems. The Clos Network can still handle up to 512 processors, which is more than the Omega Network's 256 processors and the Butterfly Network's 128 processors. This means that as the system gets bigger, the Hybrid and Clos Networks are better at keeping communication running smoothly, while the Butterfly and Omega Networks might have trouble keeping up with the extra work. Another important factor is speed. The Clos Network has the best throughput at 9.8 Gbps, which shows that it can handle big amounts of data quickly. With 9.2 Gbps, the Hybrid Network also works well. The

Omega and Butterfly Networks, on the other hand, have slower speeds of 7.5 Gbps and 6.0 Gbps, respectively. The Clos and Hybrid Networks' higher rate shows how well they work for high-speed data transfer, which makes them better options for uses that need to send large amounts of data quickly.

Table 4: Comparative analysis of different network topologies and their Inter-Processor Communication (IPC) efficiency

Parameter	Butterfly Network	Omega Network	Clos Network	Hybrid Network
IPC Efficiency (%)	70	75	90	88
Average Latency (ms)	13.5	12.0	8.0	9.5
Throughput (Gbps)	5.8	6.5	9.0	8.5
Scalability (Max Processors)	128	256	512	1024
Fault Tolerance (%)	60	65	85	80

Table 4 shows a comparison of various network designs and how well they support Inter-Processor Communication (IPC). At 90%, the Clos Network has the best IPC efficiency, the lowest average delay (8.0 ms), and the highest speed (9.0 Gbps).

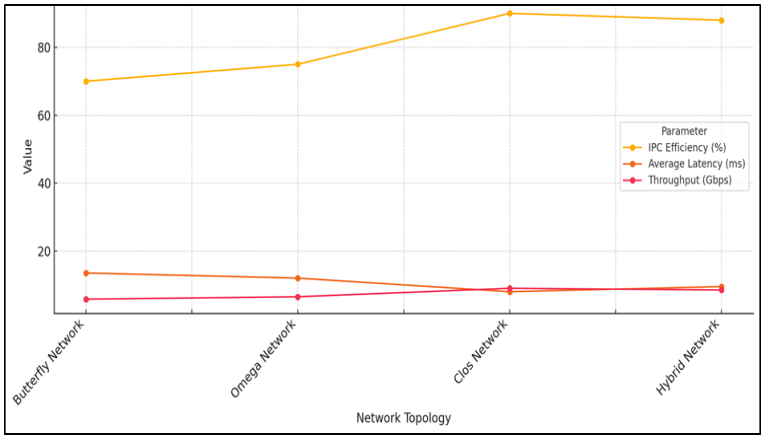

Figure 6: IPC Efficiency, Average Latency, and Throughput across Different Network Topologies

This makes it the best choice for high-performance computing. The Hybrid Network is very close behind, with 88% IPC efficiency. It also has great scaling, allowing up to 1024 processors, which makes it good for large-scale apps. The Omega Network has average performance, with an IPC efficiency of 75% and a reasonable delay of 12.0 ms, as shown in figure 6. The Butterfly Network, on the other hand, has the worst IPC efficiency (70%) and fault tolerance (60%). The Clos and Hybrid Networks are better than others in terms of speed, scalability, and fault tolerance. This makes them perfect for high-end systems with multiple cores.

7. Conclusion

A study of the performance of multi-core systems in multistage interconnection networks (MINs) shows how important it is for processors to be able to talk to each other effectively in order for the system to work at its best. This research looked at different routing methods and network designs, like Butterfly, Omega, Clos, and Hybrid Networks, to find the most important things that affect how well IPC works. These components incorporate delay, clog, transfer speed utilization, speed, and blame resistance. The comes about appear that the Clos and Cross breed Systems do superior than others in terms of IPC effectiveness, adaptability, and blame resilience. This implies that these systems are the finest picks for circumstances with a part of handling control. When compared to settled strategies, versatile steering calculations significantly diminish delay and blockage, appearing that they are great at taking care of information stream adaptably inside MINs. This adaptability makes beyond any doubt that communication works well indeed when activity loads alter, which is exceptionally critical for keeping framework execution tall in huge multi-core frameworks. With its non-blocking plan and different course ways, the Clos Arrange continually gives superior transfer speed utilization and speed. On the other hand, the Half breed Organize is much more versatile and can handle more computers. But issues like over-burdening, longer hold up times, and taking care of blunders are still enormous issues, particularly in greater frameworks. Routing methods, congestion control mechanisms, and fault-tolerant systems need to be improved all the time to make sure that transmission is stable and works well.

References

[1] Gupta, S.; Pahuja, G.L. SEGIN-Minus: A New Approach to Design Reliable and Fault-Tolerant MIN. Recent Adv. Comput. Sci. Commun. Former. Recent Patents Comput. Sci. 2020, 13, 370–380. [Google Scholar] [CrossRef]

[2] Amodu, O.A.; Othman, M.; Yunus, N.A.M.; Hanapi, Z.M. A Primer on Design Aspects and Recent Advances in Shuffle Exchange Multistage Interconnection Networks. Symmetry 2021, 13, 378. https://doi.org/10.3390/sym13030378

[3] Mnejja, S.; Aydi, Y.; Abid, M.; Monteleone, S.; Catania, V.; Palesi, M.; Patti, D. Delta Multi-Stage Interconnection Networks for Scalable Wireless On-Chip Communication. Electronics 2020, 9, 913. https://doi.org/10.3390/electronics9060913.

[4] Aydi, Y.; Mnejja, S.; Mohammed, F.Q.; Abid, M. Mapping of Deep Neural Network Accelerators on Wireless Multistage Interconnection NoCs. Appl. Sci. 2024, 14, 56. https://doi.org/10.3390/app14010056

[5] Hammami, O.; M'zah, A.; Hamwi, K. Design of 3D-IC for butterfly NOC based 64 PE-multicore: Analysis and design space exploration. In Proceedings of the IEEE International 3D Systems Integration Conference (3DIC)—2011 IEEE International, Osaka, Japan, 31 January–2 February 2012; pp. 1–4.

[6] Hammami, O.; M'zah, A.; Jabbar, M.H.; Houzet, D. 3D IC Implementation for MPSOC architectures: Mesh and butterfly based NoC. In Proceedings of the 4th Asia Symposium on Quality Electronic Design (ASQED), Penang, Malaysia, 10–11 July 2012; pp. 155–159.

[7] Swaminathan, K.; Thakyal, D.; Nambiar, S.G.; Lakshminarayanan, G.; Ko, S.B. Enhanced Noxim simulator for performance evaluation of Network-on-chip topologies. In Proceedings of the Recent Advances in Engineering and Computational Sciences (RAECS), Chandigarh, India, 6–8 March 2014; pp. 1–5.

[8] Zhang, D.; Liu, Z.; Jia, W.; Liu, H.; Tan, J. Path Enhanced Bidirectional Graph Attention Network for Quality Prediction in Multistage Manufacturing Process. IEEE Trans. Ind. Inform. 2020, 18, 1018–1027.

[9] Suzumura, T.; Zhou, Y.; Barcardo, N.; Ye, G.; Houck, K.; Kawahara, R.; Anwar, A.; Stavarache, L.; Klyashtorny, D.; Ludwig, H.; et al. Towards Federated Graph Learning for Collaborative Financial Crimes Detection. arXiv 2019, arXiv:1909.12946.

[10] Robinson, D.; Scogings, C. The detection of criminal groups in real-world fused data: Using the graph-mining algorithm, "GraphExtract". Secur. Inform. 2018, 7, 2.

[11] Fensel, A.; Akbar, Z.; Kärle, E.; Blank, C.; Pixner, P.; Gruber, A. Knowledge Graphs for Online Marketing and Sales of Touristic Services. Information 2020, 11, 253.
[12] Gepner, P. Machine Learning and High-Performance Computing Hybrid Systems, a New Way of Performance Acceleration in Engineering and Scientific Applications. In Proceedings of the 16th Conference on Computer Science and Intelligence Systems, Online, 2–5 September 2021; pp. 27–36.
[13] Migallón, H.; Migallón, V.; Penadés, J. Non-Stationary Acceleration Strategies for PageRank Computing. Mathematics 2019, 7, 911.
[14] Nagasinghe, I. Computing Principal Eigenvectors of Large Web Graphs: Algorithms and Accelerations Related to Pagerank and Hits. Ph.D. Dissertation, Southern Methodist University, Dallas, TX, USA, 2010; pp. 1–114.
[15] Liu, C.; Li, Y. A Parallel PageRank Algorithm with Power Iteration Acceleration. Int. J. Grid Distrib. Comput. 2015, 8, 273–284.
[16] Migallón, H.; Migallón, V.; Penadés, J. Parallel two-stage algorithms for solving the PageRank problem. Adv. Eng. Softw. 2018, 125, 188–199.
[17] Jiang, Z.; Liu, T.; Zhang, S.; Guan, Z.; Yuan, M.; You, H. Fast and efficient parallel breadth-first search with power-law graph transformation. arXiv 2020, arXiv:2012.10026.
[18] Ganzha, M.; Georgiev, K.; Lirkov, I.; Paprzycki, M. An application of the partition method for solving 3D Stokes equation. Comput. Math. Appl. 2015, 70, 2762–2772.
[19] Bernard, F.; Zheng, Y.; Joubert, A.; Bhatia, S. High Performance Graph Analytics on Graphcore IPUs. In Proceedings of the 2021 IEEE International Conference on High Performance Computing (HiPC), Bengaluru, India, 17–20 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 30–41.
[20] Kim, H.; Ben-Othman, J.; Mokdad, L.; Son, J.; Li, C. Research Challenges and Security Threats to AI-Driven 5G Virtual Emotion Applications Using Autonomous Vehicles, Drones, and Smart Devices. IEEE Netw. 2020, 34, 288–294. [Google Scholar] [CrossRef]
[21] Loke, S.W. Cooperative automated vehicles: A review of opportunities and challenges in socially intelligent vehicles beyond networking. IEEE Trans. Intell. Veh. 2019, 4, 509–518.
[22] Liu, Z.; Lee, H.; Khyam, M.O.; He, J.; Pesch, D.; Moessner, K.; Saad, W.; Poor, H.V. 6G for vehicle-to-everything (V2X) communications: Enabling technologies, challenges, and opportunities. arXiv 2020, arXiv:2012.07753.
[23] Ekaterina Katya, "Game Theory Applications in Network Engineering: Mathematical Insights and Practical Solutions", MathEngage: Engineering Mathematics and Applications Journal, Volume 1 Issue 1, pp: 12-22, 2024.
[24] Dr. S. A. Sivakumar, "Artificial Intelligence and Machine Learning Algorithms in Engineering: A Mathematical Perspective", MathInnoTech: Innovations in Engineering Mathematics Journal, Volume 1 Issue 1, pp: 22-33, 2024.