UNIVERSITY OF L'ARBI BEN M'HIDI OUM EL BOUAGHI

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

DOCTORAL THESIS

The Development of Reliable Routing Algorithms for Network on Chip

---

Submitted in partial fulfilment of the requirements

for the degree of Doctor of Philosophy in Computer Science

by

Habib Chawki Touati

Members of the jury:

Prof. NINI Brahim	University of Oum El Bouaghi	President
Prof. BOUTEKKOUK Fateh	University of Oum El Bouaghi	Supervisor
Prof. BELHADEF Hacene	University of Constantine 2	Examiner
Dr. LAABOUDI Zakaria (MCA)	University of Oum El Bouaghi	Examiner

March, 2021

Abstract

With the ability to embed hundreds of cores on a single chip, solving communication related issues became of paramount importance. Bus-based architectures have been predominantly used as the principal interconnect for Systems on Chip, nonetheless, their complexity increases drastically with the integration of a large number of cores and traffic saturation happens rather quickly. Crossbars have equally served as on-chip interconnects, yet they still scale poorly, requiring a large area footprint while consuming high amounts of energy.

To overcome the limited scalability of the aforementioned interconnects and in order to tackle the complex problems and demands of future System on Chip designs, as well as the numerous communication challenges such as wiring complexity, communication delays and power consumption, Network on Chip was proposed.

Inspired by the great success of switch-based networks and packet-based communication, wherein cores are interconnected via bidirectional channels and exchange data in the form of packets, transmitted by means of routers. It alleviates some of the major issues of bus-based systems, such as scalability, modularity and reusability.

The backbone of the communication fabric is the routing algorithm, which is in charge of forwarding data packets along the most appropriate routes to their eventual destination. It plays a vital role confronting numerous issues that contribute to performance deterioration.

Congestion happens to be one of the predominant problems that contribute to the obstruction of data packets and the increase of communication delays. The most effective way to limit its impact is through the implementation of a reliable,

congestion aware routing algorithm, capable of adapting to the ever-changing congestion conditions of the network and able to select the less congested routes for the transmission of data packets.

In this thesis, we analyse and evaluate the performance of an extensive list of on-chip routing schemes and extract the features that we believe are most beneficial in the design of reliable, congestion-aware routing algorithms for Network on Chip. We then present our novel congestion propagation network which enables a global view, that will be exploited by our proposed minimal fully adaptive, congestion aware routing algorithms for mesh-based Network on Chip.

The algorithms strike a balance between locally and globally congestion aware routing, in that they do not rely solely on local congestion information nor on irrelevant global congestion, while still offering a fair comparison along the entire way of data packets. For that, we propose a weight distribution technique as well as a categorization protocol based on the congestion levels of network nodes. The various experimental results showcase the proposed algorithms' effectiveness over state of the art on-chip routing algorithms with regard to average packet latency and average accepted packet rate, under the different system configurations.

Keywords

Network on Chip, Communication, Routing algorithms, Adaptivity, Congestion awareness.

Résumé

Avec la possibilité d'intégrer des centaines de cœurs sur une seule puce, la résolution des problèmes liés à la communication est devenue d'une importance capitale. Les architectures basées sur bus ont été principalement utilisées par les systèmes sur puce, mais leur complexité augmente considérablement avec l'intégration d'un grand nombre de cœurs et la saturation du trafic se produit assez rapidement.

Les crossbars ont également servi d'interconnexions sur puce, mais elles nécessitent de grande surface et consomment de grandes quantités d'énergie.

Pour surmonter la scalabilité limitée des interconnexions susmentionnées et pour résoudre les problèmes complexes et les exigences des futurs systèmes sur puce, ainsi que les nombreux défis de communication tels que la complexité du câblage, la latence et la consommation d'énergie, les réseaux sur puce ont été proposés.

Inspiré par le grand succès des réseaux commutés et de la communication par paquets, où les cœurs sont interconnectés via des canaux bidirectionnels et échangent des données sous forme de paquets, transmis au moyen de routeurs. Il atténue certains des problèmes majeurs des systèmes basés sur bus, tels que la scalabilité, la modularité et la réutilisabilité.

L'épine dorsale de la communication est l'algorithme de routage, qui est chargé de transmettre les paquets de données le long des routes les plus appropriées vers leur destination finale. Il joue un rôle essentiel face à de nombreux problèmes qui contribuent à la détérioration des performances.

La congestion est l'un des problèmes prédominants qui contribuent à l'obstruction des paquets de données et à l'augmentation des délais de communication. Le

moyen le plus efficace de limiter son impact est la mise en œuvre d'un algorithme de routage fiable et sensible à la congestion, adaptatif et capable de sélectionner les routes les moins encombrées pour la transmission des paquets de données.

Dans cette thèse, nous analysons et évaluons les performances de plusieurs algorithmes de routage sur puce et extrayons les caractéristiques que nous pensons les plus bénéfiques dans la conception des algorithmes de routage fiables et sensibles à la congestion pour les réseaux sur puce. Nous présentons ensuite notre nouveau réseau de propagation de congestion qui permet une vue globale du réseau, qui sera exploitée ensuite par nos algorithmes de routage minimal et entièrement adaptatifs proposés.

Les algorithmes proposés ne s'appuient pas uniquement sur des informations de congestion locale ni sur la congestion globale non pertinente, et offrent une comparaison équitable sur tout le trajet des paquets de données. Pour cela, nous proposons une technique de répartition des poids ainsi qu'un protocole de catégorisation basé sur les niveaux de congestion des nœuds du réseau. Les différents résultats expérimentaux montrent l'efficacité des algorithmes proposés par rapport aux autres algorithmes de routage sur puce.

Mots clés

Réseaux sur puce, Communication, Algorithmes de routage, Adaptivité, Congestion.

ملخص

مع القدرة على دمج مئات النوى في شريحة واحدة، أصبح حل المشكلات المتعلقة بالاتصال ذا أهمية قصوى داخل الأنظمة القائمة على الرقاقة.

تم في السابق استخدام العديد من البنى الأساسية للربط بين الأنوية، ولكن تعقيدها يزداد بشكل كبير مع تزايد عدد النوى حيث تتشبع بسرعة كبيرة، كما أنها تتطلب مساحة كبيرة وتستهلك كميات كبيرة من الطاقة.

للتغلب على قابلية التوسع المحدودة للبنيات الأساسية المذكورة أعلاه ومن أجل معالجة المشاكل المعقدة ومتطلبات تصميمات الأنظمة المستقبلية، بالإضافة إلى تحديات الاتصال العديدة مثل تعقيد الأسلاك ووقت الإستجابة واستهلاك الطاقة، تم اقتراح الشبكات القائمة على الرقاقة.

مستوحاة من النجاح الكبير للشبكات القائمة على التبديل والاتصالات القائمة على الحزم، حيث يتم ربط النوى عبر قنوات ثنائية الاتجاه و يتم تبادل البيانات في شكل حزم يتم إرسالها عن طريق أجهزة التوجيه. تقوم بالتخفيف من بعض المشكلات الرئيسية للأنظمة التقليدية، مثل قابلية التوسع والنمطية وإعادة الاستخدام.

العمود الفقري لنسيج اتصالات الشبكة هو خوارزمية التوجيه، حيث أنها المسؤولة عن توجيه حزم البيانات على طول المسارات الأكثر ملاءمة إلى وجهتها النهائية.

تلعب خوارزميات التوجيه دورا حيويا في مواجهة العديد من المشكلات التي تساهم في تدهور الأداء. يعتبر الازدحام أحد الأسباب الرئيسية لذلك، حيث يساهم في إعاقة حزم البيانات وزيادة زمن الوصول الإجمالي. الطريقة الأكثر فعالية للحد من تأثيرها هي من خلال تنفيذ خوارزمية توجيه موثوقة، قادرة على التكيف مع ظروف الازدحام المتغيرة باستمرار وقادرة على تحديد المسارات الأقل ازدحاما لنقل حزم البيانات.

في هذه الأطروحة، نقوم بتحليل وتقييم أداء قائمة واسعة من خوارزميات

التوجيه كما نقوم باستخراج الميزات التي نعتقد أنها الأكثر فائدة في تصميم خوارزميات توجيه موثوقة ومتكيفة تماما ومدركة للازدحام في الشبكات القائمة على الرقاقة. نقدم بعد ذلك شبكة نشر معلومات الازدحام الخاصة بنا والتي تتيح رؤية شاملة يتم استغلالها فيما بعد من طرف خوارزميات التوجيه المقترحة.

هذه الأخيرة لا تعتمد على معلومات الازدحام المحلي فقط ولا على الازدحام الشامل الغير ذي الصلة وتوفر إلى حد ما حلا وسطا بينهما، مع ضمان مقارنة عادلة على طول الطريق الكامل لحزم البيانات. نقوم كذلك باقتراح تقنية لتوزيع الأوزان بالإضافة إلى بروتوكول تصنيف عقد الشبكة بناءً على مستويات الازدحام.

تبين النتائج التجريبية المختلفة فعالية الخوارزميات المقترحة بالمقارنة مع الخوارزميات الأخرى بناء على متوسط زمن انتقال الحزم ومعدل الحزم المقبول.

الكلمات المفتاحية

الشبكات القائمة على الرقاقة، الاتصالات، خوارزميات التوجيه، التكيف، الوعي بالازدحام.

This thesis is dedicated to my beloved parents.

For their unconditional love, patience and never-ending support.

Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Fateh Boutekkouk for his continuous guidance, patience and above all, his impeccable work ethic and sincerity.

I have had the opportunity to do a research visit at the cybernetics & decision support systems laboratory at the university of Maribor. I would like to thank Prof. Andrej Škraba for the invitation, hospitality and countless constructive discussions.

I would also like to thank gentlemen of the jury, Prof. Brahim Nini, Prof. Hacene Belhadef and Dr. Zakaria Laaboudi for their constructive comments and suggestions.

Last but most importantly, i deeply thank my beloved parents for their trust, everlasting encouragement and endless patience. It has been a long bumpy ride with many valuable life experiences and this work would be virtually impossible if it weren't for you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and background

In order to refine the overall performance of a chip, manufacturers were traditionally inclined to increase the clock rate. Nonetheless, high clock rates lead to high power dissipation, which boosts the failure rate of system components and impact its reliability [1].

Because this solution is impractical, instead of implementing a single complex monolithic processor running at high frequencies, chip designers opted for integrating several cores running at lower frequencies on a single chip. This multi-core architecture provides several advantages including the fact that each processor core runs at its own supply voltage and frequency and can be turned on or off, to save power. Heat on the other hand can be distributed across the die which potentially produces lower temperatures and hence improving reliability [2, 3].

The design of such systems was made possible due to the increasing number of transistors that can be embedded on a single chip thanks to the continuous shrinking of sizes [4].

While early systems were developed following a homogeneous architecture, most recent ones are heterogeneous in nature, composed of various modules (processors, memories, input/output peripherals) on a single chip referred to as Systems on Chip (SoC).

1.1.1 Systems on Chip

To meet the ever increasing demands of state of the art embedded and consumer electronic devices, Systems on Chip have evolved from simple uni-core single memory designs to complex multi-core systems (MCSoC) [5]. They are made of a large number of intellectual property cores from a variety of vendors with standard interfaces such as display controllers, camera interfaces, sensors, connectivity modules and multimedia subsystems [6, 7].

To efficiently take full advantage of the increasing number of integrated cores, communication is becoming more relevant than computation and the interconnection starts to play a vital role in determining the performance and power consumption of the chip [8, 9].

Shared buses and crossbars are the dominant interconnect structures for simple SoCs that comprise low core counts [10, 11].

On one hand, buses represent shared bidirectional multi-bit physical channels that connect intellectual property cores (IP) and allow them to send or receive information with a single core at a time (Figure 1.1).

To start transmitting, a sender first requests access to the bus. It then broadcasts messages after being granted access among all requesters. The message will then be accepted by the intended receiver and ignored by all others.

Buses allow an easy integration of additional cores and provide low latency. However, due to the centralized architecture, they are unable to handle multiple communication flows in parallel and only scale to a small number of cores

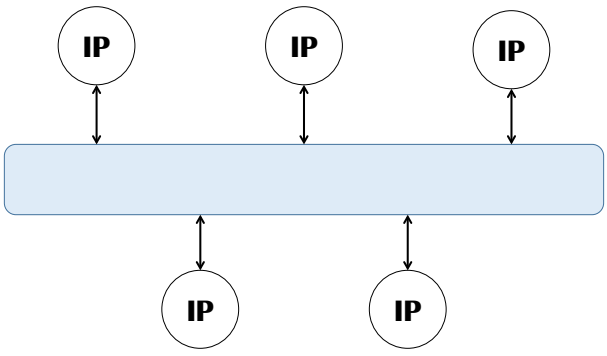


Figure 1.1: Shared bus architecture

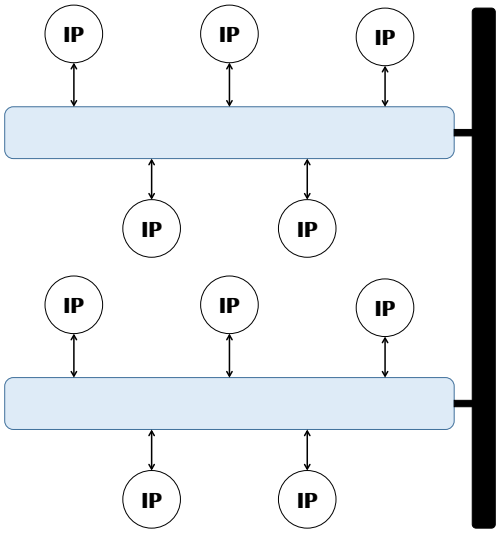


Figure 1.2: Hierarchical bus architecture

because traffic reaches saturation quickly. Hence, achieving high bandwidth becomes rather difficult.

Although bus architectures have evolved significantly from a single shared-bus topology to bridged and multilayer buses (Figure 1.2), the use of several levels of bus hierarchies interconnected by means of bridges is quite challenging to design [11].

Crossbars on the other hand, provide non-blocking connectivity between any pair of communicating cores and offer high bandwidth and relatively low latency. Nevertheless, they too scale poorly for a large number of cores requiring a large area footprint and consuming a considerable amount of power.

As the number of on-chip cores keeps increasing and because of the poor scalability of these communication designs, a scalable, low latency and high bandwidth communication fabric became of the utmost importance [12].

1.1.2 Networks on Chip

Network-on-Chip (NoC) was proposed to effectively replace the traditional interconnects in today's complex many-core systems and is becoming the de facto standard for communication. It offers a scalable solution to the integration challenges of modern SoCs by applying well established networking principles at the

chip level. It also provides a distributed architecture that can be shared by multiple traffic flows in parallel leading to high performance. Its modular nature enables the separation of communication from computation, thus allowing the integration of additional cores straightforwardly [13, 14, 15] .

Due to its ability to supply scalable bandwidth at low area and power, it is considered more cost-effective than conventional interconnects, which suffer from large area and power overheads [16].

Network-on-Chip is viewed as a system that facilitates data transmission between nodes. The latter are connected via a packet switching communication fabric on a hop-by-hop basis, similar to the way computers are connected to the internet. Transmission can be conducted in parallel if the network provides more than one channel between the pair of communicating nodes [17, 18].

NoCs integrate several essential components such as channels, network interfaces and routers [19, 20].

Channels (links) are responsible for connecting routers and allowing the exchange of data between them. They can be either bi-directional or uni-directional and might be composed of several physical or logical channels.

Network interfaces on the other hand are the medium between cores and routers. They handle the packetization of data coming from the core prior to being injected into the network and reassemble them to their initial state upon their arrival to their destination node. The use of network interfaces hides the implementation details of the communication structure in such a way that the network will have no information about the attached core and vice versa.

Last but not least, routers are the heart of the network and their functions include forwarding data, switching and flow control [21]. All of which will be discussed in detail in the upcoming chapter.

A Network on Chip displaying the separation of computation and communication is depicted in Figure 1.3, with routers (R), intellectual property cores (IP) and network interfaces (NI).

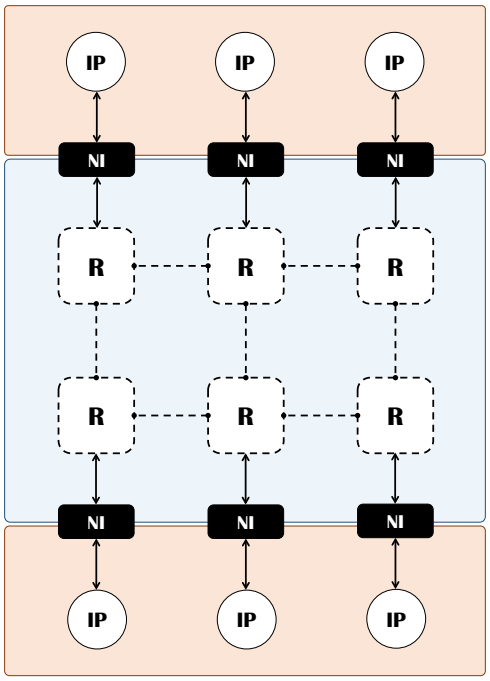


Figure 1.3: Network on Chip

1.2 Problems and motivation

Like any other high-performance interconnection network, on-chip routing has a significant impact on the overall system performance [22, 23].

One of the most complex routing problems to address is congestion, which occurs when network resources are overloaded and bandwidth is lower than the one requested. The blocking of data packets marks the main cause of performance deterioration in a congested network setting which is aggravated when the network size increases [24].

Packets passing through hot spots (congested nodes) are blocked at the head of buffer queues, obstructing other packets, even if the latter are not passing through the same nodes, which increases communication delays.

In order to avoid the blocking of data packets, certain rules imposed by routing algorithms must be adhered to, which determine the path that messages will take from a source to a destination node. Some routing algorithms limit packets adaptivity, forcing them to transit specific predetermined paths, whereas others distribute them across all possible routes in the network, achieving higher

performance and lower communication delays under highly loaded conditions. These algorithms are preferred to avoid or minimize congestion effects.

They can be locally aware of congestion by taking into consideration the information of neighbour nodes and will not be able to avoid remote congested regions in the network. They can alternatively be globally aware by also taking distant nodes into account and hence be able to make more effective decisions. Nevertheless, putting too much emphasis on remote congestion, will inevitably introduce too much noise in the decision-making process by taking into account the information of unreachable nodes. Therefore a design trade-off is required.

The design of adaptive and reliable algorithms that strike a balance between locally and globally congestion-aware routing, form the main contributions of this thesis.

1.3 Contributions

This thesis includes the following contributions:

- Analysis and performance evaluation of various adaptive, deterministic and stochastic routing schemes, with regard to average packet latency and average accepted packet rate by applying a set of traffic patterns, several of which are based on communication patterns that arise in real world applications. In addition to conducting a set of experiments to compare between 2D and 3D based on-chip networks.
- We constructed a congestion propagation network that enables a full view of the network, which allows the inclusion of global congestion information in the decision making process rather than relying solely on locally collected congestion information.
- We implemented a minimal fully adaptive congestion aware routing algorithm for mesh-based network on chip. The scheme makes use of our congestion propagation network and does not take irrelevant congestion information into account while offering a fair comparison along the entire way of data packets. We proposed two distinct variants, each relies on a different weight distribution mechanism.

- We implemented yet another minimal fully adaptive congestion aware routing algorithm that, instead of relying on a weight distribution mechanism, it categorizes network nodes based on their congestion levels. Two variants were also proposed, each with its own categorization protocol. Additionally, it provides a compromise between locally and globally aware routing, wherein congestion in parts of the network close, not only to the source, but also to the destination node is investigated.

1.4 Thesis outline

The thesis is outlined as follows:

- Chapter 2 revolves around presenting the essential elements and building blocks of networks on chip. We unveil the most relevant topologies and their characteristics, flow control mechanisms and buffer management techniques. We also introduce the micro-architecture of on-chip routers and their key constituents as well as their pipeline stages. Lastly, we take a look at the design requirements of on-chip routing, the main obstacles they encounter, in addition to their effect on the overall performance.
- Chapter 3 is all about the most relevant and related on-chip routing algorithms, their classes and attributes, as well as the set of objectives they aim to fulfil. We ultimately extract the features that we believe are the most rewarding in the design of reliable, adaptive, congestion aware routing. We then demonstrate the performance evaluation techniques and the essential metrics used to test the limits, efficiency and reliability of a given routing algorithm. We conduct a series of experiments to have a good understanding of the pros and cons of the various routing schemes and outline the interesting results.
- In Chapter 4 we introduce our novel, reliable, fully adaptive and congestion aware routing algorithms for mesh-based network on chip. We propose two moderately distinct versions for each algorithm, wherein we include both local and global congestion information in the decision-making process. Accordingly, we construct a congestion propagation network that

can be used not only in the context of our work but also as a standalone work. We eventually demonstrate the experimental results to showcase the efficiency of the proposed approaches and discuss their limitations and possible future enhancements.

- Chapter 5 concludes this thesis.

Chapter 2

On-Chip Networks

The incorporation of a considerable number of intellectual property cores on a single chip, shifted the focus from computation towards communication. A more reliable interconnection infrastructure was sought to handle the ever-increasing demands of complex System on Chip designs, especially since traditional infrastructures such as shared buses and point-to-point communication no longer serve this purpose due to their non-scalable nature [25, 13, 18].

Henceforth, Network on Chip will be utilized, as it represents a good candidate since it diminishes the wiring complexity, communication latency and power consumption issues [26]. The NoC fabric, inspired by the great success of packet-based communication and switch-based networks, consists of a set of nodes (a core connected to a router) interconnected by means of bidirectional links and data is propagated in the form of packets [15, 7].

In this chapter, we explore the main building blocks of on-chip networks, including the most pertinent topologies and their characteristics, routers' micro-architecture and their key constituents, flow control and buffer management techniques and conclude with a glance at routing (the backbone of communication), the various routing classes, their design requirements and the obstacles they face, as well as their effect on overall performance, before delving deeper into the world of routing schemes and related works in the next chapter.

2.1 Topology

The network topology governs the physical layout and arrangement of nodes, the way routers are interconnected and the available links for data transmission. It also determines the interconnect length, the number of alternative paths between nodes and the number of hops (routers) a message traverses, thus affecting many other aspects of the network such as routing, flow control and overall performance [27, 28].

Directness: topologies can be classified as either direct or indirect. In direct networks (the most utilized on chip), each core (terminal node) is associated with a router that acts as both a source and a destination for traffic. In an indirect network however, some routers act as intermediate nodes with the sole task of switching traffic to and from terminal nodes.

Regularity: topologies can additionally be classified as regular or irregular.

Regular topologies are suitable for homogeneous applications, where cores tend to have the same size and the same communication requirements and, routers tend to have the same number of ports. They are favoured mostly because of their scalability and reusable patterns.

On the other hand, irregular topologies are customized for a specific application to match its communication needs, their main drawback lies in the design time needed to outline the application, to decide the best layout.

Degree: a high topology degree, defined as the number of links at each node, implies a high router radix, defined as the number of ports. Accordingly, the implementation complexity increases, adding more area and energy overhead per router.

Diameter: the topology diameter is the maximum distance between any pair of nodes in the network. It affects the maximum latency in the absence of contention.

Hop count: hop count is defined as the number of hops a message takes from a source to a destination node, i.e., the number of links it traverses. The maximum hop count is inferred from the diameter.
The most pertinent on-chip topologies are further discussed in this section.

2.1.1 2D topologies

The crossbar is among the earliest and simplest indirect on-chip topologies, where n inputs are connected to m outputs via $n \times m$ simple crosspoint switch nodes [29]. It is non-blocking as it can always connect a sender to a unique receiver. A 4×4 crossbar is depicted in Figure 2.2a.

K-ary N-cube

The mesh and torus topologies are the most commonly used on chip. Both are direct networks and portrayed as k-ary n-cubes, where k is the number of nodes along each dimension and n is the number of dimensions [30, 31].

All nodes in a torus have the same degree, whereas in the case of mesh, nodes in the center of the network have a higher degree than those along the edges (each node is connected to at most four neighbours).

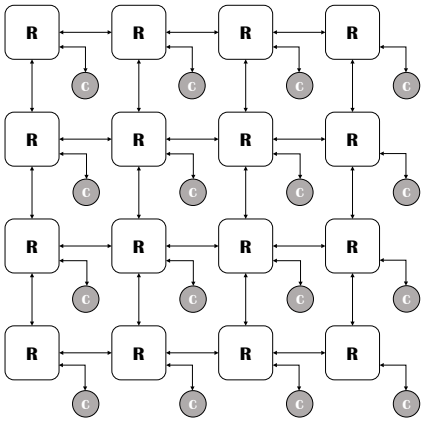
Both topologies account for the great majority of on-chip designs because of their regularity and scalability [32].

The mesh topology in particular is favoured because of its uniformity and schematic implementation of routing functions. Figure 2.1a depicts a 4×4 mesh topology.

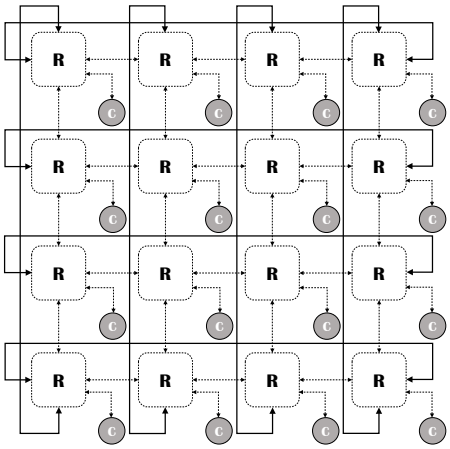
The Torus topology is a slightly adjusted mesh that offers more path diversity due to wraparound links connecting nodes on opposite edges. The links nonetheless, result in excessive delays due to their length. Figure 2.1b depicts a 4-ary 2-cube torus topology.

K-ary N-fly

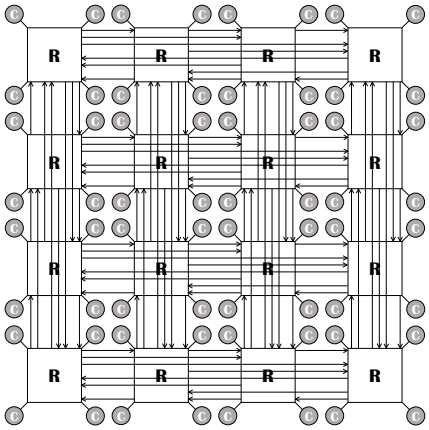
Butterflies are another form of indirect topologies described as k-ary n-fly. They consist of k^n source terminal nodes, k^n destination terminal nodes and n stages of k^{n-1} switch nodes. The channels are unidirectional where source and



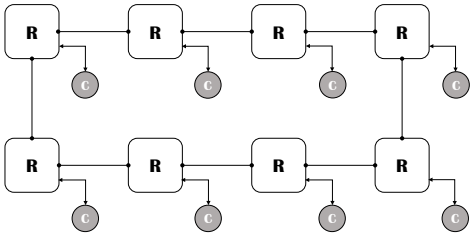
(a) Mesh



(b) Torus



(c) Flattened butterfly



(d) Ring

Figure 2.1: Direct topologies

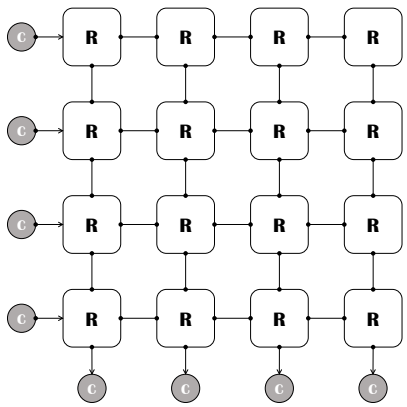
destination nodes reside on opposite sides and packets flow from the left side to the right side [33].

A 2-ary 3-fly butterfly network is depicted in Figure 2.2b with eight source nodes, eight destination nodes and three levels of four switch nodes each.

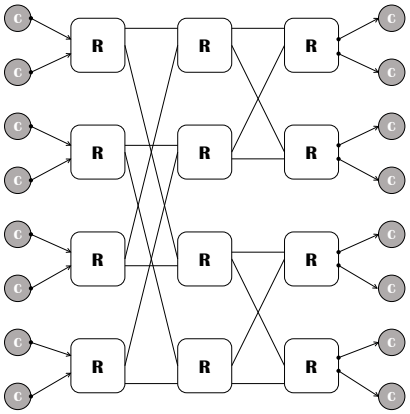
Flattening the routers in each row of a conventional butterfly, gave rise to flattened butterfly topology, wherein router interconnections are maintained still [34, 35]. In flattened butterfly, each destination can be reached with a maximum of two hops. Nevertheless, the wires connecting distant routers are long and buffers are much bigger than those in a mesh topology.

A flattened 4-ary 3-fly network is depicted in Figure 2.1c where all the routers

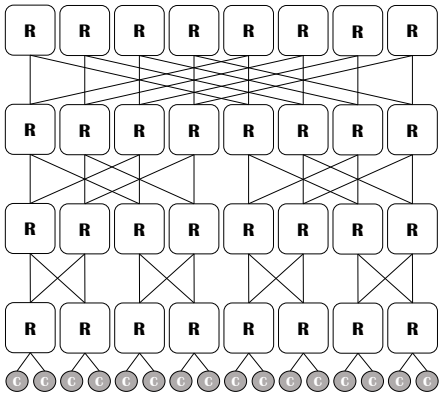
within the same row and column are fully connected. Each router has a radix of ten, four ports to connect to the processing cores and six ports for inter-router connections.



(a) Crossbar



(b) Butterfly



(c) 2-ary 4-tree

Figure 2.2: Indirect topologies

K-ary N-tree

Fat tree is another family of indirect topologies, logically characterized as a binary tree with a single root node. The leaves are terminal nodes (cores) whereas, the inner nodes are switches [36].

Since traffic tends to congest when approaching the root node, the use of higher bandwidth links is required to mitigate this problem. Links grow fatter

the closer we get to the root node, hence the topology name. The large switch sizes and high node degree are the main disadvantages of this topology [37].

Another member of the fat tree family is the k-ary n-tree topology, where n is the number of levels with k^{n-1} switches in each level [38]. A 2-ary 4-tree is depicted in Figure 2.2c with four levels of eight switches each.

Ring is another on-chip topology where each router is connected to exactly two neighbours regardless of the network's size, forming one continuous path for data. It does not scale for a high number of nodes and a single break in the path will interrupt the entire network (Figure 2.1d).

In addition to the star topology, which consists of a central router with large capacity requirements, connected to all the other nodes in the network. It easily becomes a communication bottleneck because of all the traffic passing through it.

2.1.2 3D topologies

Even though 2D based on-chip networks embrace the integration of a large number of intellectual property cores and offer better scalability compared to bus-based architectures, they still are not considered the ultimate solution for communication in future very large scale Systems on Chip [39].

When the size of the network grows larger, the distance between communicating cores increases, which in turn leads to an increase in transmission delays, power consumption and the overall chip size, which has a grave impact on performance.

As an alternative, 3D-based networks were proposed to overcome the limited scalability of their 2D counterparts [40, 41].

3D NoC is a natural evolution of 2D NoC where multiple 2D layers are stacked on top of each other and interconnected with through silicon vias (TSVs) tunnelling through them vertically (Figure 2.3). Consequently, a significant improvement in wire lengths, chip size, energy consumption and overall communication delay is achieved, which in turn implies a major performance improvement [42, 43, 44].

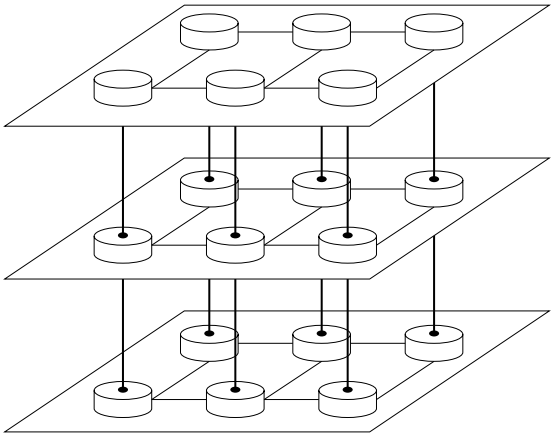


Figure 2.3: 3D mesh

This design progress led to the construction of novel 3D specific architectures such as the symmetric, hybrid and ciliated topologies and opened the door for new challenges [45, 46, 47, 48].

Symmetric mesh

A direct extension of its 2D counterpart, constructed by laying a number of 2D mesh layers on top of one another, interconnected vertically.

Stacked mesh

Another derivative and a modified version of the 3D symmetric mesh. It is a hybrid between a packet-switched network and a bus connecting the layers vertically. Routers are equipped with only six ports, as opposed to those of the symmetric topology which have seven.

Ciliated mesh

In a ciliated mesh, routers are restricted to a single layer and cores are exclusively housed in the other layers. Routers are equipped with at least five ports to connect to the cardinal neighbours and to the intellectual property core. Additional ports are needed to connect to the other cores located at different layers [49].

2.2 Units of communication

Whereas messages are the logical units of communication on-chip, packets are the physical units that transit the network.

Prior to being injected, a message is split into several packets, which are made of a set of fixed-length flits (Figure 2.4). A head flit carries the packet's routing information and allocates network resources such as buffers and channel bandwidth, followed by a set of body flits and a tail flit indicating the end of a packet to release the previously allocated resources. Since they hold no routing information, body and tail flits must remain in order and follow the head flit along its route (they may include a virtual channel identifier).

2.3 Flow control

Flow control supervises the allocation and deallocation of network resources and resolves contention. It dictates when buffers and links are assigned to messages, the granularity at which they are allocated and how resources can be effectively shared. A good flow control mechanism enables the delivery of messages with low latency and, allows the network to achieve a high proportion of its ideal bandwidth [50, 51].

The most basic flow control mechanisms are bufferless, wherein blocked packets are either misrouted or discarded rather than being temporarily buffered

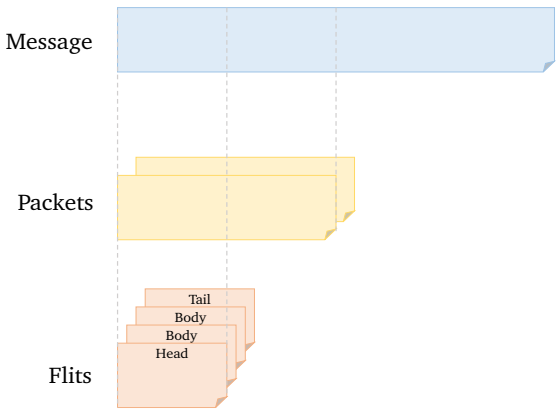


Figure 2.4: Units of communication

[52]. Circuit switching on the other hand, is more complex and efficient than bufferless flow control, wherein only packet headers are buffered. The header transits the network and reserves the entire path (circuit) before any payload packets can follow. Both mechanisms sacrifice costly channel bandwidth in favour of less costly storage space, hence, they are deemed inefficient.

Efficient flow control is achieved by buffering data while waiting for network resources. It is done either in units of packets, which happens to be the case of store-and-forward and virtual-cut-through, or at the finer granularity of flits, as in wormhole and virtual-channel flow control [53].

2.3.1 Circuit switching

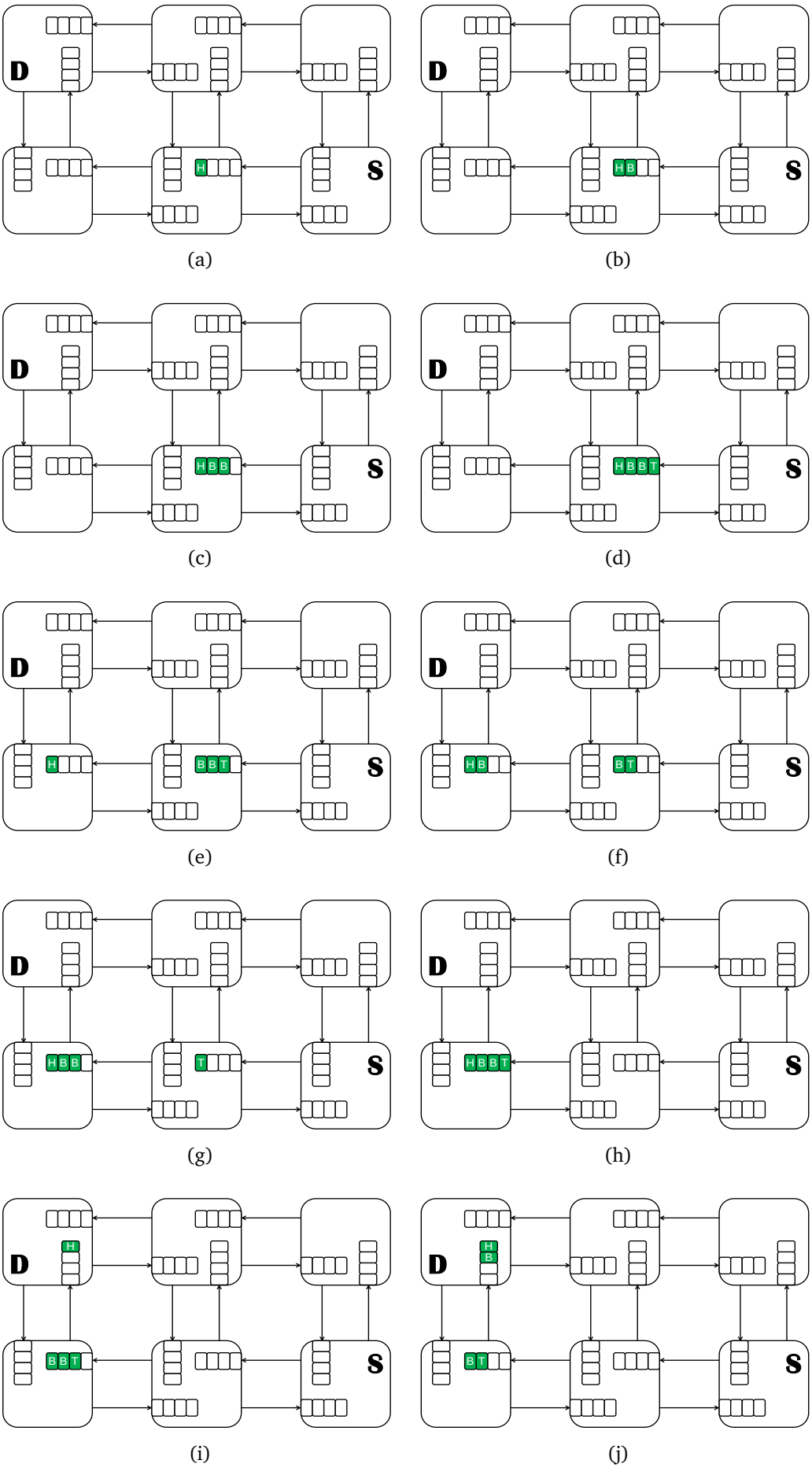
Circuit switching is regarded as a form of bufferless flow control, apart from the header flit, which has to wait in case it fails to allocate a resource straight away.

Initially, the header flit transits the network from the source to the destination node, allocating the channels it traverses along its way, which are needed afterwards to transmit the entire message. Upon arriving at the destination node, an acknowledgement is transmitted back to the source node signalling a successful establishment of the circuit. The source node will then inject the message which will travel quickly through the network without further control. Once the message reaches its predetermined destination, the tail flit will deallocate the circuit and release the resources.

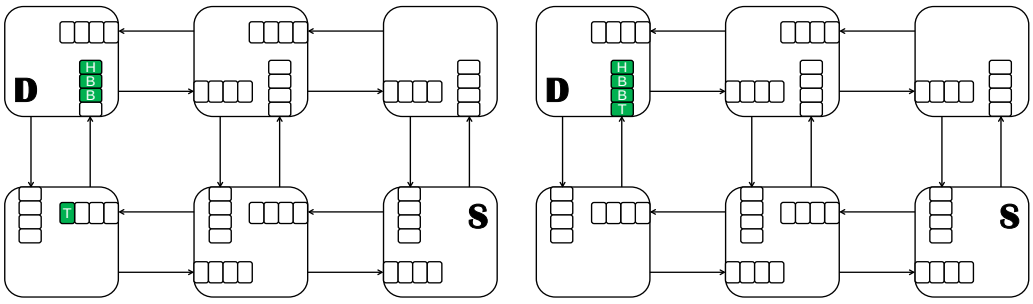
Circuit switching is befitting for large messages and steady traffic and suffers from poor bandwidth utilization, due to the fact that links remain idle between the initial setup and data transfer (other messages seeking to allocate resources are blocked).

2.3.2 Store and Forward

With store-and-forward, every node halts until a packet has been stored (buffered) in its entirety, before forwarding it downstream. The packet must be granted a buffer in the downstream node, as well as exclusive access to the



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)



(k) (l)

Figure 2.4: Store and forward flow control

channel in order to advance. As a result, only a single buffer is occupied by the packet and channels will not be left idle.

With this technique however, buffers are required to be as large as the largest packet in the network and because whole packets have to be stored before initiating transmission, the average latency increases, which is not suitable for delay-critical on-chip networks.

A demonstration of store-and-forward flow control is illustrated in Figure 2.4 with S and D being the source and destination nodes respectively (each sub-figure represents a single cycle). The packet portrayed in green is made of a total of four flits, with H, B and T standing for Head, Body and Tail flit, respectively.

The packet is injected flit by flit and housed in its entirety in the next node along the route (Figure 2.4 (a to d)). Even though the downstream buffer is completely empty, the packet will not make any progress until it is received as a whole, then, and only then, will it proceed to the next node (Figure 2.4 (e to h)). The packet will finally reach its destination (Figure 2.4 (i to l)).

2.3.3 Virtual Cut Through

With the virtual cut-through technique and unlike store-and-forward, packets do not have to be stored as a whole before initiating transmission, which takes place as soon as the head flit is received and the required resources are allocated. This way, delays are drastically reduced.

Nevertheless, since resources are allocated in units of packets, buffers are still required to be as large as the largest packet in the network, which is still not

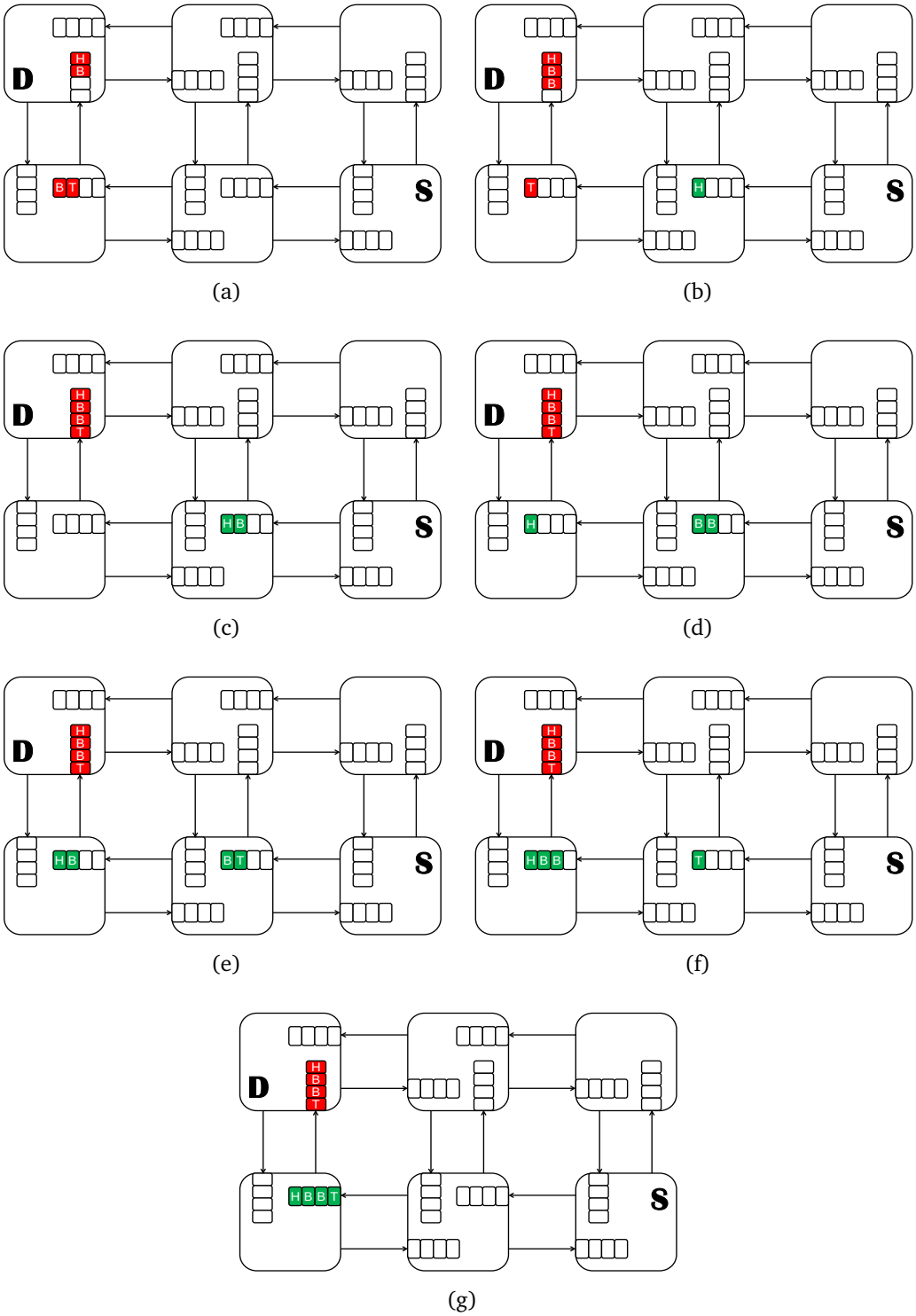


Figure 2.5: Virtual cut-through flow control

suitable for on-chip networks with tight area and power constraints. Particularly when multiple independent buffers are needed to mitigate blocking or provide deadlock avoidance. Which is where allocating buffers in units of flits excels and enables more effective use of storage.

A demonstration of virtual cut-through is illustrated in Figure 2.5 with source node S and destination node D. Another packet (depicted in red) is already transiting through node D (Figure 2.5a) when S starts injecting its packet (depicted in green) in Figure 2.5b. For some reason (congestion or otherwise) the red packet is blocked at the southern buffer of node D (only a portion of the network is shown).

As soon as enough space is available in the downstream node to house the entire green packet (Figure 2.5c), transmission starts without having to await the subsequent flits (Figure 2.5d).

Transmission continues until the green packet is completely housed in the buffer (Figure 2.5 (e-f)). However, because the red packet is already blocked at node D, the green packet will not be able to make any further progress (Figure 2.5g), which justifies the need for buffers to be as large as the largest packet in the network using this technique (the blocked packet occupies a single buffer).

2.3.4 Wormhole

Because large packets call for large buffering requirements at every router, allocating buffer space and channel bandwidth on a flit-by-flit basis is favoured.

Wormhole flow control functions similarly to virtual cut-through, except it allocates storage and bandwidth to flits rather than entire packets, which enables the use of smaller buffers. Every flit can advance to the next hop, as soon as a single flit buffer is available at the downstream node and even before the reception of the entire packet at the current node.

Head flits allocate the output port for the entire packet. This way, the subsequent payload flits use the output port acquired and follow the head flit, nonetheless, they still have to allocate a buffer and channel bandwidth. The tail flit will eventually release the resources.

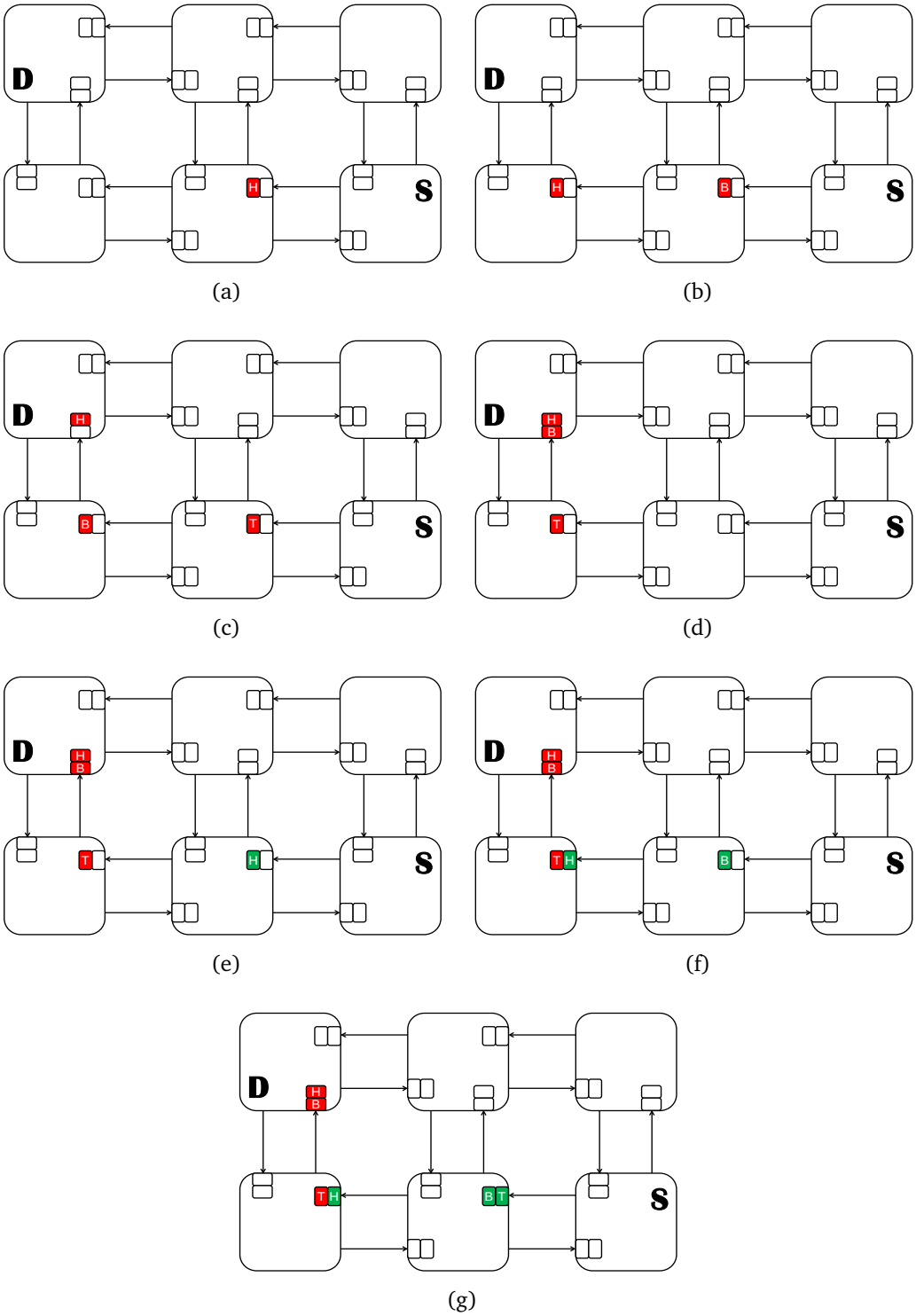


Figure 2.6: Wormhole flow control

Due to tight area and power constraints, wormhole is the predominant technique and ideal flow control candidate adopted for on-chip networks thus far.

Nevertheless, because buffers operate in a first-in-first-out policy and because channels are allocated to packets, when a packet is blocked, all the links it holds are left idle (packets can span multiple nodes). Other packets behind the blocked one will be unable to use the idle links, even if they require a different output port. This is referred to as *head of line blocking*.

A demonstration of wormhole flow control is illustrated in Figure 2.6 with a packet size of three flits. The buffers are significantly smaller (two slots) because allocation is done on the smaller granularity of flits. Two packets generated at node S transit the shown portion of the network. While the green packet is destined for node D, the red packet is only transiting through it.

Flits will move forward as soon as a single slot is available at the downstream node (Figure 2.6 (a-d)). When the red packet comes to a halt at node D, due to a blockage in its route, the green packet is injected in the network (Figure 2.6e). The latter will also inevitably come to a halt behind the red packet because the channel is occupied (*head of line blocking*, depicted in Figure 2.6g). The blocked packets in this case span three nodes (more if packets sizes were larger).

2.3.5 Virtual channels

Virtual channels overcome the blocking problem of wormhole flow control by allowing several packets to share the available channel bandwidth, that would otherwise be left idle when a packet blocks. To successfully achieve this, several virtual channels, each with its own separate flit-buffer, are associated with every physical channel [54, 55].

A demonstration of virtual channel flow control is illustrated in Figure 2.7 with two virtual channels per input port.

Assuming the same scenario as before and, even though the red packet is blocked, the green packet headed towards node D will still be able to reach its destination, due to the free path provided by the second virtual channel.

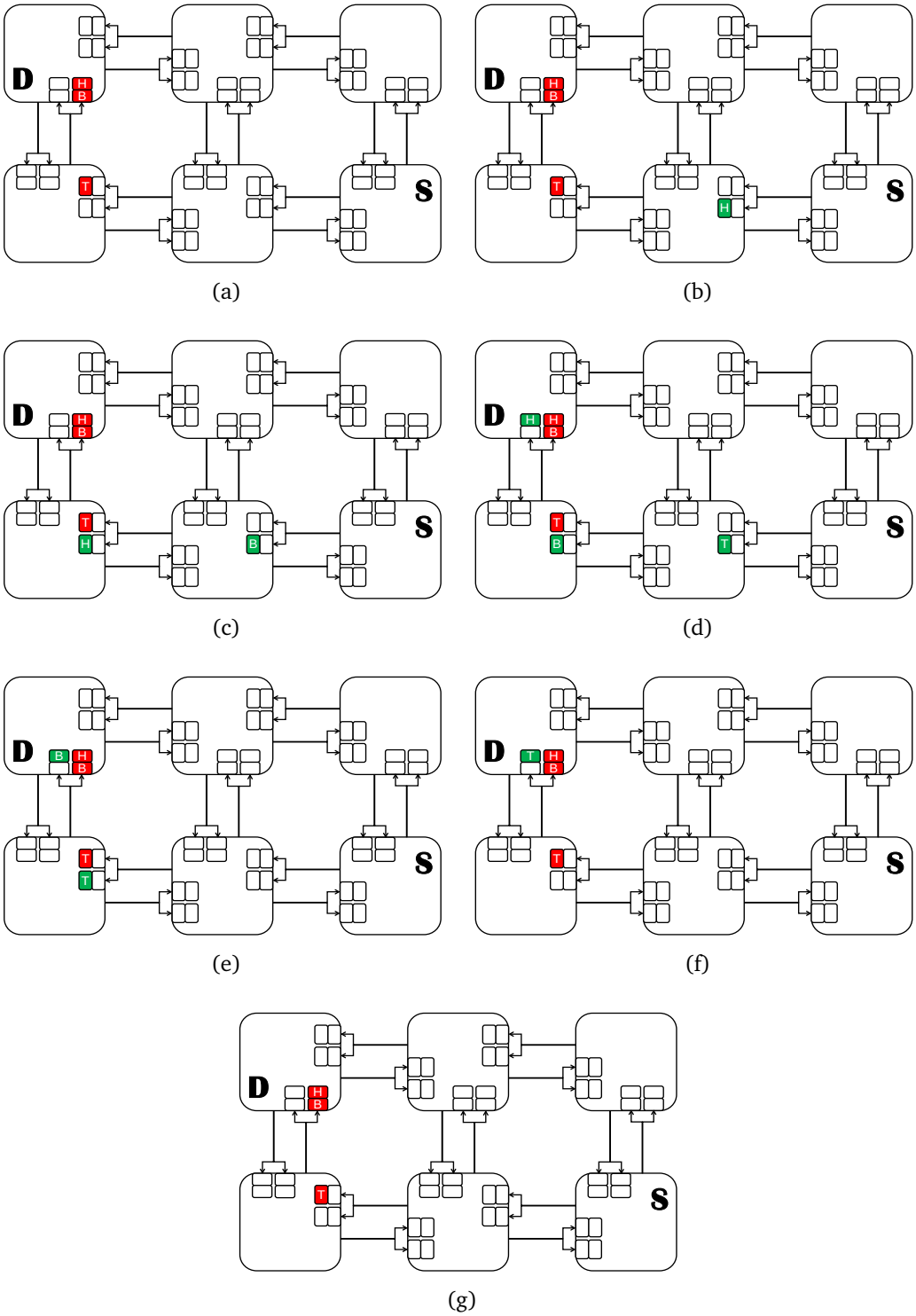


Figure 2.7: Virtual channels flow control

2.3.6 Summary

A diagram demonstrating the relationship and classification of flow control techniques is depicted in Figure 2.8.

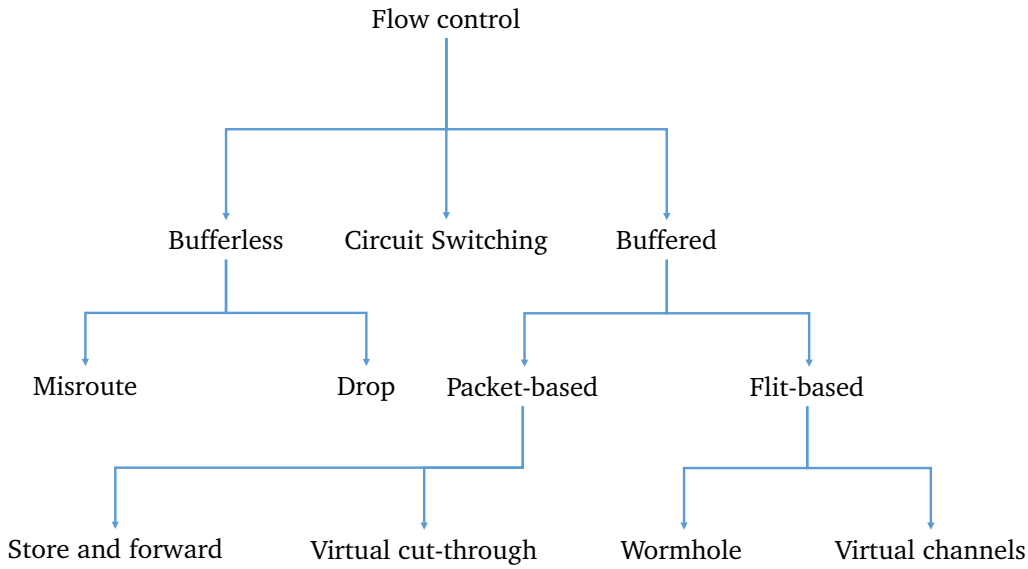


Figure 2.8: Flow control techniques

2.4 Buffer management

Buffered flow control requires a means to communicate buffer availability between adjacent routers. An upstream router is entitled to find out if enough storage space is available at the downstream router, to accommodate flits or packets. The latter depends on the specific flow control protocol, since store-and-forward and virtual cut-through manage buffers in units of packets, while wormhole and virtual channel flow control manage buffers in units of flits. Circuit switching on the other hand, being bufferless, does not require buffer management.
The most commonly used buffer management techniques are the following:

2.4.1 Credit-based

Every router keeps track of every downstream neighbour's free buffers' space, in the form of credits. When it transmits a flit downstream, it decrements the

corresponding credit count. When the downstream router has an available free slot, it sends a credit back upstream incrementing the count. If credits are no longer attainable, the sender halts the transmission of data packets.

2.4.2 On/Off

When the downstream router is no longer capable of buffering packets/flits due to lack of storage space, it dispatches an *Off* signal upstream, to halt data transmission. Whenever that router restores the capability of storing more data, it dispatches an *On* signal back to the sender, to carry on with data transmission.

2.4.3 Ack/Nack

When downstream routers receive data, they send an acknowledgement *Ack* back upstream, signalling a successful transmission. Nonetheless, if enough storage space is unavailable, data is dropped and a negative acknowledgement *Nack* is sent instead, signalling the need for retransmission. Thus, routers have to keep copies of their data until they receive an *Ack*.

2.5 Router micro-architecture

Routers, being the pillars of communication, are designed to satisfy throughput and latency demands, which becomes rather challenging when these demands grow [56].

A state of the art input-queued, credit-based, virtual-channel, router micro-architecture with four pipeline stages depicted in Figure 2.9, consists of the following essential units:

- *Input buffers:* where packets are stored upon their arrival and housed throughout their duration in the router.
- *Route computation:* to determine and assign the appropriate output ports and virtual channels (if any) to packets.
- *Virtual channel allocator:* to grant a given packet access to the desired virtual channel among all requesters.
- *Switch allocator:* to grant a given packet access to the desired output port.

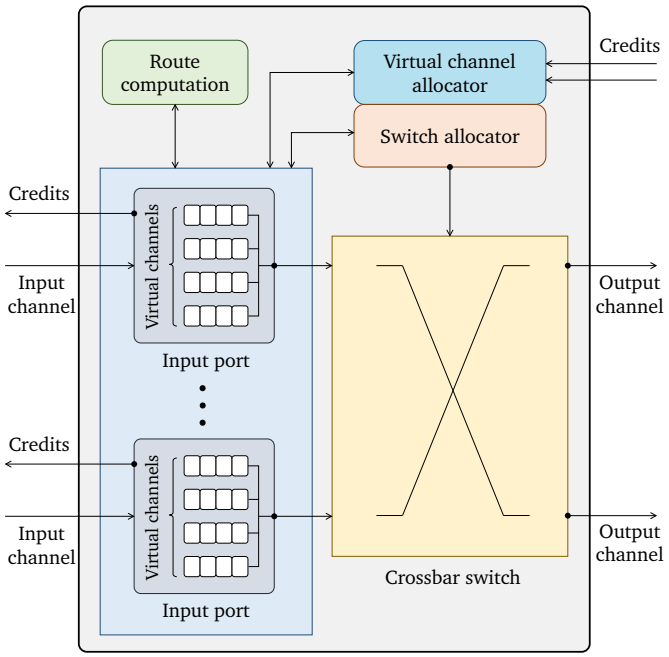


Figure 2.9: Input-queued credit-based virtual-channel router

- *Crossbar switch:* configured to connect any input buffer to any output channel and, responsible for physically moving flits from the input to the output ports.

A flit in one of the current node's input buffers (the upstream node) must be allocated space in the downstream node's input buffer, bandwidth on the next channel of the route and, must also win the arbitration to traverse the crossbar switch in order to be able to progress.

In the buffer writing pipeline stage, the head flit is buffered upon its arrival in the appropriate input virtual channel. The route computation unit will then determine its output port.

In the virtual channel allocation stage, the header flit arbitrates for a virtual channel at the next router's input port. It then advances to the switch allocation stage, where it arbitrates for the switch output port. The flit proceeds to traversing the crossbar switch in the switch traversal stage. The Crossbar switch is non-blocking, i.e. the connection of any permutation of inputs and outputs is supported, which enables high throughput.

Lastly, the flit goes through the link traversal stage to arrive at the down-

stream node. The body and tail flits follow a similar pipeline, but they do not undergo the route computation and virtual channel allocation stages, which are performed only once per packet. The header flit allocates the virtual channel and the tail flit, upon leaving the router, deallocates it.

The virtual channel allocation stage is not required in a wormhole router where input ports are associated with a single buffer queue (the absence of virtual channels eliminates the need for a virtual channel allocator).

Since every flit has to go through all these stages at each hop, this design could introduce undesirable performance degradation.

Other designs rely on initiating two or more of these stages in parallel for better efficiency, such as the one proposed in [57]. In which, the next-port direction of the downstream router is pre-calculated and embedded in the header flit. Upon its arrival at the downstream node, the information will be extracted and used by the switch arbiter to directly grant the output-port.

2.6 Routing

The backbone of any NoC communication fabric is the routing algorithm, which is essentially responsible for deciding which path packets will cross, to ensure their delivery to their proper destination [22, 24].

It comprises two fundamental functions: routing and selection. The former resolves a set of eligible routes for data transmission, whereas the latter decides which one will be selected [58].

Its prime goal is to distribute traffic evenly across the network, in order to avoid hotspots and minimize contention, so as to improve the overall packet latency and throughput [59].

Routing algorithms in Networks-on-Chip play a vital role and are classified as either deterministic, oblivious or adaptive, with regard to how they select between the set of possible paths.

2.6.1 Deterministic routing

Deterministic algorithms invariably appoint the same path for every pair of source and destination nodes. Path diversity is completely ignored and the selection function becomes dispensable. They inadequately balance the load across the network, which results in increased overall latency especially in the presence of congestion.

They are also likely to suffer from throughput deterioration when the packet injection rate is high. However, they have been widely adopted on-chip due to their simplicity in router design (easy to implement and make deadlock free).

Figure 2.10 illustrates a routing example with a deterministic routing scheme, with S and D being the source and destination nodes respectively. Even though the highly congested nodes (denoted in red) can be easily avoided due to the presence of multiple available routes between the pair of communicating nodes, still, due the static nature of the scheme, packets will have to pass through them.

2.6.2 Oblivious routing

With oblivious routing, routes are chosen without any consideration for the network's condition. The selection function makes its decisions based solely on the information provided by the header flit, without exploiting any information regarding links and buffers' state [60, 61].

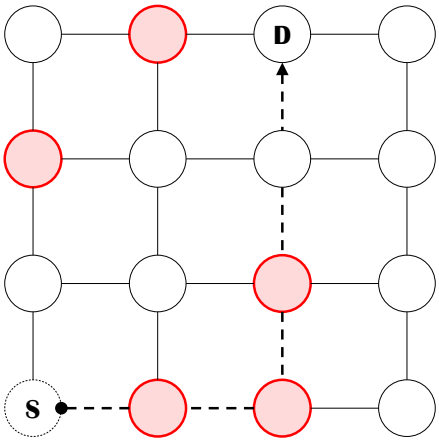


Figure 2.10: Deterministic routing

Random algorithms that uniformly distribute traffic across all the paths are considered oblivious. The main issue with oblivious routing schemes is that they affect an irregular load, which is especially high in the middle areas of the network [22].

2.6.3 Adaptive routing

With adaptive routing, paths are determined based on congestion information and the network's state, and packets are not restricted to a single path along the way to their destination [62].

With a better load distribution across links, adaptive algorithms are able to boost the overall network performance (reduce average latency and enhance throughput) and also provide tolerance if link or router failure occurs [63, 64, 65]. They have an edge on their deterministic and oblivious peers, in that they ensure avoiding the formation of hotspots and congested regions in the network.

Adaptive routing is best suited in the presence of irregular, non-uniform, bursty traffic due to their ability to adapt to the ever-changing congestion conditions in the network, which is a key objective in the design of reliable routing algorithms for networks-on-chip [24].

Figure 2.11 illustrates a routing example with an adaptive routing scheme. The latter helps manoeuvre packets around the highly congested nodes, and allows them to reach their destination following the less congested route.

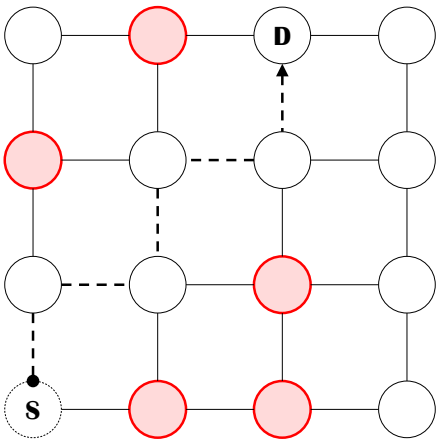


Figure 2.11: Adaptive routing

Categories

Adaptive algorithms are referred to as minimal, if they only permit the selection of the shortest paths between any given pair of source and destination nodes, or non-minimal otherwise. The latter has the potential to improve load balance beyond the limits of minimal routing but at the cost of greater implementation complexity and potentially higher per-packet latency.

Adaptive schemes can also be either fully adaptive, if they do not inflict any restrictions on path selection, or partially adaptive if they do [66, 53].

They can further be classified as either progressive or backtracking. With progressive routing, the packet's header only moves forward, whereas with backtracking, the header is allowed to backtrack, releasing previously reserved channels, which is mostly used when fault tolerance is desired [15].

2.7 Routing challenges

The implementation of a reliable routing scheme can be a rather challenging task, since there are many conflicting objectives to fulfil, wherein the improvement of one might deteriorate the others.

2.7.1 Congestion awareness

Congestion is one of the most common issues that contribute to performance deterioration and affect the reliability of on-chip communication. Therefore, congestion awareness is a top priority in the design of adaptive routing schemes [67, 68].

The latter, can be locally aware of congestion by taking into consideration the information of directly connected neighbours (adjacent nodes), such as the number of their free buffer slots or virtual channels. Alternatively, by having a broad vision of the network, they can be globally aware of congestion and hence be able to make more effective decisions as opposed to locally aware schemes.

Making use of the vacated space on chip, global congestion information can be propagated using a dedicated congestion propagation network, separate from the data network [69, 70].

2.7.2 Fault tolerance

Besides congestion awareness, one of the common issues that routing algorithms are capable of tackling is fault tolerance [71, 72].

Faults are likely to occur due to several unpredictable reasons and are rather challenging to detect and mend. They could be either transient, when they remain in the system for a limited amount of time, or permanent, caused by physical damages such as manufacturing defects and device wear-out [73, 74].

Routing schemes attempt to tolerate faulty nodes or links by forwarding packets through alternative paths and around the fault. However, by doing so, it might result in the formation of congested regions, inevitably affecting the overall performance, hence a design trade-off is required.

2.7.3 Deadlock and livelock

Even though adaptiveness reduces the chance of traversing hot-spots, helps avoid faulty components and diminishes the blocking probability of packets, it is still not quite enough. Another important aspect of a routing algorithm is guaranteeing deadlock and live-lock freedom.

Deadlock

Deadlock is the situation where a few packets reserve some of the network resources and rely on other packets to release the rest of the resources, which never takes place and all packets are blocked indefinitely and will never be able to reach their destination.

A deadlock situation is depicted in Figure 2.12, with four nodes labelled N1, N2, N3 and N4.

All four nodes inject packets at the same time according to the following communication pattern: N1 sends a packet to N3 and vice versa and N2 sends a packet to N4 and vice versa.

- N1 → N3 through N2
- N2 → N4 through N3
- N3 → N1 through N4

- N4 → N2 through N1

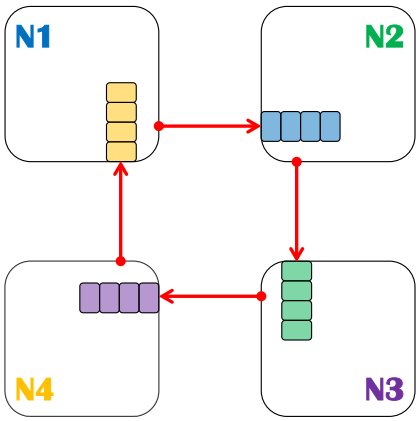


Figure 2.12: Deadlock

A cyclic dependency is formed where each packet is waiting for the downstream node's buffer to be released, which leads to a full blockage and all four packets will not be able to make any further progress.

Therefore, deadlock avoidance is a critical feature in routing schemes' design. It could be achieved by either using virtual channels or by prohibiting the formation of certain turns altogether, to break the cyclic dependencies [75]. The former can be expensive in terms of area overhead and power consumption due to the additional control modules, and the latter will reduce the scheme's resilience significantly and limit the choices. Both methods ensure that deadlock never takes place to begin with.

The application of recovery techniques is another possible approach to prevent deadlocks after their occurrence. A deadlock detection mechanism is required in this case and inevitably, one or more packets involved must be dropped to break the cyclic dependency.

Other recovery techniques like the one proposed in [76] and instead of dropping packets and having to somehow send a notification back to their source nodes, a deadlock free lane is constructed and used as an escape route by one of the packets involved in the deadlock situation to be routed minimally towards its destination.

In 3D topologies, deadlock avoidance is somewhat more complex as cycles

can either be formed within the same plane or include vertical channels.

Livelock

When non-minimal routing is applied, livelock becomes rather an issue. It is the situation where a packet roams around the network but never reaches its destination due to misrouting. It could be mitigated if a maximum number of misroutes is allowed per packet or by assigning higher priorities to packets that have been misrouted the most.

2.8 Conclusion

In this chapter we discussed the main building blocks of on-chip networks. We examined the different topologies that govern the physical layout of the network, in addition to the communication units, flow control and buffer management techniques, as well as the router micro-architecture and its pipeline stages. We concluded by glancing at routing algorithms, their classes and the major challenges they run into.

In the next chapter we will dive deeper into the world of on chip routing protocols.

Chapter 3

On-Chip Routing Protocols

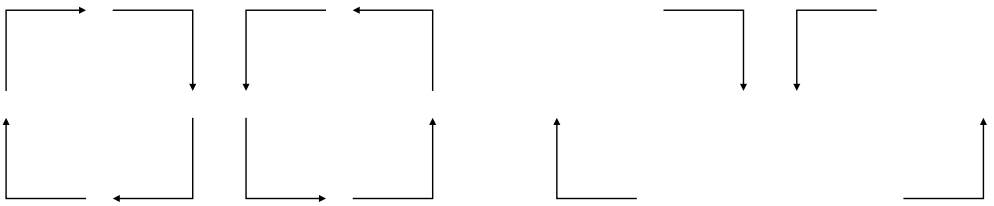
Network on Chip is expected to meet numerous ever-increasing performance needs such as low latency, high throughput and low power consumption, which is commonly attained by means of routing algorithms. The latter are used to forward data packets along the most appropriate routes between any pair of communicating nodes. They are chiefly expected to provide congestion awareness, deadlock and livelock avoidance, path diversity and balanced load distribution across the whole network [22, 24].

In this chapter we unveil a list of the most relevant on-chip routing schemes with their different classes and characteristics, as well as the set of objectives they aim to fulfil. We conclude with the metrics used to evaluate the performance and test the efficiency and reliability of a given routing algorithm and have a good understanding of its pros and cons.

3.1 Deterministic routing

With deterministic routing, packets always follow the same path from a source to a destination node. The deterministic dimension ordered routing algorithm (DOR) is considered the most popular in mesh-based network-on-chip, due to its simplicity in router design.

To reach their ultimate destination, packets are first routed along the X dimension and then along the Y dimension (or vice versa in YX routing). When a packet is forwarded along the X dimension (moving east or west), it is allowed



(a) All possible turns (b) XY allowed turns

Figure 3.1: Allowed turns with dimension-ordered routing

to switch to the Y dimension (make a north or south turn), but once it is moving along the Y dimension, all other turns are disallowed [77, 78].

Dimension ordered routing's allowed and disallowed turns are depicted in Figure 3.1.

Without path diversity, DOR is unable to route around faults or avoid congested regions in the network, exactly one path exists between every source and destination pair.

Such scheme is simple to implement but unable to balance the load across the links in the presence of bursty traffics, which causes significant performance deterioration, hence, it is best suited when a uniformly distributed traffic is applied. In most real world applications, a node communicates with a set of specific nodes in the network, more frequently than others.

Because it only allows certain turns, it eliminates the formation of a cyclic resource dependency and ensures deadlock freedom. It is also livelock-free by nature, due to the fact that it is minimal.

The conventional XY-based router's pipeline design consists of: buffer writing, route computation, switch allocation and crossbar traversal stages. Since every flit has to go through all these stages at each hop, this design might introduce undesirable performance degradation.

Other designs rely on initiating two or more of these stages in parallel for better efficiency, such as the one proposed in [57]. In which, the next-port direction of the downstream router is pre-calculated and embedded in the header flit. Upon its arrival at the downstream node, the information will be extracted and used by the switch arbiter to directly grant the output-port.

Because of this parallel process, the pipeline design is optimized. Furthermore, It is associated with the no-load bypass technique, where the buffer writing stage is overlapped. In case the input buffer is empty, flits don't have to be stored and continue their path straight to the routing computation and switch arbitration, where both stages are still executed in parallel.

3.2 Oblivious routing

3.2.1 Flooding

With flooding, copies of an incoming packet are transmitted on all outgoing links, irrespective of the ultimate destination. The copies spread through the network like a flood, hence the name. Eventually, one of the packets reaches the destination and the other copies are discarded.

The scheme can take advantage of the on-chip network's regularity to direct the flood towards the vicinity of the destination. Nonetheless, it requires significant network resources and packets will reach undesired areas in the network [79].

3.2.2 Deflection

Common virtual channel routers have buffers associated with their input ports where flits are settled in, until an output port is available. As buffers are power hungry and buffer management circuits are complex, bufferless routing (deflection) gained popularity and have been widely used to lower hardware overhead and power consumption in NoCs [80, 81].

With bufferless Hot-Potato routing, incoming packets are always forwarded through a free output channel without stalling or storing them. When two packets arrive at the same time and desire the same output port, the router resolves the contention by dropping one of the two packets or deflecting it to an arbitrary output port [82].

Prior to injecting a message into the network, the source node must have a free outgoing channel available at that time, which contributes to the total time

it takes to deliver the packet to its destination.

The worst case scenario involves misrouting packets to a distant region in the network far away from their destination, where they interfere with other packets causing them to be deflected as well, which triggers a "domino effect" wherein each packet deflects another packet. Additionally, livelock becomes a major concern because packets are likely to be continuously deflected, thus constructing livelock-free mechanisms is a requirement for deflection routing.

3.2.3 Valiant

Valiant is a non-minimal randomized routing scheme that comprises two stages to ensure that packets generated at the same source, and headed towards the same destination, do not transit the same path [83].

In the initial stage, packets are forwarded from the source to an intermediate node in the network selected arbitrarily, before being delivered to their proper destination in the final stage. On that account, it effectively balances the load across the entire network regardless of its topology. The scheme relies on virtual channels to prevent deadlock and since packets will eventually reach their destination, livelock is also mitigated notwithstanding the scheme's non-minimal nature.

Although any arbitrary routing scheme can be used in the two stages, DOR is a perfect candidate for mesh-based networks as it is best suited for balancing the load under uniform traffic. Load balancing however, comes at the expense of considerable hop count due to the longer distance that packets have to cover, which in turn increases average packet latency.

Valiant routing is illustrated in Figure 3.2 with S, i and D being the source, intermediate and destination nodes, respectively.

3.2.4 Randomized oblivious multi-phase minimal routing

The randomized oblivious multi-phase minimal routing scheme (ROMM), inherits the randomization of valiant and the uniformity and minimality of DOR. The former ensures that packets travelling back and forth between the same pair

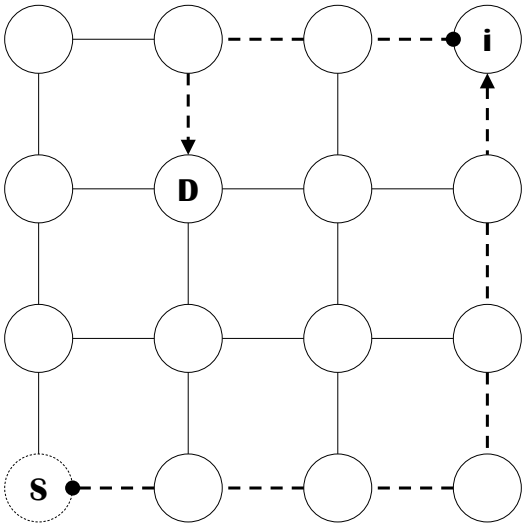


Figure 3.2: Valiant randomized routing

of source and destination nodes, seldom follow the same route. The latter on the other hand, ensures that only minimal paths are selected to limit randomization compared to valiant, which frequently results in non-minimal paths and leads to higher demand for network bandwidth and longer routing time [84].

ROMM comprises p phases, with $p-1$ nodes chosen randomly as intermediate hops along the minimal path between the source and destination pair of communicating nodes (Figure 3.3).

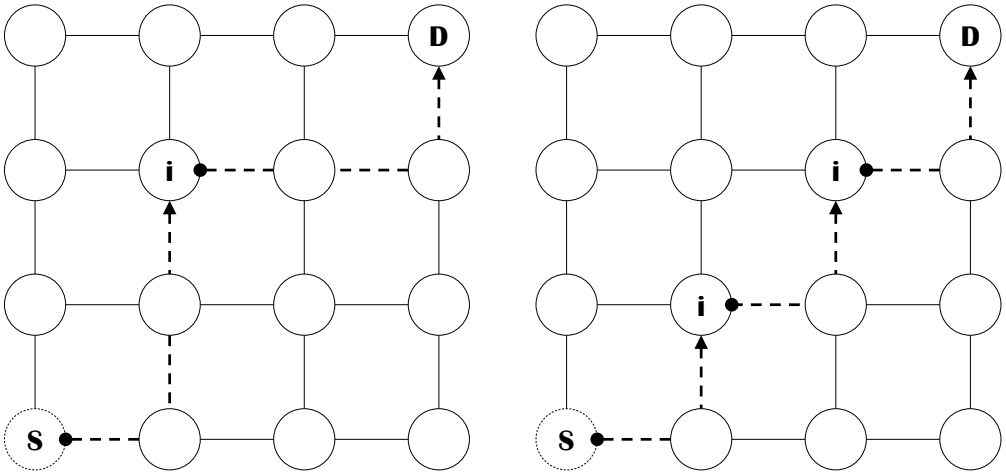
Due to its minimal nature, livelock can not occur and deadlock is mitigated through the use of virtual channels.

3.2.5 Orthogonal one-turn routing

Because ROMM is afflicted by poor worst case throughput and valiant by excessive hops and poor latency, orthogonal one-turn routing (O1TURN) was proposed to overcome these problems [85].

In order to guarantee their arrival with the minimum number of hops, O1TURN enables packets to traverse one of the two dimension-ordered routes XY (X-first Y-next) or YX (Y-first X-next) randomly and with equal probability (packets are allowed to make only one turn along the entire route, hence the name).

It resembles rather a restricted version of ROMM with only one phase and



(a) 1-phase Romm (b) 2-phase Romm

Figure 3.3: Randomized oblivious multi-phase minimal routing

one intermediate node (one of the minimal path boundary nodes).

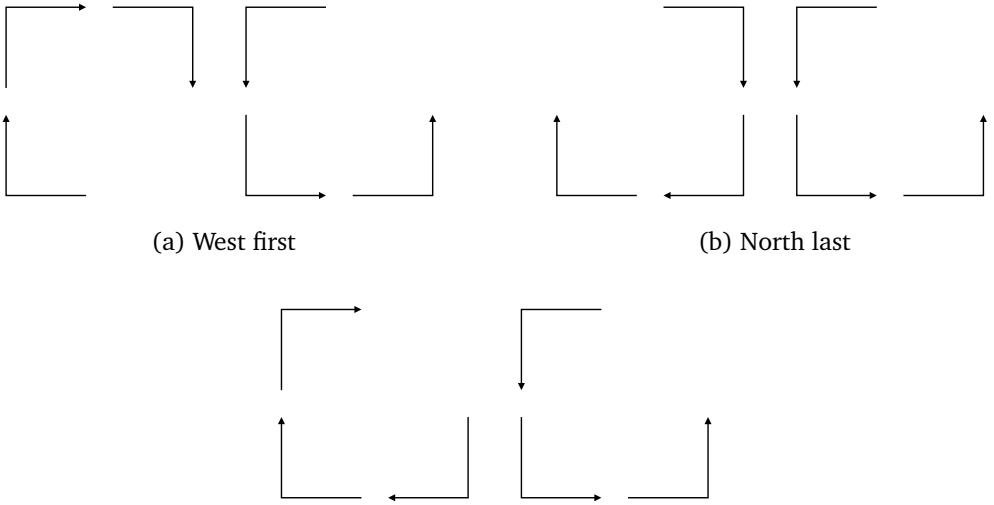
Since one virtual channel is sufficient to achieve deadlock freedom with DOR, O1TURN relies on two virtual channels per physical channel to divide the network into two virtual networks, one used by packets moving along the XY route and the other by packets moving along the YX route.

3.3 Turn models

The turn models for adaptive routing were proposed based on the vital idea that a routing scheme would prevent deadlock altogether when certain turns in the network are prohibited. One turn in each of the many possible cycles is disallowed, but not more turns than necessary, otherwise, the algorithm's flexibility will diminish [86].

Three variants have been proposed: west-first, north-last and negative-first (Figure 3.4). Each prohibiting two out of the eight possible turns to break any cyclic dependencies and ensure deadlock freedom:

- *West-first:* packets are routed first to the far west and then adaptively south, east and north because turns to the west are prohibited.
- *North-last:* turns to the north should be taken last. Packets can be routed



(a) West first (b) North last

(c) Negative first

Figure 3.4: Turn models

adaptively west, south and east, but once they turn north, all other turns are disallowed.

- *Negative-first:* any turn from a positive direction (north and east) to a negative direction (west and south) is prohibited. In order for a packet to travel in the positive direction, it must travel in the negative direction first.

The 4N-First and 4P-First are extensions of the turn models in 3D-based mesh networks:

- Two of the possible three positive directions, north, east and up are selected as the last two turns taken by packets to reach their destination (north and east in the case of 4N-First) making the total of forbidden turns, six (east-up, east-down, east-south, north-up, north-down and north-west).
- Two of the possible three negative directions, south, west and down are selected as the last two turns taken by packets (south and west in the case of 4P-first), thus six possible turns are prohibited (west-up, west-down, west-north, south-up, south-down and south-east).

4NP-First is a combination of both 4N-First and 4P-First and aims predominantly at fault tolerance.

Every packet is forwarded using the 4N-First model and its redundant copy using the 4P-First model, when the fault rate in the network exceeds a certain threshold [87].

To ensure deadlock freedom, two virtual channels are used, one devoted to original packets and the other to their copies. In case the fault rate is below the threshold, only the original packet is sent using the 4N-First model and the other virtual channel is shut down to reduce energy consumption.

With 4NP-First, minimal paths are the principal choice and always given the highest priority. Nevertheless, if they are not a feasible option, other paths along the middle areas of the network that offer more diversity afterwards, are favoured over those closer to the boundary.

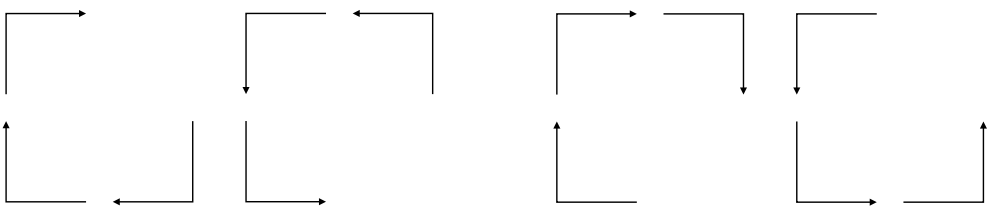
3D network on chip and fault tolerant routing is discussed in more detail in the appendix.

3.4 Odd-Even turn model

The odd-even turn model was also proposed to avoid deadlock in mesh-based on-chip networks, where two rules are applied to restrict the number of turns in odd and even columns (Figure 3.5) [88].

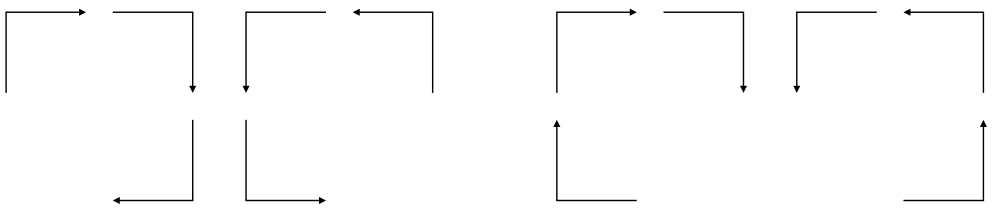
- *Rule 1:* north-west and south-west turns are disallowed in odd columns.
- *Rule 2:* east-north and east-south turns are disallowed in even columns.

A modified version was also introduced where turn restrictions are applied row-wise other than column-wise (Figure 3.6).



(a) Odd column (b) Even column

Figure 3.5: Odd-even turn model



(a) Odd row (b) Even row

Figure 3.6: Modified odd-even turn model

- *Rule 1:* in odd rows, west-north and east-north turns are disallowed.
- *Rule 2:* in even rows, south-west and south-east turns are disallowed.

It was then extended to the third dimension (which will be discussed in detail in the appendix), with an additional rule to eliminate deadlocks that might involve vertical links [89].

- *Rule 3:* XY-Down and XY-Up turns are disallowed in odd planes (packets travelling within an odd plane cannot leave through the downward or upward direction). Similarly, Up-XY and Down-XY turns are disallowed in even planes (packets travelling upwards or downwards cannot enter an even plane).

Even though turn models help prevent deadlock, they impose a lot of restrictions on the possible turns that packets can make when transiting the network, which significantly reduces their resilience and might eventually steer them towards unwanted congested regions.

3.5 Dynamic Adaptive Deterministic routing

Dynamic Adaptive Deterministic routing (DyAD) combines the advantages of both adaptive and deterministic routing. The former is able to achieve a much higher saturation throughput compared to the latter, whereas the latter excels at low network workloads in terms of average latency. On that account, DyAD switches back and forth between deterministically routing packets when there is relatively no congestion in the network, and adaptively routing them when congestion exceeds a pre-set threshold [65].

Accordingly, every router monitors its input buffers and asserts a congestion flag with the value 1, if the occupation ratio reaches 60% of the total buffer capacity, indicating a congested node. Every node will monitor its neighbouring nodes' flag values and activates the adaptive mode if at least one is asserted, otherwise it switches back to deterministic mode.

Although technically any adaptive and deterministic schemes combination can be suitable to form DyAD, for deadlock avoidance purposes, odd-even for the adaptive mode and odd-even-fixed for the deterministic mode are used. The latter is a deterministic version of the odd-even model where instead of multiple eligible output channels, the routing function returns only a single one.

3.6 Locally congestion aware adaptive routing

3.6.1 Dynamic XY routing

Dynamic XY (DyXY) is a minimal adaptive routing scheme, where packets are routed along either the X or the Y dimension, based on the state of directly connected neighbours (Figure 3.7) [90].

Because only local congestion information is included in the decision-making process (the free buffer slots of adjacent neighbours), non-optimal decisions are likely to occur (steering packets towards a heavily congested region in the network), which ultimately impacts the overall performance.

Because packets are allowed to be routed along both the X and Y dimensions without imposing any restrictions, a mechanism is required to guarantee deadlock avoidance [91]. Accordingly, two virtual channels along the Y dimension are needed to partition the network into two separate sub-networks and routing takes place as follows:

- If the destination node is located eastward, packets will be forwarded along the first sub-network;
- The second sub-network will otherwise be used if the destination node is located westward;

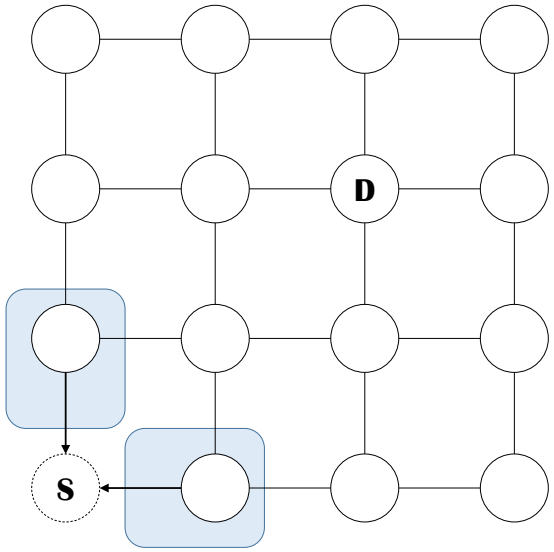


Figure 3.7: Dynamic XY

- When the communicating pair reside at the same row or column, either sub-network is used.

3.6.2 Enhanced Dynamic XY

Enhanced Dynamic XY (EDXY) is an upgraded version that aims at avoiding a particular problem that arises with DyXY routing [63].

When the destination node is one row (column) away, and because the current node is oblivious of congestion related to all other nodes, other than the two adjacent neighbours, designating the one that resides at the destination row (column) as the next hop, may prove to be a poor decision if the other nodes leading to the destination happen to be highly congested. Packets in this case will be deterministically routed along the X or Y direction, since no more options are available.

In Figure 3.8, the source node S is one row away from the destination node D. In this case, forwarding packets to node 4 forces packets to transit the two highly congested nodes 5 and 6.

The problem is mitigated by using a congestion flag for each row and column in the network, which is activated if the number of occupied slots of the buffer exceeds a predetermined threshold.

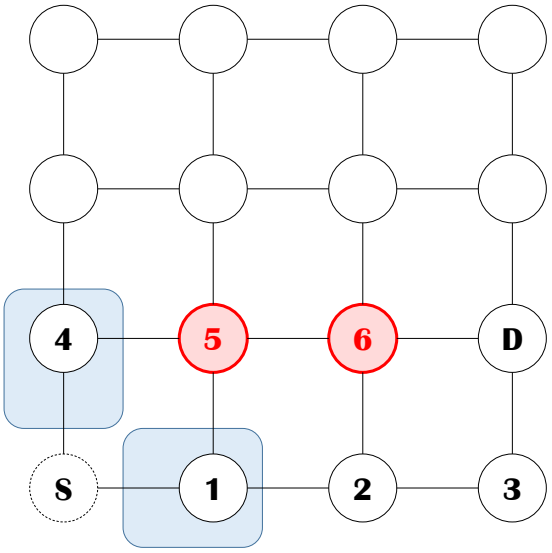


Figure 3.8: Enhanced dynamic XY

The flag value (1 or 0) is obtained by *ORing* the current node's flag and that of the downstream node. Nodes will then inform their left-hand side neighbours of congestion in the right-hand side and vice versa. An activated flag indicates that at least one node along the destination row (column) is highly congested with no regard to their location and should be avoided.

Ordinarily, EDXY is identical to DyXY, however when the destination node is one row or column away, besides the free buffer slots of adjacent neighbours, the congestion flags are also taken into account when resolving the next hop.

3.6.3 Dynamic XYZ

Dynamic XYZ (DyXYZ) is a direct extension to the third dimension, where at each intermediate node, packets can be delivered through one of the three possible directions X, Y or Z as the less congested route is selected [92].

To guarantee deadlock freedom, the network is divided into eight distinct sub-networks, each uses a different set of virtual channels (four virtual channels are required along each dimension).

To further optimize the design, the number of virtual channels for one of the three dimensions is lowered from four to only two. Accordingly, the whole network is split along that dimension into two main sub-networks, each made

of four auxiliary sub-networks.

The auxiliary sub-networks within each main sub-network use different sets of virtual channels on the two remaining dimensions. Therefore, the eight sub-networks are disjoint and the whole network is deemed deadlock-free.

3.6.4 Neighbours on Path

Because locally congestion-aware routing algorithms that rely solely on directly connected neighbours, will most likely make poor decisions due to their myopic view of the network, eventually steering packets through heavily congested areas. The state of neighbours a few hops away should also be taken into account for better efficiency [93]. Which happens to be the case of the Neighbours on Path (NoP) scheme, where congestion-related information of two-hop neighbours (a NoP) is harvested and decisions are based on their free buffer slots (Figure 3.9).

A scoring mechanism is implemented, where the channel with the highest score is selected. The latter is defined as the total sum of every available NoP's free buffer slots. If two or more channels have the same score, the channel will then be selected randomly.

However, the non-optimal decision-making problem remains intact, and routing packets through congested regions is sometimes inevitable.

3.7 Globally congestion aware adaptive routing

3.7.1 Agent-based routing

With Agent-based routing, a lightweight agent network made of a group of clusters is built upon the data network. Clusters incorporate a group of routers and a cluster agent, responsible for spreading congestion information (Figure 3.10). Accordingly, routing decisions are based on a combination of local information and the congestion levels of the neighbouring clusters [59, 94].

A congestion status is associated with each input buffer, which is asserted to the value 1, when the space occupied exceeds a predetermined threshold (it

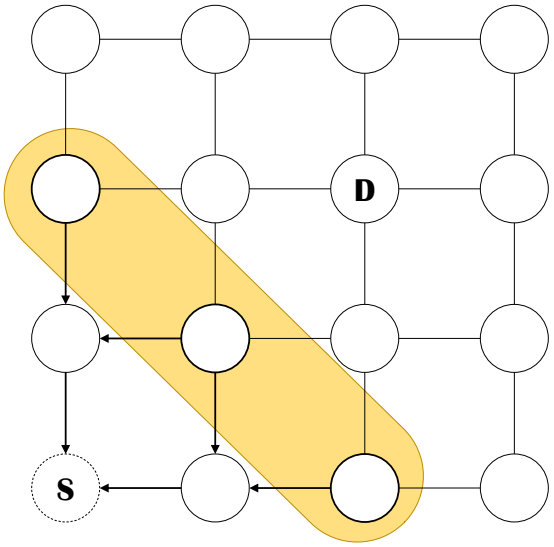


Figure 3.9: Neighbours on path

defaults to 0). The sum of the buffers' congestion statuses, yields the router's congestion level.

Each cluster agent will then collect the congestion levels of all the attached routers and distribute the information to the neighbouring agents. The latter will forward the information to their attached routers.

The less congested route along the minimal path will then be selected as follows:

- *When the source and destination cluster agents are located at the same row or column:* the weighted congestion levels of 1, 2 and 3 hop neighbours are summed with an emphasis on local information (the weights 3, 2 and 1 are assigned to 1, 2 and 3 hop neighbours, respectively).
- *When the source and destination cluster agents are not located at the same row or column:* the weighted congestion levels of the directly connected neighbour and the neighbouring cluster are summed with an emphasis on local information (the weights 3 and 2 are assigned to the directly connected neighbour and the neighbouring cluster, respectively).

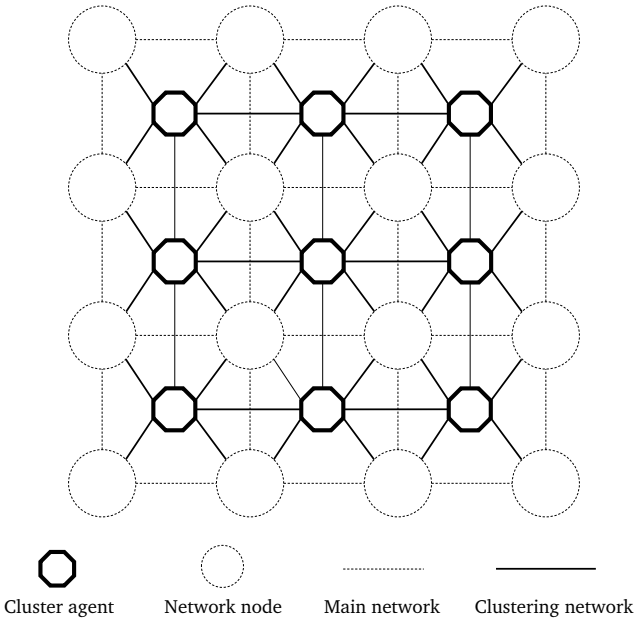


Figure 3.10: Agent-based network

3.7.2 Regional congestion awareness

Harvesting remote congestion information typically requires a separate congestion propagation network. Due to the large abandoned space on chip and with the use of dedicated control wires, constructing such network alongside the actual data network is a straightforward task.

Regional Congestion Awareness (RCA) is a family of scalable, light-weight schemes that include congestion related information, propagated from several parts of the network, in the route selection process [95].

It relies on a monitoring network to allow the exchange of congestion information amongst adjacent nodes, wherein each node aggregates its local congestion estimate with that of neighbouring nodes and propagates it upstream.

Because different weights are assigned to different nodes, emphasizing local congestion information makes RCA act as a locally-aware scheme, whereas putting too much weight on remote congestion, increases the risk of making decisions based on distant and possibly unreachable parts of the network.

Three RCA variants were proposed including RCA-1D, where the weights are halved with each hop from the source node all the way along the X and Y di-

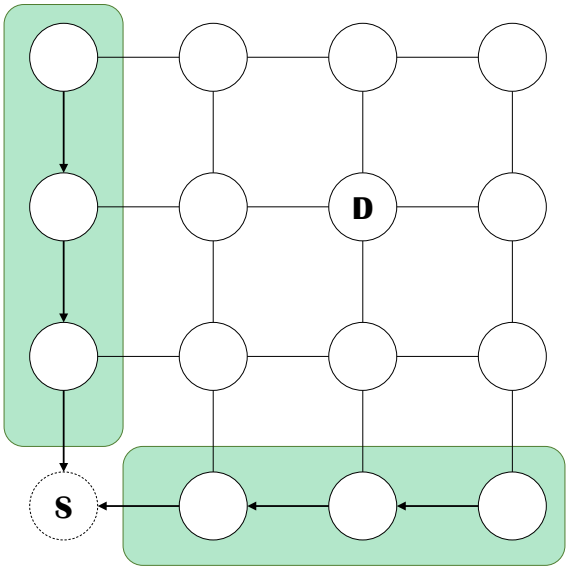


Figure 3.11: Regional congestion awareness

mensions (Figure 3.11). Congestion along both dimensions in mesh-based NoC is then assessed and the less congested route is chosen for packet transmission (determined based on the sum of each node's occupied buffer slots × the assigned weight).

3.7.3 Destination based adaptive routing

Destination Based Adaptive Routing (DBAR) is designed based on the fact that an efficient routing scheme combines high adaptivity with dynamic workload isolation and that both redundant and insufficient congestion information, deteriorate the overall performance [96].

With DBAR, only nodes residing at the minimal path along the X and Y dimensions are considered, while other nodes are completely ignored (Figure 3.12).

A lightweight congestion propagation network is constructed to leverage both local and remote congestion information in path selection. The number of available virtual channels is used as congestion metric, and is forwarded by every node in the network to every other node residing at the same dimension.

Accordingly, every node has two registers used to store the congestion information received from the X and Y dimensions.

To further enhance the design, nodes forward congestion information in a

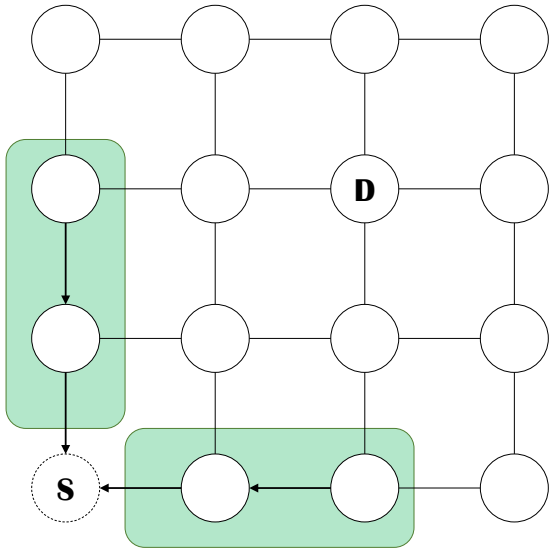


Figure 3.12: Destination based adaptive routing

on/off manner. A threshold is set to 4 occupied channels out of a total of 8 available, in which case a 0 is forwarded indicating a congested node. When more than 4 virtual channels are available, 1 is forwarded to indicate the absence of congestion.

Table 3.1 summarizes the aforementioned routing algorithms and their characteristics.

3.8 Performance evaluation

To evaluate the performance of a given routing scheme, designers pay attention to a couple of key metrics, average packet latency and average accepted packet rate (throughput).

Latency is defined as the number of cycles between the injection of a message issued by a processing core and the time when the message is completely delivered to the destination node. Throughput on the other hand is defined as the average number of packets arriving per core per clock cycle.

An ideal routing scheme would exhibit a balance between low latency at low traffic and high saturation throughput as traffic increases.

Table 3.1: Routing algorithms description summary

Algorithm	Type	VCs	Congestion	Deadlock avoidance	Ref
DOR	Deterministic	No	Oblivious	Restricted turns	-
Flooding	Random	No	Oblivious	Unsupported	[79]
Hot-Potato	Random	No	Oblivious	Deflection	[82]
Valiant	Random	Yes	Oblivious	Virtual channels	[83]
ROMM	Random	Yes	Oblivious	Virtual channels	[84]
O1TURN	Random	Yes	Oblivious	Virtual channels	[85]
Turn models	Adaptive	No	Locally aware	Restricted turns	[86]
Odd-even	Adaptive	No	Locally aware	Restricted turns	[88]
DyAD	Hybrid	No	Locally aware	Restricted turns	[65]
DyXY	Adaptive	Yes	Locally aware	Virtual channels	[90]
NoP	Adaptive	Yes	Locally aware	Virtual channels	[93]
Agent-based	Adaptive	Yes	Globally aware	Virtual channels	[59]
RCA	Adaptive	Yes	Globally aware	Virtual channels	[95]
DBAR	Adaptive	Yes	Globally aware	Virtual channels	[96]

3.8.1 Traffic patterns

To test the efficiency and reliability of a given routing scheme and test its limits as well as have a good understanding of its pros and cons, designers apply a set of traffic patterns, several of which are based on communication patterns that arise in particular applications.
The most common traffic patterns applied are the following:

- *Uniform random:* each node sends packets to other nodes with an equal probability. By making the traffic uniformly distributed, the load will be evenly balanced across the network.
- *Local:* packets are sent to the nearby neighbouring nodes with higher probability than remote ones.
- *Hotspot:* more packets are destined to a specific set of nodes in the network (hotspots) with higher than average probability.
- *Bit permutation:* all traffic generated at a given source node is destined to a single destination, which is computed by permuting the bits of the source node. With bit reverse for instance, the destination node's id is the reversed id of the source node (e.g. source node with id = 6 (110) communicates with destination node with id = 3 (011)).
- *Transpose:* each node sends packets to another node with the address of the reversed dimension index: node (i , j) sends packets to node (Nx - i , Ny - j), Nx and Ny being the network dimensions along the X and Y dimensions, respectively.
- *Negative exponential distribution:* with which the likelihood that a node sends a packet to another node, exponentially decreases with the hop distance between the two nodes [97].

3.8.2 Evaluation metrics

The average packet latency and average accepted packet rate (throughput) are typically reported by means of a curve for different packet injection rates under various traffic patterns.

The latency curve has a distinct shape that starts at the zero-load value, which is the delay experienced by a packet in the absence of contention in the network.

As the packet injection rate increases, more and more packets will contend for resources and will be stalled, which gives rise to congestion, consequently, latency will also increase.

Eventually, all packets will experience very high latencies and the network will no longer be able to handle the load. Henceforth, the curve will approach the vertical asymptote.

The throughput curve on the other hand starts at zero and rises gradually, as the greater the number of injected packets, the higher the throughput will be.

Ultimately, the saturation point is reached, at which the network is no longer able to deliver packets as fast as they are created, thus the curve is somewhat flattened. The network might as well become unstable, causing the throughput to drop dramatically as the load increases beyond saturation.

3.8.3 Experimental results

We conducted a set of experiments with regard to average packet latency and average throughput, to make a comparison between adaptive, deterministic and stochastic routing under various system configurations and took note of the interesting results.

We used a cycle accurate on chip network simulator developed with C++ to evaluate the performance of an 8×8 mesh network [98].

The comparison took place between the deterministic dimension ordered scheme, the stochastic randomized valiant scheme and the adaptive dynamic XY routing. Packet size was set to 4 flits with a buffer depth of 8 flits, with wormhole switching and credit-based buffer management. The full system configuration is depicted in Table 3.2.

We first applied a random, uniformly distributed traffic pattern. From the average latency results depicted in Figure 3.13, the deterministic dimension ordered routing clearly outperforms the stochastic valiant scheme. Additionally, it slightly outperforms the adaptive dynamic XY as well, which is all due to the

Table 3.2: System configuration for routing assessment

Topology	Mesh
Network dimensions	8 × 8
Flit size	32 bits
Packet size	4 flits
Buffer depth	8 flits
Warm-up time	1000 cycles
Simulation time	20000 cycles
Traffic patterns	Uniform random, Transpose
Routing scheme	dor, valiant, dynamic xy
Switching technique	Wormhole
Buffer management	Credit-based

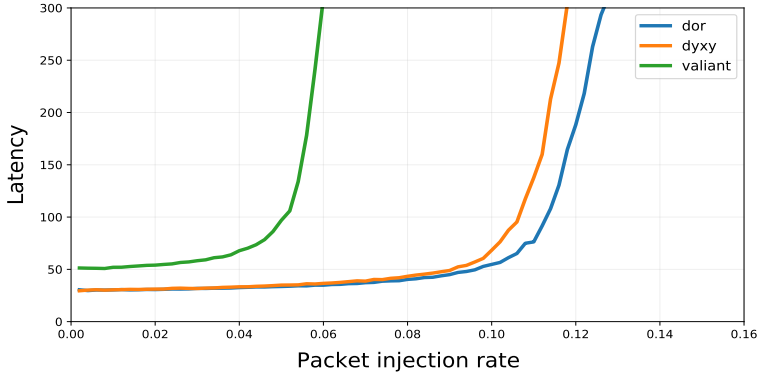


Figure 3.13: Latency evaluation under uniform traffic

traffic's uniform distribution, as deterministic schemes are well suited for this kind of traffic.

From an average throughput perspective (Figure 3.14), valiant reaches saturation rather quickly at an approximate injection rate of 0.06. Dor and dyxy on the other hand, achieve a much higher saturation throughput beyond the 0.11 injection rate mark, which solidifies the average latency results. The gap between the two is rather small as they yield approximately the same saturation throughput.

Under the transpose traffic, dor performs even worse than the stochastic valiant, while the adaptive dyxy clearly outperforms them both (Figure 3.15). Which consolidates the superiority of adaptive routing schemes under unpredictable and non-uniform traffic patterns.

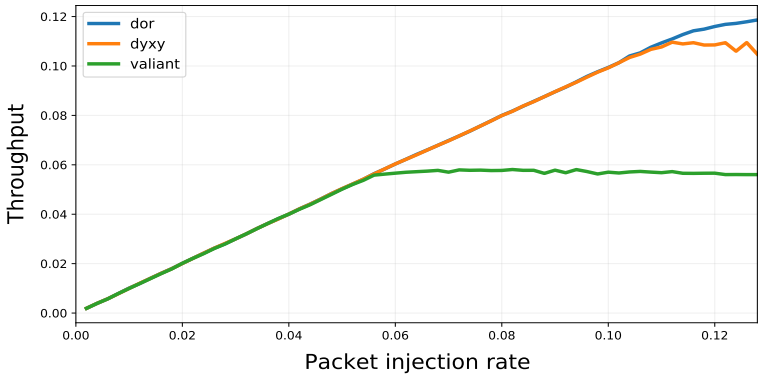


Figure 3.14: Throughput evaluation under uniform traffic

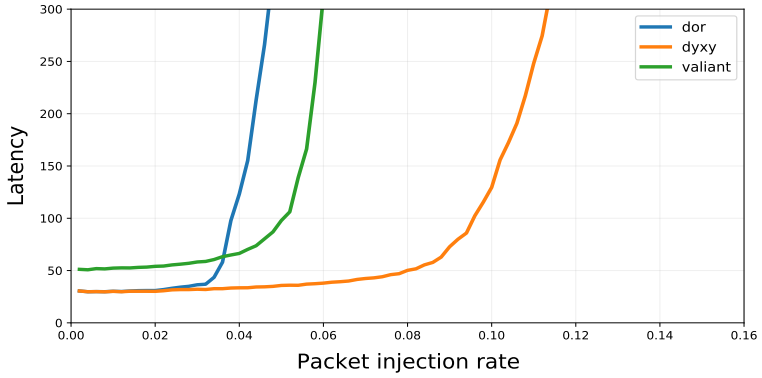


Figure 3.15: Latency evaluation under transpose traffic

3.8.4 Evaluation of 3D based networks

3D based networks on chip are regarded as an evolution of 2D based networks and the future communication infrastructure for very complex systems on chip with thousands of integrated cores.

We conducted a set of experiments to evaluate the performance and compare between a 2D and a 3D network on chip in terms of both average packet latency and throughput.

Both networks are mesh-based with 64 nodes in total, with 8×8 on the 2D layout and $4 \times 4 \times 4$ on the 3D layout. For the sake of a fair comparison, both systems were configured similarly: each input port is associated with 4 virtual channels with a buffer depth of 8 flits each. Packet size was set to 4 flits and the deterministic dimension ordered routing is employed under random uniform traffic. The full system configuration is depicted in Table 3.3.

Table 3.3: System configuration for topology assessment

Topology	2D mesh, 3D symmetric mesh
Nodes count	64
Network dimensions	8×8, $4 \times 4 \times 4$
Flit size	32 bits
Packet size	4 flits
Buffer depth	8 flits
Traffic patterns	Uniform random
Routing scheme	dimension order routing
Switching technique	Wormhole
Buffer management	Credit-based

The results reveal the 3D network's evident superiority over its 2D counterpart, as it reaches a much higher saturation throughput (Figure 3.16) and exhibits better latency (Figure 3.17). All due to the reduced distance between communicating nodes which gives rise to a much lower hop count.

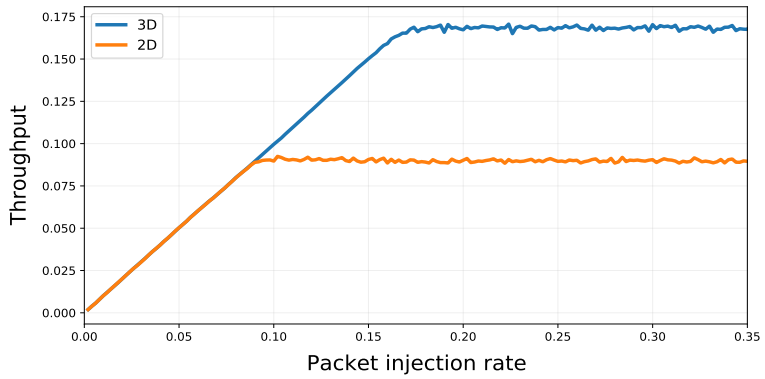


Figure 3.16: 3D vs 2D throughput

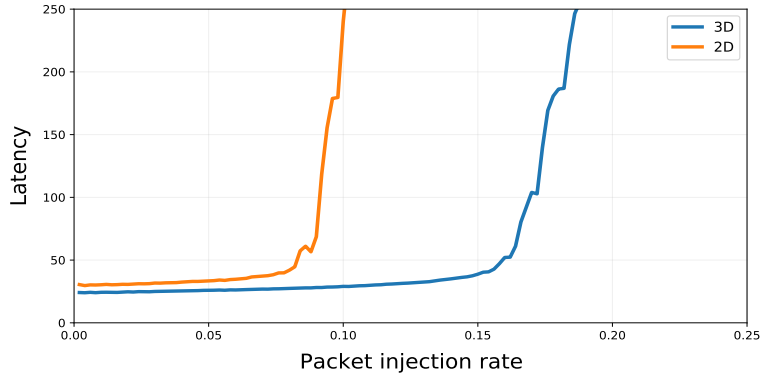


Figure 3.17: 3D vs 2D latency

3.9 Conclusion

In this chapter we have explored an exhaustive list of on-chip routing schemes, from which we were able to extract the pertinent features that we believe are most beneficial in the design of reliable routing algorithms for networks-on-chip.

When it comes to deadlock avoidance, virtual channels and restricted turns are primarily the sought out methods. The former are the most widely adopted with buffered flow control.

Congestion is predominantly determined based on either the number of occupied (or free) buffer slots or, the number of available (or busy) virtual channels. It can be propagated in the form of flags, which stem from aggregating congestion values or setting up a threshold.

In addition, putting too much emphasis on the congestion values of directly connected neighbours renders the scheme exclusively locally aware. The latter

will have a narrow field of vision and will not be able to avoid remote congested regions in the network.

On the other hand, putting too much emphasis on remote congestion values, will introduce too much noise in the decision-making process by taking into account the information of unreachable remote nodes.

Chapter 4

Reliable Congestion Aware Routing

Congestion awareness is deemed a key feature in the design of reliable routing schemes for on-chip networks, because of its profound impact on performance by stalling packets and increasing delays. Nonetheless, such task is by no means trivial, as locally aware schemes have a rather narrow field of vision, which could lead to poor decision-making. Globally aware routing on the other hand, requires complex computation units and introduces the collection of excessive and sometimes irrelevant information.

In this chapter, we present two reliable, fully adaptive and congestion aware routing algorithms for mesh-based network on chip. The proposed algorithms do not rely solely on local congestion information nor on irrelevant global information, they rather leverage both in path selection and provide somewhat a compromise between locally and globally aware routing.

We also propose two moderately distinct versions for each algorithm, wherein congestion in parts of the network close, not only to the source, but also to the destination node is investigated. For that purpose, we construct a congestion propagation network that can be used not only in the context of our work but also as a standalone work.

Experimental results showcase the proposed algorithms' edge over state of the art NoC routing algorithms for various configurations in terms of both average latency and throughput.

4.1 Motivation

As previously demonstrated with regional congestion aware routing (RCA), the weights are halved with each hop from the source node all the way to the edges of the network along both dimensions, wherein the less congested route is selected. The congestion level is computed separately for each dimension by aggregating the congestion values of all the nodes it accommodates.

The main drawback with this approach is including irrelevant congestion information in the decision-making process.

Figure 4.1 exhibits the shortcomings of such approach, with source node S and destination node D.

Although node 0 resides in the non-minimal path, it will be taken into account and will intervene in path selection (Figure 4.1a):

- *X dimension congestion level* $= 0(\frac{1}{2}) + 1(\frac{1}{4}) + 3(\frac{1}{8}) = 0.625$
- *Y dimension congestion level* $= 0(\frac{1}{2}) + 2(\frac{1}{4}) + 3(\frac{1}{8}) = 0.875$

Therefore, node 13 along the X dimension is chosen as the next hop. The latter will forward packets to its eastern neighbour node 14 because congestion along the X dimension is less than that of the Y dimension (Figure 4.1b):

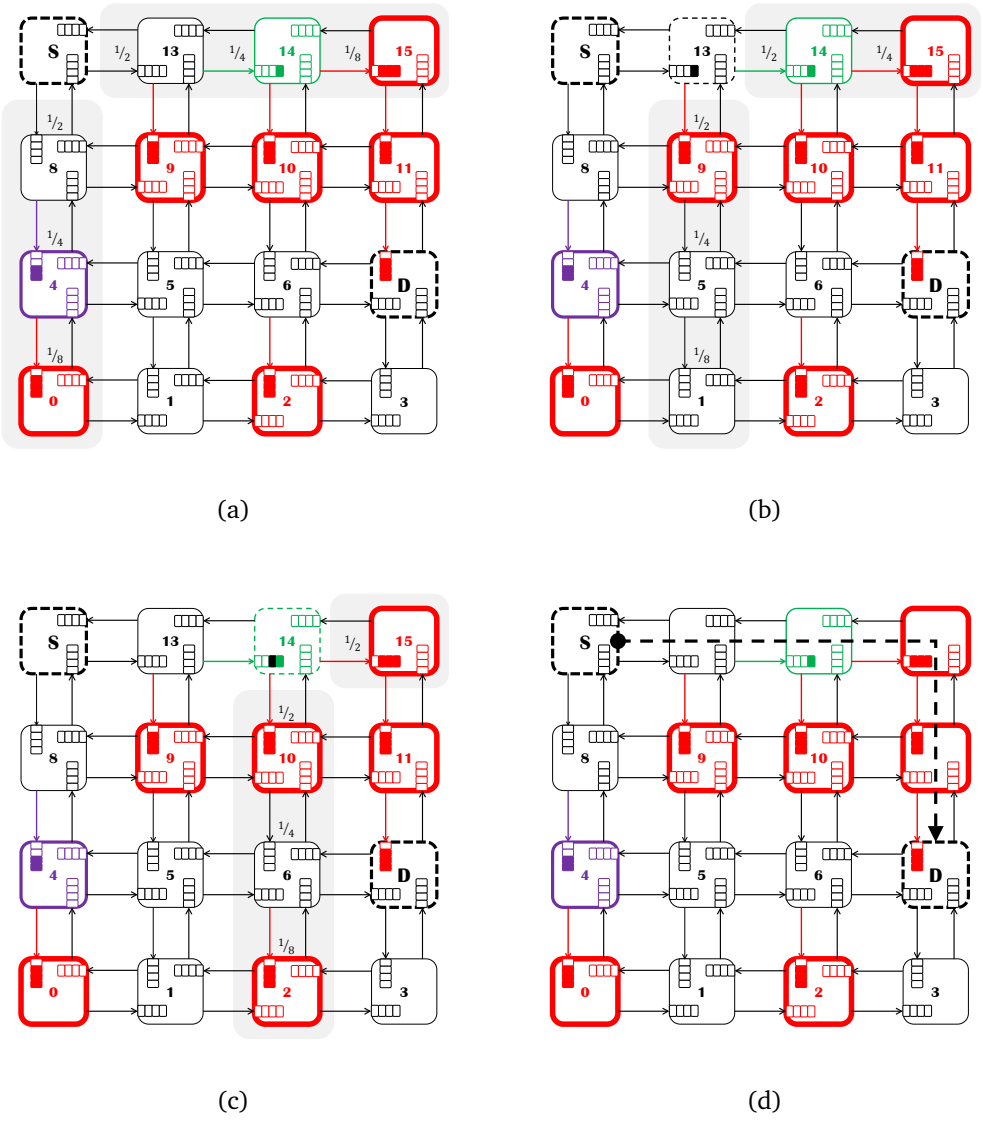
- *X dimension congestion level* $= 1(\frac{1}{2}) + 3(\frac{1}{4}) = 1.25$
- *Y dimension congestion level* $= 3(\frac{1}{2}) + 0(\frac{1}{4}) + 0(\frac{1}{8}) = 1.5$

At node 14 (Figure 4.1c):

- *X dimension congestion level* $= 3(\frac{1}{2}) = 1.5$
- *Y dimension congestion level* $= 3(\frac{1}{2}) + 0(\frac{1}{4}) + 3(\frac{1}{8}) = 1.875$

Eventually, packets will have to go through the highly congested route with nodes 14, 15 and 11 (Figure 4.1d). Which could have been avoided altogether if nodes not belonging to the minimal path, such as node 0 and node 2 were excluded from the decision-making process.

This happens to be the case of destination based adaptive routing (DBAR), where only nodes residing at the minimal path along the X and Y dimensions



(a) (b)

(c) (d)

Figure 4.1: RCA shortcomings

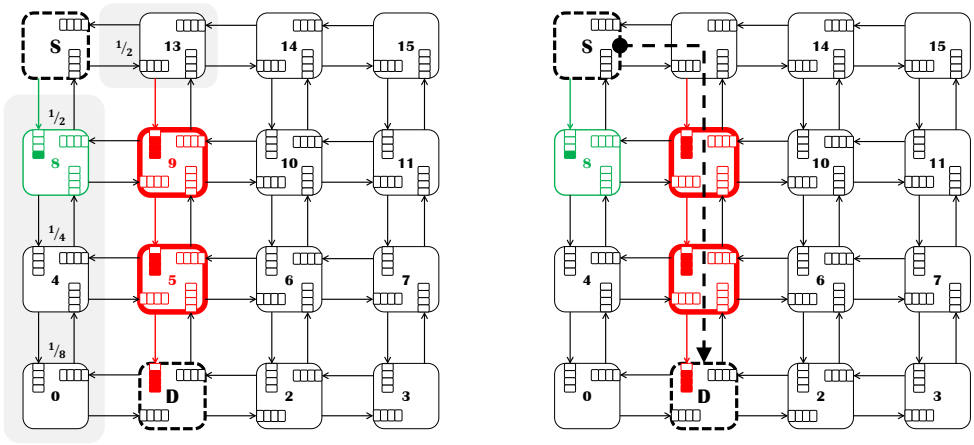
are considered. Congestion estimation is similar to that of RCA and weights are also halved with each hop.

The major problem emerges when the number of nodes along one dimension is greater than that along the other.

In Figure 4.2a, with the pair of source and destination nodes S and D respectively. Along the X dimension, only one node (13) resides in the minimal path (X congestion level = 0) compared to three nodes (8, 4 and 0) along the Y dimension (Y congestion level = 0.5), the unfair comparison triggers a poor decision, and packets will be forwarded through the highly congested area encompassing the two nodes 9 and 5 (Figure 4.2b).

The whole scenario could have been avoided if the source node had had some information regarding congestion in the region close to the destination node.

Which motivated us to come up with a novel, weighted minimal congestion-aware routing scheme, that eliminates the collection of irrelevant congestion-related information and the unfair comparison problems inflicted by both RCA and DBAR. It offers somewhat a middle ground between all the aforementioned extremes.



(a) (b)

Figure 4.2: DBAR shortcomings

4.2 Dynamic XY-YX

Every node in the mesh network, computes the congestion level along the XY as well as the YX routes, which is defined as the sum of the congestion levels of all the nodes between the source and destination pair.

- Node congestion level = occupied slots × assigned weight
- Route congestion level = $\sum_{i=1}^{n} \text{node}_i$ congestion level

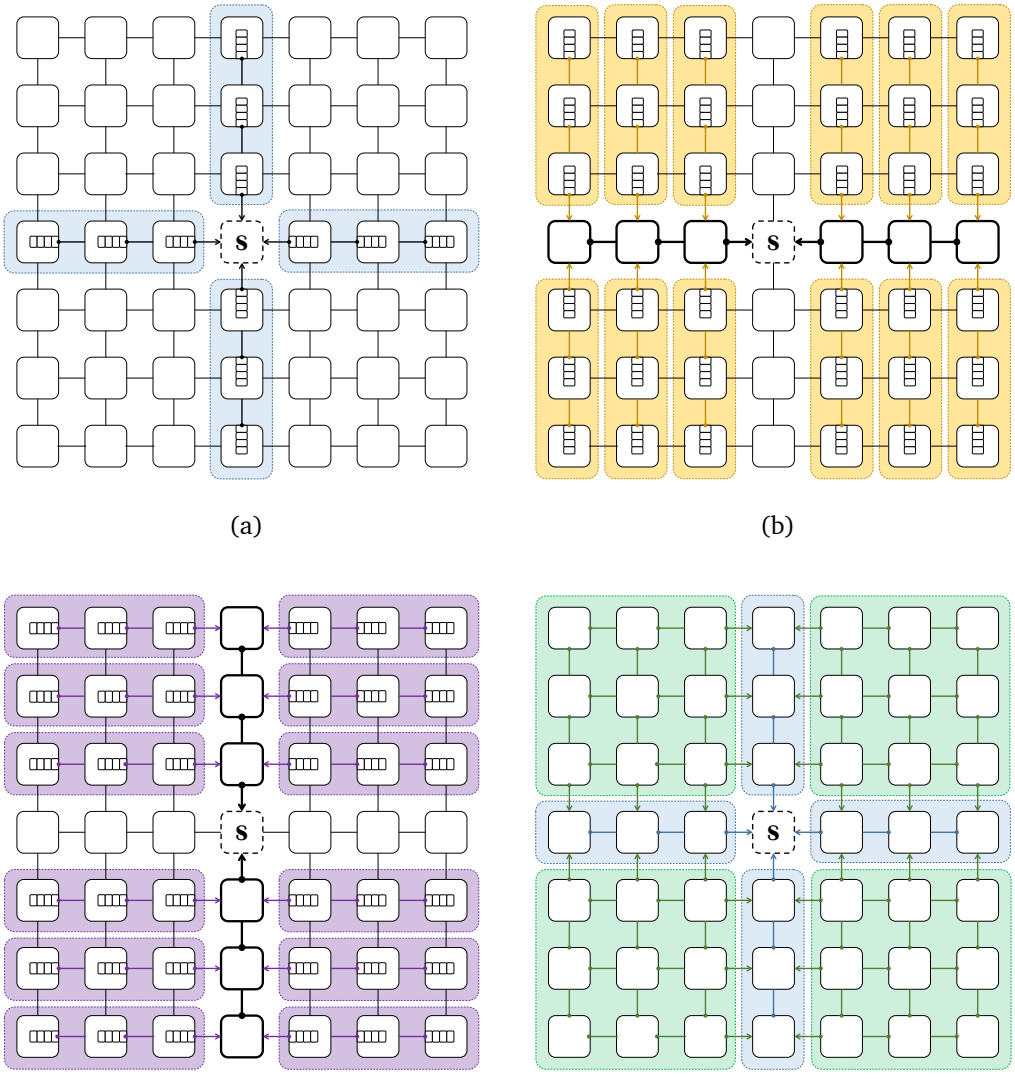
To lay hold of the number of occupied buffer slots of all the nodes along the XY and YX routes between any pair of communicating nodes, we constructed a congestion propagation network.

4.2.1 Congestion propagation network

Each node in the network keeps track of its buffers' occupied slots, in all four cardinal directions. It then transmits the information to the downstream neighbours. The number of the eastern (western) buffer's occupied slots is transmitted east (west), and the number of the northern (southern) buffer's occupied slots is transmitted north (south). The downstream nodes will then embed the received information with their own buffers' occupied slots and relay the bulk to their own downstream neighbours [99].

Eventually, every node in the network ends up with four congestion vectors (eastern, western, northern and southern) and will be aware of congestion along the entire row and column in which it resides within the mesh network (all the nodes along the X and Y dimensions are covered).

From the viewpoint of node S in the example depicted in Figure 4.3a, the neighbour node to the extreme north (three hops along the Y dimension) extracts its own southern buffer occupied slots and sends the information to its southern neighbour. The latter extracts its own southern buffer occupied slots, attaches it with the received information (constructs a congestion vector) and relays it to its southern neighbour. Which in turn does the same and sends the bulk to node S. The latter will be aware of congestion information, related to all of its northern neighbours.



(a) (b)

(c) (d)

Figure 4.3: Proposed propagation network

The same process takes place along the east, west and south. Accordingly, node S will end up with the four vectors and will be aware of congestion related to all the nodes along the entire row and column in which it resides (depicted in blue).

Alongside the four independent vectors, each node constructs a congestion map as follows:

- The eastern and western congestion vectors are relayed north and south.
- The northern and southern congestion vectors are relayed east and west.

In Figure 4.3b, each eastern (western) neighbour of node S along the same row, retrieves the congestion vectors of its northern and southern neighbours, then relays the information west (east). The downstream nodes will do the same until the map of vectors reaches node S, containing all the congestion vectors of all the remote eastern and western nodes (depicted in orange).

The northern and southern neighbours of node S along the same column, will also retrieve the congestion vectors of their eastern and western neighbours, then relay the information north and south, until the map of vectors reaches node S, containing all the congestion vectors of all the remote northern and southern nodes in the network (depicted in violet in Figure 4.3c).

By doing so, a full view of the network is enabled and decisions are made based on the global congestion level (Figure 4.3d).

4.2.2 Routing and selection

Dynamic XY-YX (DyXYYX) is based on a minimal fully adaptive routing function, the pseudo code of which is illustrated in Algorithm 4.2.1.

The route congestion level computation will then take place and serve as input to the selection function, the pseudo code is illustrated in Algorithm 4.2.2.

Finally, the selection function is triggered with the pseudo code depicted in Algorithm 4.2.3.

Algorithm 4.2.1: Minimal fully adaptive routing function

Data: positions of current and destination nodes
Result: list of eligible routes for packet transmission

if *(Current and destination nodes reside at the same column or row)* **then**
  send packets directly along the X or Y direction ;
else if *(Destination node is located north-east)* **then**
  return [north, east] ; `/* north and east directions */`
else if *(Destination node is located south-east)* **then**
  return [south, east] ; `/* south and east directions */`
else if *(Destination node is located north-west)* **then**
  return [north, west] ; `/* north and west directions */`
else
  return [south, west] ; `/* south and west directions */`
end if

Algorithm 4.2.2: Route congestion level computation

Data: current and destination nodes positions
Result: congestion level along the XY or YX route

route_congestion_level $\leftarrow \sum_{i=1}^{n}$ node$_i$ occupied slots $\times$ assigned weight ;
`/* congestion level sum of all the nodes along the XY or YX route */`
return route_congestion_level ;

Algorithm 4.2.3: Selection function

Data: XY route congestion level, YX route congestion level
Result: the next hop along the less congested route

if *(XY route congestion level < YX route congestion level)* **then**
  return neighbour node along the X dimension
else if *(XY route congestion level > YX route congestion level)* **then**
  return neighbour node along the Y dimension
else
  `// check directly connected neighbours`
  if *(X neighbour congestion level ≠ Y neighbour congestion level)* **then**
    return neighbour node with the smallest congestion level
  else
    return an arbitrary neighbour node `/* last resort */`
  end if
end if

4.2.3 Versions and weight assignment

Two distinct versions in terms of weight distribution, DyXYYXV1 and DyXYYXV2 are proposed:

- With DyXYYXV1, weights are halved with each hop from the source all the way to the destination node along both the XY and YX routes (Figure 4.4 with the pair of communicating nodes S and D, respectively).
- With DyXYYXV2, weights are halved with each hop taken from the source node all the way to the destination column (row), and halved with each hop from the destination node all the way to the source row (column) (Figure 4.4 with the pair of communicating nodes T and D, respectively).

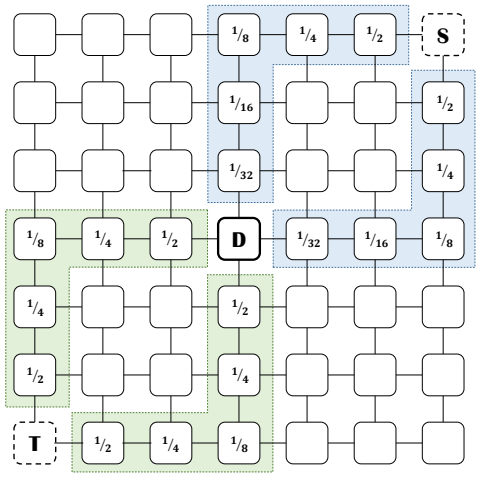


Figure 4.4: DyXYYX weight assignment

4.2.4 Routing illustration

In Figure 4.5, an illustration of DyXYYXV1 routing is depicted, with the pair of communicating source and destination nodes S and D, respectively:

- At the source node S (Figure4.5a), congestion levels along the XY and YX routes is assessed for path selection, by summing the congestion levels of nodes 13, 14, 15, 11 and 7 and comparing it with the sum of congestion levels of nodes 8, 4, 0, 1 and 2 (a fair comparison between five nodes in each direction).

  - *XY congestion level* $= 1(\frac{1}{2}) + 2(\frac{1}{4}) + 3(\frac{1}{8}) + 3(\frac{1}{16}) + 1(\frac{1}{32}) = 1.59375$
  - *YX congestion level* $= 0(\frac{1}{2}) + 2(\frac{1}{4}) + 2(\frac{1}{8}) + 3(\frac{1}{16}) + 0(\frac{1}{32}) = 0.9375$

  $\Rightarrow$ YX Route is chosen and packets are forwarded towards node 8.

- At node 8 (Figure 4.5b), congestion levels along the XY route (nodes 9, 10, 11 and 7) and YX route (4, 0, 1 and 2) are assessed (a fair comparison between four nodes in each direction).

  - *XY congestion level* $= 0(\frac{1}{2}) + 1(\frac{1}{4}) + 3(\frac{1}{8}) + 1(\frac{1}{16}) = 0.6875$
  - *YX congestion level* $= 2(\frac{1}{2}) + 2(\frac{1}{4}) + 3(\frac{1}{8}) + 0(\frac{1}{16}) = 1.875$

  $\Rightarrow$ XY Route is chosen and packets are forwarded towards node 9.

- At node 9 (Figure 4.5c), congestion along the XY route (nodes 10, 11 and 7) and YX route (5, 1 and 2) is assessed (a fair comparison between three nodes in each direction).

  - *XY congestion level* $= 1(\frac{1}{2}) + 3(\frac{1}{4}) + 1(\frac{1}{8}) = 1.375$
  - *YX congestion level* $= 1(\frac{1}{2}) + 3(\frac{1}{4}) + 0(\frac{1}{8}) = 1.25$

  $\Rightarrow$ YX Route is chosen and packets are forwarded towards node 5.

- At node 5 (Figure 4.5d), congestion along the XY route (nodes 6 and 7) and YX route (1 and 2) is assessed (a fair comparison between two nodes in each direction).

  - *XY congestion level* $= 0(\frac{1}{2}) + 1(\frac{1}{4}) = 0.25$
  - *YX congestion level* $= 3(\frac{1}{2}) + 0(\frac{1}{4}) = 1.5$

  $\Rightarrow$ XY Route is chosen and packets are forwarded towards node 6.

- At node 6 (Figure 4.5e), congestion along the XY route (node 7) and YX route (node 2) is assessed (a fair comparison between one node in each direction).

  - *XY congestion level* $= 1(\frac{1}{2}) = 0.5$
  - *YX congestion level* $= 0(\frac{1}{2}) = 0$

  $\Rightarrow$ YX Route is chosen and packets are forwarded towards node 2.

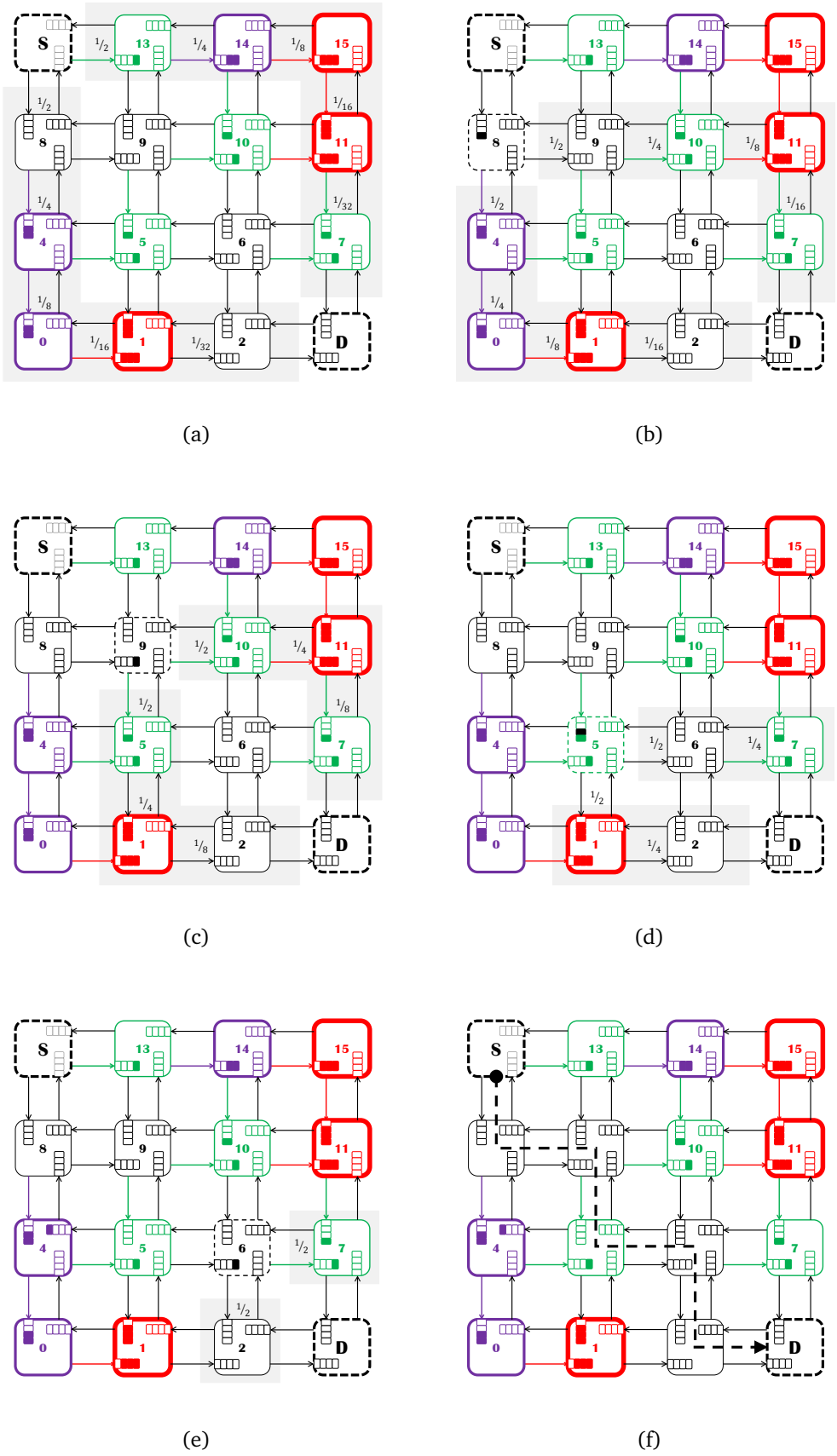


Figure 4.5: DyXYYX routing demonstration

Discussion

Packets will avoid the most congested regions in the network to eventually reach their destination, all due to the fact that each node has information about congestion, not only in close proximity to the source but also to the destination node (Figure 4.5f).

By applying DyXYYX instead of RCA in Figure 4.1, packets will reach the destination through nodes 8, 4, 5 and 6 avoiding the highly congested nodes 9, 10, 11 and 15. All due to the fact that DyXYYX excludes congestion information of nodes residing at the non-minimal path (0, 1 and 2).

Additionally, in the example depicted in Figure 4.2 and instead of DBAR, with DyXYYX, packets will reach the destination through nodes 8, 4 and 0 and avoid the highly congested nodes 9 and 5. All due to the fact that DyXYYX conducts a fair comparison between the same number of nodes along the XY and YX routes.

4.2.5 Experimental results

To evaluate the performance of the proposed routing algorithm, its two versions, alongside both DBAR and RCA were all implemented with a fully adaptive routing function (to ensure a fair comparison). As congestion metric, the free buffer slots at the neighbouring nodes is used (DBAR originally relies on the number of free virtual channels, but they both exhibit the same results [96].

The cycle accurate simulator was warmed up for 1000 cycles and then launched for a period of 20000 cycles. Each input port is associated with a buffer depth of 12 flits with packets having a variable size ranging between 2 and 16 flits (32 bits each) in the 16 × 16 mesh network. The full system configuration is outlined in Table 4.1.

Wormhole switching is used, the most adopted switching technique for NoC routers as it requires smaller buffers compared to the store and forward switching scheme [100].

As performance metrics, throughput and average latency are considered. Latency, defined as the number of cycles between the injection of the header flit of a message issued by a processing element, and the time when the tail flit is

Table 4.1: System configuration

Topology	Mesh
Network dimensions	16 × 16
Flit size	32 bits
Packet size	2 to 16 flits
Buffer depth	12 flits
Warm-up time	1000 cycles
Simulation time	20000 cycles
Traffic patterns	Uniform random, Hotspot, Transpose
Routing function	Fully adaptive
Switching technique	Wormhole

delivered to the destination node. Throughput on the other hand is defined as the average number of flits arriving per IP block per clock cycle.

To test the limits of the proposed scheme, three traffic patterns are applied: Uniform random, Hotspot and Transpose.

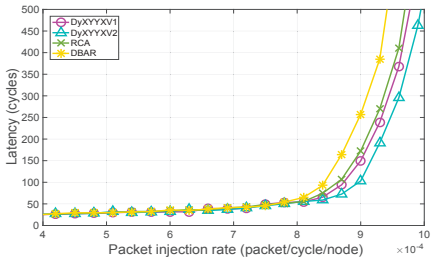
With the uniform pattern, each node sends packets to another random node in the network with an equal probability of selection, whereas in the case of hotspot traffic, communicating nodes are chosen randomly with more packets destined to a set of specific nodes in the network (hotspots) with higher than average probability.

With the transpose traffic, each node sends packets to another node with the address of the reversed dimension index: node (i, j) sends packets to node (Nx - i, Ny - j), Nx and Ny being the network dimensions in X and Y directions, respectively.

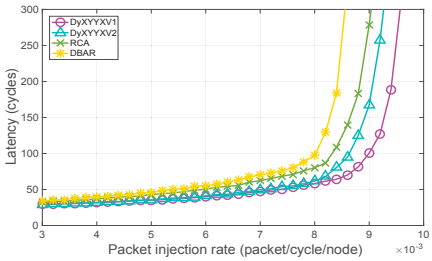
4.2.6 Discussion

With regard to latency, under the hotspot traffic with the four central nodes 119, 120, 135 and 136 designated as hotspots receiving 20% more than average traffic: DyXYYXV2 outperforms DyXYYXV1, DBAR and RCA (Figure 4.6).

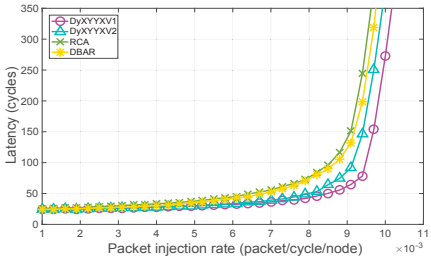
This is mainly due to the fact that, congestion is predominantly high in the centre of the network and with the weight distribution of DyXYYXV2, congestion information close to the destination node is emphasized in the decision making



(a) Hotspot



(b) Uniform



(c) Transpose

Figure 4.6: DyXYYX average latency

process.

DyXYYXV1 prevails under the other two traffic patterns. It exhibits an average improvement of 37.91% and 29.16% as opposed to DBAR and RCA respectively (Table 4.2), in terms of average latency measured for each traffic pattern at an injection load right below the saturation point, the point where the network is no longer able to deliver packets as fast as they are created.

In our experiments, we set the saturation point to where latency is three times higher than the zero-load latency, which is experienced by packets during the complete absence of congestion in the network.

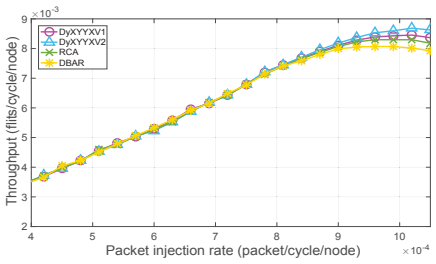
The results imply that despite the fact that remote congestion information is

Table 4.2: Average latency improvement

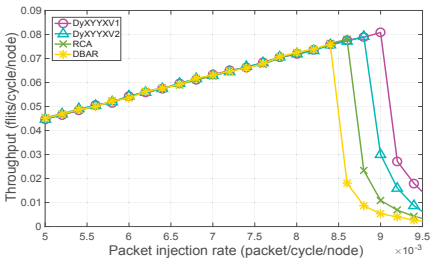
Traffic pattern	**Injection rate**	**Average latency (cycles)**			**Improvement rate**	
		DBAR	RCA	DyXYYX	vs. DBAR	vs. RCA
Uniform	0.008	97.5508	80.2607	58.1906	40.34%	27.49%
Transpose	0.0085	89.6696	96.283	50.2609	43.94%	47.73%
Hotspot	0.00084	92.7938	74.6104	65.4567	29.46%	12.26%
			Average improvement		37.91%	29.16%

highly useful for making more efficient decisions, it ought not be as emphasized as local congestion.

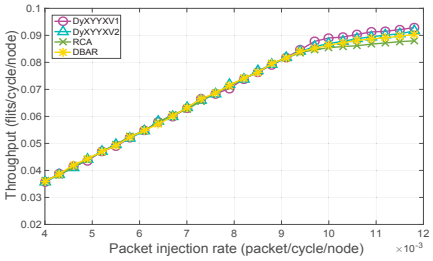
With regard to throughput, while DyXYYXV2 reaches a higher saturation point in the hotspot traffic than DyXYYXV1. The latter, exhibits an enhanced throughput under the transpose and uniform traffics in contrast to the former, and both algorithms outperform DBAR and RCA respectively (Figure 4.7). A high saturation throughput indicates that the network can accept a large amount of traffic before all packets experience very high latencies.



(a) Hotspot



(b) Uniform



(c) Transpose

Figure 4.7: DyXYYX average throughput

In Table 4.3, the improvement rate of DyXYYXV1 compared to DBAR and RCA in terms of packet injection rate at the saturation point is depicted. The proposed scheme performs better than DBAR and RCA by an average of 8.61% and 6.93% respectively.

4.2.7 Limitations

Note that the more nodes we have in the network, the more congestion information gathering is required, which happens to be an aspect of globally aware routing. This might have an undesired effect on the overall performance and

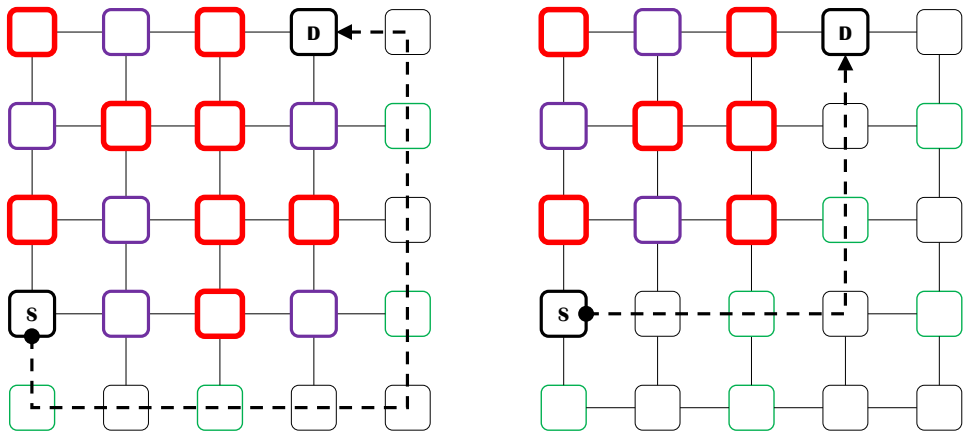
Table 4.3: Saturation injection rate improvement

Traffic pattern	Saturation injection rate			Improvement rate	
	DBAR	RCA	DyXYYX	vs. DBAR	vs. RCA
Uniform	0.0082	0.0084	0.009	9.75%	7.14%
Transpose	0.0088	0.0088	0.0097	10.22%	10.22%
Hotspot	0.00085	0.00087	0.0009	5.88%	3.44%
		Average improvement		8.61%	6.93%

therefore, the congestion propagation scheme ought to be improved by reducing the number of data points in the congestion map.

On the other hand, due to the minimal nature of the algorithm, we might encounter a scenario where the area in the minimal path is highly congested, yet packets will still have to move through it. In this case, an escape route through the non-minimal path would be more appropriate to deliver packets (Figure 4.8a).

Another scenario depicted in Figure 4.8b and even though it is clear from the very start that the XY route is the one to be traversed by packets to reach the destination node D, every node along that route will still unnecessarily assess congestion along the XY and YX routes (the computation time will certainly affect latency).



(a) (b)

Figure 4.8: DyXYYX pitfalls

4.3 Fully Adaptive Congestion Aware Routing Scheme

Unlike DyXYYX, Fully Adaptive Congestion Aware Routing Scheme (FACARS) does not make use of weights and instead of relying on the number of occupied buffer slots and having to compute each route's congestion level at every intermediate hop along the way, each node marks itself as being either congested or otherwise, by raising a congestion flag [101].

Each node keeps track of its own congestion level and raises flags with different values (depending on the algorithm version). The main congestion metric used is the number of free buffer slots. Accordingly, four flags are needed to cover all four cardinal directions.

4.3.1 Versions

Two slightly different versions are proposed, wherein network nodes are divided into categories as follows:

- With the first version (FACARSV1), nodes are divided into two categories: *congested* and *non-congested*.
- With the second version (FACARSV2), nodes are divided into three distinct categories: *non-congested*, *congested* and *highly congested*.

Each node belongs to each category in the following manner:

with FACARSV1:

- When the number of a given buffer's free slots is less than or equals the third of the total buffer slots, that node is labelled as *congested* by raising the flag of value '1', otherwise;
- The flag of value '0' is asserted, indicating a *non-congested* node.

The congestion flags of FACARSV1 are depicted in Figure 4.9a and the categorization pseudo code is outlined in Algorithm 4.3.1.

Algorithm 4.3.1: FACARSV1 node congestion level computation

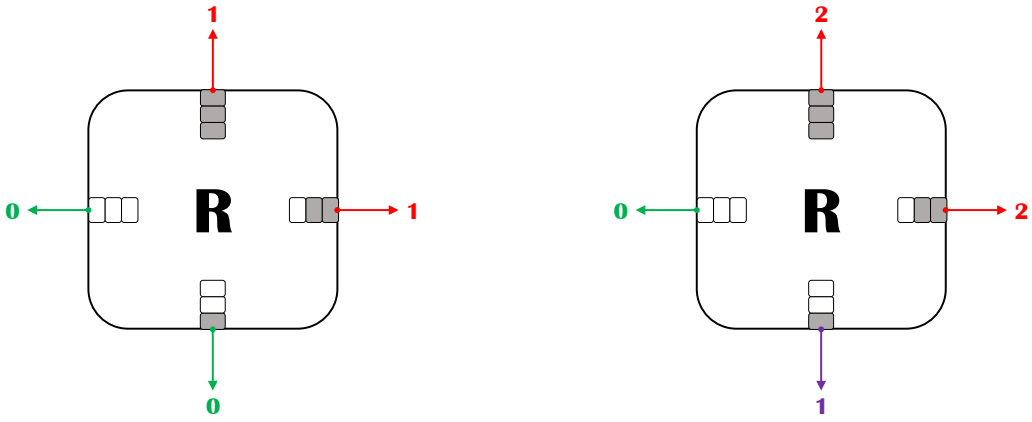
Data: free slots of a given buffer
Result: node congestion level (congestion flag)

if *(free buffer slots* $\leq \frac{1}{3}$ *buffer size)* **then**
  return 1 ;
else
  return 0 ;
end if

with FACARSV2:

- When the number of free slots of a given buffer is less than or equals the third of the total buffer depth, that node is deemed *highly congested* by raising the flag of value '2'.
- When the number of free buffer slots is enclosed between a third and two thirds of the total number of buffer slots, that node will be marked as *congested* by asserting the flag of value '1', otherwise;
- The value '0' is asserted indicating a *non-congested* node.

The congestion flags of FACARSV2 are depicted in (Figure 4.9b) and the categorization pseudo code is outlined in Algorithm 4.3.2.



(a) FACARSV1 (b) FACARSV2

Figure 4.9: Congestion flags

Algorithm 4.3.2: FACARSV2 node congestion level computation

Data: free slots of a given buffer
Result: node congestion level (congestion flag)

if *(free buffer slots* $\leq \frac{1}{3}$ *buffer size)* **then**
  return 2 ;
else if *(* $\frac{1}{3}$ *buffer size <* $\leq \frac{2}{3}$ *buffer size)* **then**
  return 1 ;
else
  return 0 ;
end if

4.3.2 Routing and selection

FACARS is implemented with a minimal fully adaptive routing strategy. Thus, every possible direction in the minimal path is passed to the selection function, with at most two directions at each intermediate hop. The routing function's pseudo code is outlined in Algorithm 4.3.3.

Algorithm 4.3.3: Minimal fully adaptive routing function

Data: position of current and destination nodes
Result: list of eligible routes for packet transmission

if *(Current and destination nodes reside at the same column or row)* **then**
  send packets directly along the X or Y direction ;
else if *(Destination node is located north-east)* **then**
  return [north, east] ; `/* north and east directions */`
else if *(Destination node is located south-east)* **then**
  return [south, east] ; `/* south and east directions */`
else if *(Destination node is located north-west)* **then**
  return [north, west] ; `/* north and west directions */`
else
  return [south, west] ; `/* south and west directions */`
end if

Every node in the network, computes the congestion level along the XY and YX routes, by aggregating the congestion flags associated with all the nodes on that route. The less congested route is then chosen for packet transmission. The route congestion level computation function is outlined in Algorithm 4.3.4.

When the two routes have the same congestion level (XY route congestion

Algorithm 4.3.4: Route congestion level computation

Data: current and destination nodes positions
Result: congestion level along the XY or YX route

route_congestion_level $\leftarrow \sum_{i=1}^{n}$ node$_i$ congestion flag ; `/* n denotes the number of nodes along the XY or YX route */`

return route_congestion_level ;

level = YX route congestion level), the number of occupied buffer slots at the neighbouring routers is then checked and the node with the smallest is selected (same behaviour as Dynamic XY routing). In case they are equal, the path is then chosen randomly. The selection function is outlined in Algorithm 4.3.5

Algorithm 4.3.5: Selection function

Data: XY route congestion level, YX route congestion level
Result: the next hop along the less congested route

if *(XY route congestion level < YX route congestion level)* **then**
  return neighbour node along the X dimension
else if *(XY route congestion level > YX route congestion level)* **then**
  return neighbour node along the Y dimension
else
  `// check directly connected neighbours`
  if *(X neighbour congestion level ≠ Y neighbour congestion level)* **then**
    return neighbour node with the smallest congestion level
  else
    return an arbitrary neighbour node `/* last resort */`
  end if
end if

4.3.3 Routing illustration

With FACARS and just like DyXYYX, DBAR's unfair comparison problem is alleviated. In addition, congestion information related to nodes close to the destination is obtained, which will help improve the decision-making process and the selection of the most suitable routes.

Referring to the example depicted in Figure 4.10 and by employing FACARSV2:

- At source node S (Figure 4.10a), congestion along the XY and YX routes

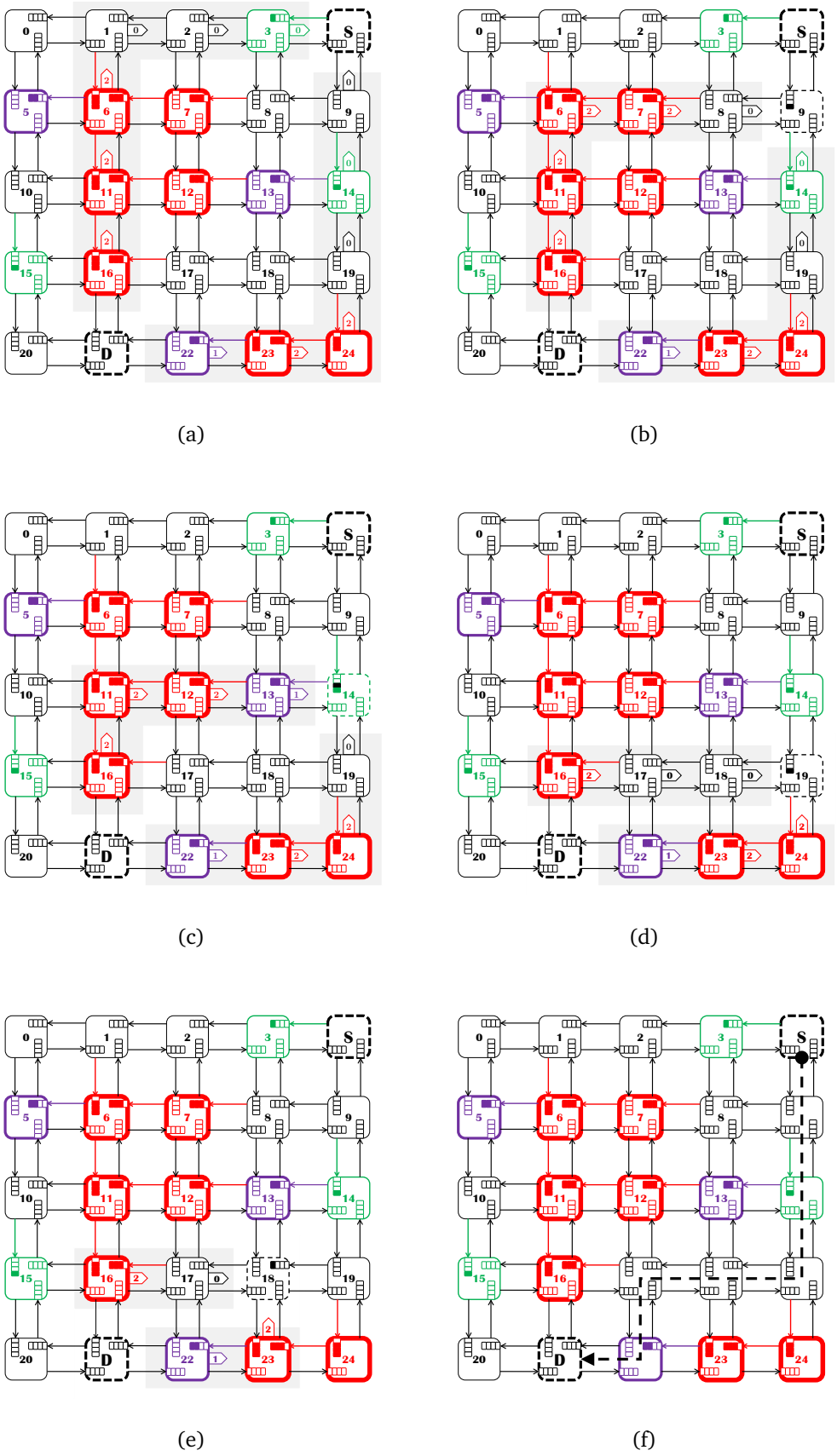


Figure 4.10: FACARS routing demonstration

is computed by aggregating the congestion flag of every node along each route:

  - *XY route congestion level* = 6 (three highly congested nodes)
  - *YX route congestion level* = 5 (two highly congested nodes and one congested node)

  ⇒ Hereby, YX route is favoured and packets are transmitted to node 9 (a fair comparison takes place between six nodes in the two possible directions leading to the destination node 21).

- At node 9 (Figure 4.10b):
  - *XY route congestion level* = 8
  - *YX route congestion level* = 5

  ⇒ YX route is selected (node 14)
- At node 14 (Figure 4.10c):
  - *XY route congestion level* = 7
  - *YX route congestion level* = 5

  ⇒ YX route is selected (node 19)
- At node 19 (Figure 4.10d):
  - *XY route congestion level* = 2
  - *YX route congestion level* = 5

  ⇒ XY route is selected (node 18)
- At node 18 (Figure 4.10e):
  - *XY route congestion level* = 2
  - *YX route congestion level* = 3

  ⇒ XY route is selected (node 17)
- At node 17 (Figure 4.10f):
  - *XY route congestion level* = 2
  - *YX route congestion level* = 1

  ⇒ YX route is selected (node 22)

Because congestion is investigated in parts of the network close to the destination node, the highly congested area encompassing nodes 6, 7, 11, 12 and 16 is avoided altogether. Besides, a fair comparison always takes place at each hop between any pair of source and destination nodes. By considering only nodes residing at the minimal path, the problem of irrelevant excessive information is also mitigated.

4.3.4 Performance evaluation

To evaluate the efficiency of the proposed routing algorithm, we modified Noxim, a cycle accurate NoC simulator developed in SystemC [102].

The simulator was warmed up for 1000 cycles and then launched for 20000 cycles. For the sake of a fair comparison, the four algorithms, including DBAR, RCA and the two versions of the proposed scheme FACARSV1 and FACARSV2, were all implemented with a minimal fully adaptive routing function.

As performance metrics, throughput and average latency are considered. As congestion metrics, the free buffer slots is used. Originally, DBAR uses the number of free virtual channels instead, but they both achieve similar performance [96].

To test the limits of the proposed scheme, three traffic patterns are applied: hotspot, transpose and bit-reversal.

In the hotspot traffic pattern, more packets are destined to a specific set of nodes in the network (hotspots), with higher than average probability (20% in our case). With bit-reversal, each node has exactly one destination and each node receives packets from exactly one source node. The destination node's id is the reversed id of the source node (e.g. source node with id = 6 (110) communicates with destination node with id = 3 (011)). Under the transpose traffic pattern, each node sends packets to another node with the address of the reversed dimension index: node (i, j) sends packets to node (Nx - i, Ny - j); Nx and Ny are the NoC dimensions in X and Y directions, respectively.

4.3.5 Experimental results

Two sets of experiments were carried out, in which a mesh topology was considered with wormhole switching [100] [103].

Packet size was set to 8 flits with a buffer depth of 4 flits and an 8 × 8 mesh in the first set, whereas in the second set, packets had a variable size ranging between 2 and 12 flits and a buffer depth of 8 flits in a 16 × 16 mesh network. The full system configuration is outlined in Table 4.4.

Table 4.4: System configuration

Topology	Mesh	
Network size	8×8	16×16
Packet size	8 flits	2 to 12 flits
Buffer depth	4 flits	8 flits
Simulation time	20000 cycles	
Warm-up time	1000 cycles	
Traffic patterns	Random hotspot, Transpose, Bit-reversal	
Routing function	Fully adaptive	
Switching technique	Wormhole	

Latency

Under both, the hotspot traffic, with the four central nodes: 27, 28, 35 and 36 designated as hotspots receiving 20% more than average traffic and the bit-reversal traffic pattern: FACARSV2 outperforms FACARSV1, DBAR and RCA respectively. Whereas, under the transpose traffic, FACARSV1 is slightly better than FACARSV2 and still surpasses both DBAR and RCA.

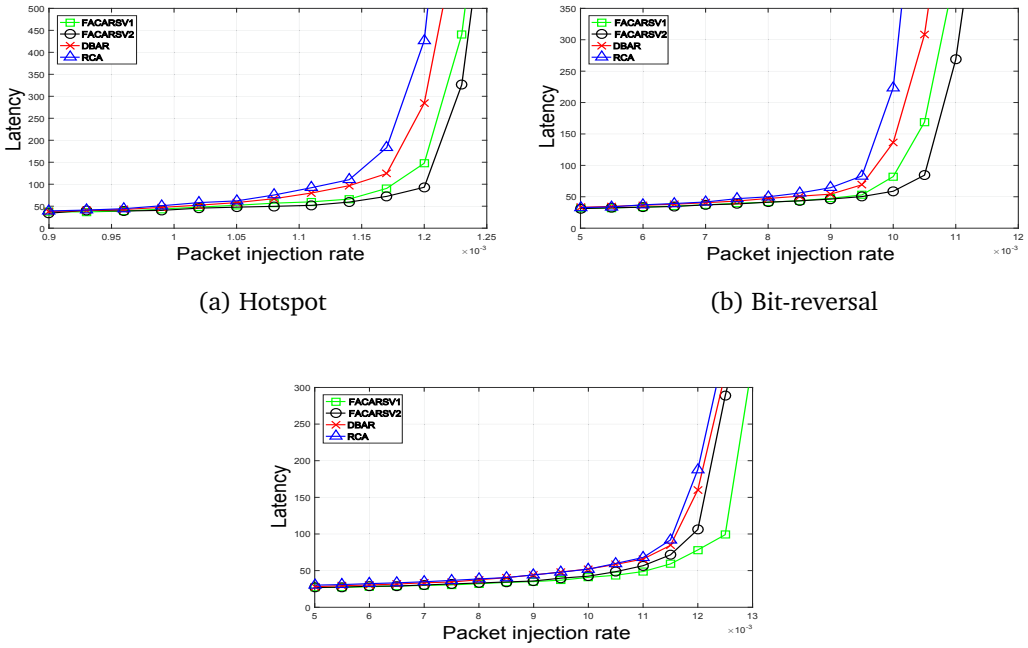
The results are approximately the same for the 16 × 16 mesh depicted in Figure 4.11 (The four central nodes 119, 120, 135 and 136 are the appointed hotspots under the hotspot traffic), an average improvement of 24.17% and 34.05% is obtained compared to DBAR and RCA respectively.

Table 4.5 outlines the average latency measured for each traffic pattern at an injection load right below saturation, as well as FACARSV2's improvement rate in contrast with DBAR and RCA.

Throughput

While FACARSV2 reaches a higher saturation point than FACARSV1 for the hotspot and bit-reversal traffic patterns, FACARSV1 exhibits an enhanced throughput under the transpose traffic and both algorithms outperform DBAR and RCA for the 8 × 8 and the 16 × 16 networks (Figure 4.12).

The improvement rate of FACARSV2 compared to DBAR and RCA in terms of packet injection rate at the saturation point is depicted in Table 4.6.



(a) Hotspot

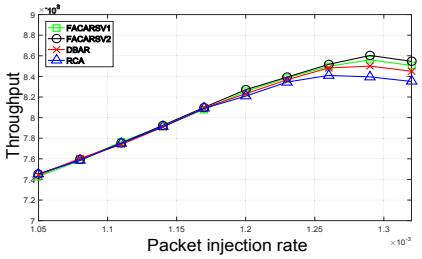
(b) Bit-reversal

(c) Tranpose

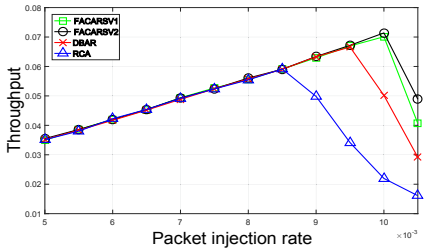
Figure 4.11: FACARS average latency

Table 4.5: Average latency improvement

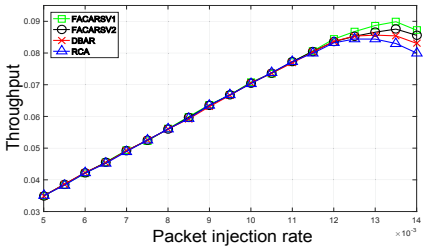
Traffic pattern	Injection rate	Average latency (cycles)			Improvement rate	
		DBAR	RCA	FACARS	vs. DBAR	vs. RCA
Hotspot	0.00111	80.0536	92.1377	51.9585	30.09%	40.60%
Transpose	0.0115	84.4309	91.8317	71.5332	15.27%	22.10%
Bit-reversal	0.0095	69.081	83.1202	50.3053	27.17%	39.47%
			Average improvement		24.17%	34.05%



(a) Hotspot



(b) Bit-reversal



(c) Transpose

Figure 4.12: FACARS average throughput

FACARSV2 performs better than both algorithms by an average of 5.21% and 11.67% respectively in the 16 × 16 network.

Table 4.6: Sturation injection rate improvement

Traffic pattern	**Saturation injection rate**			**Improvement rate**	
	DBAR	RCA	FACARS	vs. DBAR	vs. RCA
Hotspot	0.00126	0.00123	0.00129	2.38%	4.88%
Transpose	0.0125	0.012	0.0135	8%	12.5%
Bit-reversal	0.0095	0. 0085	0.01	5.26%	17.65%
		Average improvement		5.21%	11.67%

4.4 Conclusion

In this chapter, we presented our proposed novel fully adaptive and congestion aware routing schemes for mesh-based network on chip.

The schemes harvest congestion information from areas close to both the source and destination nodes and ensure fairness along the entire way of data packets.

Two variants of each algorithm that differ in terms of weight distribution and node categorization were also presented. The performance evaluation revealed that they exhibit better results compared to state of the art on chip routing algorithms.

To further optimize the designs and as future work, we are looking forward to enhancing the congestion propagation scheme, by reducing the size of the congestion map.

In addition, we are exploring the construction of their 3D extensions and the inclusion of fault tolerance, which is also deemed an important feature of reliable routing algorithms.

Chapter 5

Conclusion

5.1 Summary

With its ability to overcome the various communication challenges, Network on Chip is in effect the infrastructure of future System on Chip designs.

Although it alleviates some of the major deficiencies of traditional communication infrastructures, it is faced with new challenges.

With the ability to include hundreds of communicating cores, the design of reliable routing algorithms became of paramount importance.

As routing algorithms are in charge of forwarding data packets and represent the backbone of the network communication fabric, they face many challenges of their own. Congestion happens to be one of the most disturbing ones, which contribute to the obstruction of data packets and the undermining of the routing algorithm's reliability.

Designing routing algorithms, capable of adapting to the ever-changing congestion conditions of the network, happens to be the main contribution of this thesis.

We explored the design of various reliable fully adaptive and congestion-aware routing algorithms in mesh-based Network on Chip, as well as their accompanying congestion propagation network. The algorithms exclude irrelevant global congestion from the decision-making process and guarantee fairness. The various experimental results showcased the proposed algorithms' effectiveness

over state of the art globally aware routing schemes.

5.2 Future work

To optimize the proposed schemes and as future work, we are looking forward to enhancing the congestion propagation network, by reducing the size of the congestion map. Moreover, we will explore the possibility of testing the efficiency of the proposed techniques in terms of area overhead and power consumption, by applying real application-specific traffic.

Additionally, we will investigate the potential application of machine and reinforcement learning in the context of on-chip routing.

Furthermore, we are exploring the inclusion of fault tolerance and the development of reliable routing algorithms for 3D-based networks on chip.

List of publications

Journal publications

- Habib Chawki Touati and Fateh Boutekkouk. "Reliable Weighted Globally Congestion Aware Routing for Network on Chip." *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)* 11, no. 3 (2020): 48-66.
- Habib Chawki Touati and Fateh Boutekkouk. "Reliable routing schemes in 3D network on chip." *International Journal of Embedded Systems* 12, no. 1 (2020): 39-61.

International conference publications

- Habib Chawki Touati and Fateh Boutekkouk. "FACARS: A novel fully adaptive congestion aware routing scheme for network on chip." In *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1-6. IEEE, 2018.
- Habib Chawki Touati and Fateh Boutekkouk. "A weighted minimal fully adaptive congestion aware routing algorithm for Network on Chip." In *2017 First International Conference on Embedded & Distributed Systems (EDiS)*, pp. 1-5. IEEE, 2017.

Bibliography

[1] S. Borkar, "Thousand core chips: a technology perspective," in *Proceedings of the 44th annual design automation conference*, 2007, pp. 746–749.

[2] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: survey of current and emerging trends," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2013, pp. 1–10.

[3] S. Kobbe, L. Bauer, D. Lohmann, W. Schröder-Preikschat, and J. Henkel, "Distrm: distributed resource management for on-chip many-core systems," in *2011 Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*. IEEE, 2011, pp. 119–128.

[4] M. A. Al Faruque, R. Krist, and J. Henkel, "Adam: run-time agent-based distributed application mapping for on-chip communication," in *2008 45th ACM/IEEE Design Automation Conference*. IEEE, 2008, pp. 760–765.

[5] W. Wolf, A. A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (mpsoc) technology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, pp. 1701–1713, 2008.

[6] A. Jerraya and W. Wolf, *Multiprocessor systems-on-chips*. Elsevier, 2004.

[7] A. B. Abdallah, *Advanced multicore Systems-On-Chip*. Springer, 2017.

[8] D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low power methodology manual: for system-on-chip design*. Springer Science & Business Media, 2007.

[9] S. Pasricha and N. Dutt, *On-chip communication architectures: system on chip interconnect*. Morgan Kaufmann, 2010.

[10] E. Salminen, V. Lahtinen, K. Kuusilinna, and T. Hamalainen, "Overview of bus-based system-on-chip interconnections," in *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No. 02CH37353)*, vol. 2. IEEE, 2002, pp. II–II.

[11] M. Mitić and M. Stojčev, "A survey of three system-on-chip buses: Amba, coreconnect and wishbone," in *Proc. 41st Int. Conf. Inform. Commun. Energy Syst. Technol.(ICEST)*. Citeseer, 2006, pp. 282–285.

[12] J. Henkel, W. Wolf, and S. Chakradhar, "On-chip networks: A scalable, communication-centric embedded system design paradigm," in *17th International Conference on VLSI Design. Proceedings.* IEEE, 2004, pp. 845–851.

[13] L. Benini, G. De Micheli, and T. Tao Ye, "Networks on chips," in *Low-Power Electronics Design*. CRC Press, 2004, pp. 30–1.

[14] G. De Micheli, "Networks on chips," in *Design, Automation, and Test in Europe*. Springer, 2008, pp. 105–110.

[15] A. Agarwal, C. Iskander, and R. Shankar, "Survey of network on chip (noc) architectures & contributions," *Journal of engineering, Computing and Architecture*, vol. 3, no. 1, pp. 21–27, 2009.

[16] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, pp. 1–es, 2006.

[17] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design

methodology," in *Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*. IEEE, 2002, pp. 117–124.

[18] G. De Micheli, C. Seiculescu, S. Murali, L. Benini, F. Angiolini, and A. Pullini, "Networks on chips: From research to products," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*. IEEE, 2010, pp. 300–305.

[19] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Proceedings. 42nd Design Automation Conference, 2005.* IEEE, 2005, pp. 559–564.

[20] J. Kim, C. Nicopoulos, D. Park, V. Narayanan, M. S. Yousif, and C. R. Das, "A gracefully degrading and energy-efficient modular router architecture for on-chip networks," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 2, pp. 4–15, 2006.

[21] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 150–161, 2007.

[22] V. Rantala, T. Lehtonen, and J. Plosila, *Network on chip routing algorithms*. Citeseer, 2006.

[23] E. I. Moreno, C. A. Marcon, N. V. Calazans, and F. G. Moraes, "Arbitration and routing impact on noc design," in *Rapid System Prototyping (RSP), 2011 22nd IEEE International Symposium on*. IEEE, 2011, pp. 193–198.

[24] M. Palesi and M. Daneshtalab, *Routing algorithms in networks-on-chip*. Springer, 2014.

[25] P. Magarshack and P. G. Paulin, "System-on-chip beyond the nanometer wall," in *Design Automation Conference, 2003. Proceedings*. IEEE, 2003, pp. 419–424.

[26] N. E. Jerger, T. Krishna, and L.-S. Peh, "On-chip networks," *Synthesis Lectures on Computer Architecture*, vol. 12, no. 3, pp. 1–210, 2017.

[27] W. J. Dally, "Express cubes: Improving the performance ofk-ary n-cube interconnection networks," *IEEE Transactions on Computers*, vol. 40, no. 9, pp. 1016–1023, 1991.

[28] U. Y. Ogras and R. Marculescu, ""' it's a small world after all": Noc performance optimization via long-range link insertion," *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 14, no. 7, pp. 693–706, 2006.

[29] M. Loghi, F. Angiolini, D. Bertozzi, L. Benini, and R. Zafalon, "Analyzing on-chip communication in a mpsoc environment," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, vol. 2. IEEE, 2004, pp. 752–757.

[30] W. J. Dally and C. L. Seitz, "The torus routing chip," *Distributed computing*, vol. 1, no. 4, pp. 187–196, 1986.

[31] Dally and Seitz, "Torus routing chip," Jun. 12 1990, uS Patent 4,933,933.

[32] M. Mirza-Aghatabar, S. Koohi, S. Hessabi, and M. Pedram, "An empirical investigation of mesh and torus noc topologies under different routing algorithms and traffic models," in *10th Euromicro conference on digital system design architectures, methods and tools (DSD 2007)*. IEEE, 2007, pp. 19–26.

[33] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express cube topologies for on-chip interconnects," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. IEEE, 2009, pp. 163–174.

[34] J. Kim, J. Balfour, and W. Dally, "Flattened butterfly topology for on-chip networks," in *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. IEEE, 2007, pp. 172–182.

[35] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *Proceedings of the 34th annual international symposium on Computer architecture*, 2007, pp. 126–137.

[36] S. R. Ohring, M. Ibel, S. K. Das, and M. J. Kumar, "On generalized fat trees," in *Proceedings of 9th international parallel processing symposium*. IEEE, 1995, pp. 37–44.

[37] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang, "Optimized infinibandtm fat-tree routing for shift all-to-all communication patterns," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 2, pp. 217–231, 2010.

[38] F. Petrini and M. Vanneschi, "k-ary n-trees: High performance networks for massively parallel architectures," in *Proceedings 11th international parallel processing symposium*. IEEE, 1997, pp. 87–93.

[39] H. C. Touati and F. Boutekkouk, "Reliable routing schemes in 3d network on chip," *International Journal of Embedded Systems*, vol. 12, no. 1, pp. 39–61, 2020.

[40] Y. Qian, Z. Lu, and W. Dou, "From 2d to 3d nocs: a case study on worst-case communication performance," in *2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers*. IEEE, 2009, pp. 555–562.

[41] K. Tatas, K. Siozios, D. Soudris, and A. Jantsch, *Designing 2D and 3D network-on-chip architectures*. Springer, 2014, no. IKEEBOOK-2017-046.

[42] M. Motoyoshi, "Through-silicon via (tsv)," *Proceedings of the IEEE*, vol. 97, no. 1, pp. 43–48, 2009.

[43] F. Clermidy, F. Darve, D. Dutoit, W. Lafi, and P. Vivet, "3d embedded multi-core: Some perspectives," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 2011, pp. 1–6.

[44] R. S. Jagtap, "A methodology for early exploration of tsv interconnects in 3d stacked ics," 2011.

[45] A. Bartzas, N. Skalis, K. Siozios, and D. Soudris, "Exploration of alternative topologies for application-specific 3d networks-on-chip," in *Proc. of WASP*, vol. 5, 2007, p. 43.

[46] B. S. Feero and P. P. Pande, "Networks-on-chip in a three-dimensional environment: A performance evaluation," *IEEE Transactions on computers*, vol. 58, no. 1, pp. 32–45, 2009.

[47] V. F. Pavlidis and E. G. Friedman, "3-d topologies for networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1081–1090, 2007.

[48] A.-M. Rahmani, K. Latif, P. Liljeberg, J. Plosila, and H. Tenhunen, "Research and practices on 3d networks-on-chip architectures," in *NORCHIP, 2010*. IEEE, 2010, pp. 1–6.

[49] B. S. Feero and P. P. Pande, "Networks-on-chip in a three-dimensional environment: A performance evaluation," *IEEE Transactions on computers*, vol. 58, no. 1, pp. 32–45, 2008.

[50] U. Y. Ogras and R. Marculescu, "Prediction-based flow control for network-on-chip traffic," in *Proceedings of the 43rd annual design automation conference*, 2006, pp. 839–844.

[51] R. S. Ramanujam, V. Soteriou, B. Lin, and L.-S. Peh, "Design of a high-throughput distributed shared-buffer noc router," in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*. IEEE, 2010, pp. 69–78.

[52] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proceedings of the 36th annual international symposium on Computer architecture*, 2009, pp. 196–207.

[53] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[54] W. J. Dally *et al.*, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed systems*, vol. 3, no. 2, pp. 194–205, 1992.

[55] W. J. Dally, "Virtual-channel flow control," *ACM SIGARCH Computer Architecture News*, vol. 18, no. 2SI, pp. 60–68, 1990.

[56] A. Kumary, P. Kunduz, A. P. Singhx, L.-S. Pehy, and N. K. Jhay, "A 4.6 tbits/s 3.6 ghz single-cycle noc router with a novel switch allocator in 65nm cmos," in *2007 25th International Conference on Computer Design*. IEEE, 2007, pp. 63–70.

[57] A. B. Ahmed and A. B. Abdallah, "La-xyz: low latency, high throughput look-ahead routing algorithm for 3d network-on-chip (3d-noc) architecture," in *The 6th IEEE international symposium on embedded multicore SoCs*, 2012, pp. 167–174.

[58] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Input-output selection based router for networks-on-chip," in *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*. IEEE, 2010, pp. 92–97.

[59] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Agent-based on-chip network using efficient selection method," in *VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th International Conference on*. IEEE, 2011, pp. 284–289.

[60] W.-c. Feng and K. G. Shin, "Impact of selection functions on routing algorithm performance in multicomputer networks," in *Proceedings of the 11th international conference on Supercomputing*, 1997, pp. 132–139.

[61] H. G. Badr and S. Podar, "An optimal shortest-path routing policy for network computers with regular mesh-connected topologies," *IEEE transactions on computers*, vol. 38, no. 10, pp. 1362–1371, 1989.

[62] H. C. Touati and F. Boutekkouk, "A weighted minimal fully adaptive congestion aware routing algorithm for network on chip," in *2017 First International Conference on Embedded & Distributed Systems (EDiS)*. IEEE, 2017, pp. 1–5.

[63] P. Lotfi-Kamran, A.-M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi, "Edxy–a low cost congestion-aware routing algorithm for network-on-chips," *Journal of Systems Architecture*, vol. 56, no. 7, pp. 256–264, 2010.

[64] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, and Z. Navabi, "Barp-a dynamic routing protocol for balanced distribution of traffic in nocs," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 1408–1413.

[65] J. Hu and R. Marculescu, "Dyad: smart routing for networks-on-chip," in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 260–263.

[66] A. Charif, A. Coelho, N.-E. Zergainoh, and M. Nicolaidis, "Mini-espada: A low-cost fully adaptive routing mechanism for networks-on-chips," in *Test Symposium (LATS), 2017 18th IEEE Latin American*. IEEE, 2017, pp. 1–4.

[67] M. K. Puthal, V. Singh, M. S. Gaur, and V. Laxmi, "C-routing: An adaptive hierarchical noc routing methodology," in *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*. IEEE, 2011, pp. 392–397.

[68] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Lear–a low-weight and highly adaptive routing method for distributing congestions in on-chip networks," in *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*. IEEE, 2012, pp. 520–524.

[69] M. Ramakrishna, V. K. Kodati, P. V. Gratz, and A. Sprintson, "Gca: Global congestion awareness for load balance in networks-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 2022–2035, 2016.

[70] W. Zong, M. O. Agyemen, X. Wang, and T. Maky, "Unbiased regional congestion aware selection function for nocs," in *Proceedings of the 9th International Symposium on Networks-on-Chip*. ACM, 2015, p. 19.

[71] S. Jovanovic, C. Tanougast, S. Weber, and C. Bobda, "A new deadlock-free fault-tolerant routing algorithm for noc interconnections," in *2009 International Conference on Field Programmable Logic and Applications*. IEEE, 2009, pp. 326–331.

[72] A. Patooghy and S. G. Miremadi, "Xyx: A power & performance efficient fault-tolerant routing algorithm for network on chip," in *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*. IEEE, 2009, pp. 245–251.

[73] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 8, 2013.

[74] T. Schonwald, J. Zimmermann, O. Bringmann, and W. Rosenstiel, "Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures," in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*. IEEE, 2007, pp. 527–534.

[75] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE transactions on parallel and distributed systems*, vol. 4, no. 12, pp. 1320–1331, 1993.

[76] K. Anjan and T. M. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: Disha," in *ACM SIGARCH Computer Architecture News*. ACM, 1995, pp. 201–210.

[77] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. C. Sancho, "A survey and evaluation of topology-agnostic deterministic routing algorithms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 405–425, 2011.

[78] A. V. de Mello, L. C. Ost, F. G. Moraes, and N. L. V. Calazans, "Evaluation of routing algorithms on mesh based nocs," *PUCRS, Av. Ipiranga*, p. 22, 2004.

[79] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect," in *IEEE computer society annual symposium on VLSI*. IEEE, 2004, pp. 46–51.

[80] G. Nychis, C. Fallin, T. Moscibroda, and O. Mutlu, "Next generation on-chip networks: What kind of congestion control do we need?" in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 12.

[81] J. Jose, B. Nayak, K. Kumar, and M. Mutyam, "Debar: deflection based adaptive router with minimal buffering," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1583–1588.

[82] U. Feige and P. Raghavan, "Exact analysis of hot-potato routing," in *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*. IEEE, 1992, pp. 553–562.

[83] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," in *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, 1981, pp. 263–277.

[84] T. Nesson and S. L. Johnsson, "Romm routing on mesh and torus networks," in *Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, 1995, pp. 275–287.

[85] D. Seo, A. Ali, W.-T. Lim, and N. Rafique, "Near-optimal worst-case throughput routing for two-dimensional mesh networks," in *32nd International Symposium on Computer Architecture (ISCA'05)*. IEEE, 2005, pp. 432–443.

[86] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *ACM SIGARCH Computer Architecture News*, vol. 20, no. 2, pp. 278–287, 1992.

[87] S. Pasricha and Y. Zou, "A low overhead fault tolerant routing scheme for 3d networks-on-chip," in *Quality Electronic Design (ISQED), 2011 12th International Symposium on*. IEEE, 2011, pp. 1–8.

[88] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on parallel and distributed systems*, vol. 11, no. 7, pp. 729–738, 2000.

[89] N. Dahir, T. Mak, A. Yakovlev *et al.*, "Highly adaptive and deadlock-free routing for three-dimensional networks-on-chip," *IET Computers & Digital Techniques*, vol. 7, no. 6, pp. 255–263, 2013.

[90] M. Li, Q.-A. Zeng, and W.-B. Jone, "Dyxy: a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proceedings of the 43rd annual Design Automation Conference*. ACM, 2006, pp. 849–852.

[91] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, no. 2, pp. 62–76, 1993.

[92] M. Ebrahimi, X. Chang, M. Daneshtalab, J. Plosila, P. Liljeberg, and H. Tenhunen, "Dyxyz: Fully adaptive routing algorithm for 3d nocs," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2013, pp. 499–503.

[93] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip," *IEEE Transactions on Computers*, vol. 57, no. 6, pp. 809–820, 2008.

[94] M. Ebrahimi, "Fully adaptive routing algorithms and region-based approaches for two-dimensional and three-dimensional networks-on-chip," *IET Computers & Digital Techniques*, vol. 7, no. 6, pp. 264–273, 2013.

[95] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*. IEEE, 2008, pp. 203–214.

[96] S. Ma, N. Enright Jerger, and Z. Wang, "Dbar: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in *ACM SIGARCH Computer Architecture News*, vol. 39, no. 3. ACM, 2011, pp. 413–424.

[97] A.-M. Rahmani, A. Afzali-Kusha, and M. Pedram, "A novel synthetic traffic pattern for power/performance analysis of network-on-chips using negative exponential distribution," *Journal of Low Power Electronics*, vol. 5, no. 3, pp. 396–405, 2009.

[98] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. J. Dally, "Booksim 2.0 user's guide," *Standford University*, 2010.

[99] H. C. Touati and F. Boutekkouk, "Reliable weighted globally congestion aware routing for network on chip," *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, vol. 11, no. 3, pp. 48–66, 2020.

[100] R. V. Boppana and S. Chalasani, "A comparison of adaptive wormhole routing algorithms," *ACM SIGARCH Computer Architecture News*, vol. 21, no. 2, pp. 351–360, 1993.

[101] H. C. Touati and F. Boutekkouk, "Facars: A novel fully adaptive congestion aware routing scheme for network on chip," in *2018 7th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2018, pp. 1–6.

[102] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Cycle-accurate network on chip simulation with noxim," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 27, no. 1, p. 4, 2016.

[103] P. Mohapatra, "Wormhole routing techniques for directly connected multicomputer systems," *ACM Computing Surveys (CSUR)*, vol. 30, no. 3, pp. 374–410, 1998.