Towards Fault Adaptive Routing in Metasurface Controller Networks

Dimitrios Kouzapas[1], Constantinos Skitsas[1], Taqwa Saeed[1,2], Vassos Soteriou[3], Marios Lestas[2], Anna Philippou[1], Sergi Abadal[4], Christos Liaskos[5], Loukas Petrou[6], Julius Georgiou[6], and Andreas Pitsillides [1]

[1]Department of Computer Science, University of Cyprus, Nicosia, Cyprus
[2]Department of Electrical Engineering, Frederick University Cyprus, Nicosia, Cyprus
[3]Dept. of Electrical Eng., Computer Eng. and Informatics, Cyprus University of Technology, Limassol, Cyprus
[4]Department of Computer Architecture, Universitat Politcnica de Catalunya, Barcelona, Spain
[5]Institute of Computer Science, Foundation of Research and Technology Hellas, Heraklion, Greece
[6]Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus
{dkouza01,cskits01,annap,andreas.pitsillides}@cs.ucy.ac.cy[1], {st009698@stud.fit.eng,lm@frederick}.ac.cy[2], vassos.soteriou@cut.ac.cy[3], abadal@ac.upc.edu[4], cliaskos@ics.forth.gr[5], {petrou.loukas,julio}@ucy.ac.cy[6]

Abstract*—HyperSurfaces (HSFs) comprise structurally reconfigurable metasurfaces whose electromagnetic properties can be changed via a software interface, using an embedded miniaturized network of controllers, enabling novel capabilities in wireless communications, including 5G applications. Resource constraints associated with a hardware testbed of this breakthrough technology, currently under development, necessitate an interconnect architecture of a Network of Controllers (CN) that is distinct from, yet reminiscent to, those of conventional Network-on-Chip (NoC) architectures. To meet the purposes of our HSF testbed, we rationalize the construction of an irregular topology where its controllers are interconnected in a Manhattan-like geometry, with the flow of control directives conducted in a handshaking mode, and routing operated by an XY-YX algorithm that is agnostic of the CN connectivity, determined following the results of model specification and model checking techniques. With such controllers prone to the appearance of permanent faults, threatening the operation of such HSFs, we propose, develop and evaluate two fault adaptive routing algorithms aiming to enhance the successful delivery of packetized control directives to their recipients: (1) Loop Free Algorithm (LFA), and (2) Reliable Delivery Algorithm (RDA) of deterministic and probabilistic variants. LFA and RDA are developed based on utilizing said topology-agnostic XY-YX routing algorithm as a base, along with an appropriate adoption of routing turn rules, to address said HSF CN challenges that deviate from traditional fault tolerant routing algorithms seen in NoCs. Experimental evaluation results obtained using a custom developed simulator, show that probabilistic RDA exhibits top performance in terms of successful packet delivery ratio and topology coverage, albeit at the expense of a higher path hop count. Pointers in addressing tradeoffs between HSF CN performance and resource utilization are also provided.

I. Introduction

Hypersurfaces (HSFs) constitute a recently proposed [1] metamaterial-based paradigm, envisioned to realize a new generation of applications, as for example, programmatically-controlled wireless environments [2]. The core technology involved in HSFs is metasurfaces [3], essentially planar artificial structures comprising a periodically repeated element, the *meta-atom*, over a substrate. Metasurfaces may be engineered and then reconfigured to exhibit customized Electromagnetic (EM) characteristics fully defined by the chosen form of the meta-atom. As such, HSFs can be used to realize application-related functionalities such as perfect absorption, beam steering via anomalous reflection, or polarization control, all leading to a plethora of effective applications. For instance, the authors in [4] propose the concept of HSF-enabled wireless environments, where HSFs are integrated into walls to control the intensity and direction of reflections in mmWave communications suitable for 5G, leading to the implementation of physical-layer security, or to maximize received power in non-trivial ways for non-line-of-sight wireless communication.

Early metasurface structures were static in nature, i.e., implemented for a very specific EM functionality (e.g., absorption, beam steering, etc.) and fixed operational conditions (e.g., angle of incidence or reflection, frequency, etc.), severely limiting their scope. Such metasurfaces cannot adapt to environmental changes or host multiple said functionalities. To this end, intense research activity has been spent in developing reconfigurable metamaterials [5] to render dynamic metasurfaces. Such setup involves materials responsive to stimuli, as well as switches which can change their behavior in response to different types of signals such as thermal, optical, or even mechanical. Although dynamic metasurfaces constitute a major breakthrough, the lack of programmatic control over the functionality has motivated the introduction of the concept of software-defined metasurfaces, or HSFs for short [6].

The main underlying concept in the HSF paradigm is the introduction of an integrated Network of Miniaturized Controllers, dubbed *CN* for short, through which software directives are transformed into reconfiguration stimuli on the metasurface [6]. Such directives originate from external user input (which can be transmitted wirelessly), inserted via a gateway (that is itself interconnected to a distinct network of devices) connected directly to the edge of the CN, as Figure 3 shows. These directives are then translated into said specific

This work was partially funded by the European Union via the Horizon 2020: Future Emerging Topics call (FETOPEN), grant EU736876, project VISORSURF (http://www.visorsurf.eu) and Cyprus RPF HSadapt, COMPLEMENTARY/0916/0008.

commands, that are packetized by the gateway, to route along the CN accordingly so as to reach individual controller chips. These commands specify the state at which a unit meta-atom cell should be in, the implementation of which depends on the unit cell structure and the tuning mechanisms in place, resulting to changes in the structure of the meta-arom and thus the EM properties of the metasurface on demand. For instance, the state can refer to resistance or capacitance levels of a variable resistor or a varactor, or to the position of a given diode. In the most basic version of this HSF concept, the controllers activate or deactivate corresponding switches, that affect electromagnetic loading elements. The CN design is challenged by a number of factors [6], [7], such as the relatively small meta-atom size, the large number of nodes that conceivably need to be accommodated, the need to avoid EM emissions during configuration, and the possible appearance of faults. The above necessitate simple, low implementation cost, low power-operating, and fault-tolerant solutions [8].

A. NoCs vs. HSF CNs: Resemblance and Differences

At the system-level, there exists resemblance between on-chip network interconnected multiprocessor architectures and the HSF's CN interconnect layout, in terms of level of miniaturization and overall blueprint. As such, this clearly points us to the direction that Network-on-Chip (NoC) methodologies can be readily adopted to the HSF paradigm [5], [9], with the advantage of employing such expertise in a new application domain, i.e., the proposed HSF CN network. However, the shift in the design requirements and primarily the need for simplicity, due to scarcity in the availability of resources, suggest that although NoC methodologies can serve as kickoff points in the design procedure, customized solutions need to be developed and deployed in HSF CN structures, given that a major project objective of the same authors is the ultimate development of a prototype to realize the HSF concept [2].

Taking into consideration the design specifications outlined above, as well as additional hardware constraints, a CN architecture has been adopted [6] which possesses the following specifications, as well as differences with most traditional NoCs[1]. First the HSF CN is characterized by an asynchronous operation[2] which means that it is clockless and uses 4-way handshaking signals to send a packetized directive from one controller (the equivalent of routers in NoCs) to the next along the CN; in contrast most NoCs are clocked, albeit most of them use asynchronous-like signalling such as credits to establish correct flow-control [10], [11]. Essentially our HSF CN does not establish network-spanning flow control, as in most NoCs, as the flow of packets is carried out on a local, controller node-by-node basis. The concept of flow-control units (or flits for short) is absent here, a widespread notion in wormhole flow-controlled NoCs, as each packet acts as a unit message and is not logically split across many flits, which in turn determines our simplistic flow control protocol outlined above. The HSF controllers also comprise a single buffer vs. NoCs which contain multiple virtual channels each composed of several flit-housing buffers [11].

Next, our HSF CN possesses an irregular Manhattan-like topology [12], where alternating rows (columns) comprise links with the opposite packet flow directionality compared to their two parallel neighbors; most NoCs possess a planar fully-connected regular mesh topology [13], [11] with two opposite-directionality links among each router node pair. Following, in the HSF CN one GateWay (GW) with a single point of entry into the CN injects packetized directives to be delivered to a targeted controller(s), and subsequently a meta-atom(s); hence the HSF controllers only *consume* packetized messages, and *do not produce* any directives to be routed to any other nodes, albeit only acknowledgement messages to be received by a CN-attached ACKnowledgment GateWay (ACK GW) at the opposite topology end (see Section II-B for details) to confirm individual meta-atom EM setup; in contrast, NoC routers are each able to both consume and produce packetized messages which often contain cache-line data [11], [10].

B. Reliable Routing of Packetized Directives in HSF CNs

HSFs are complex devices that will densely integrate unit cells, tuning elements, controller chips, and on-/off-chip interconnects in non-standard packages [4]. As such, HSFs are prone to faults during fabrication, deployment, or operation. Connector misalignment, EM, wear-out, physical or intentional damage of unit cells, or even future technology miniaturization attempts, all constitute possible sources of HSF faults [8], raising high reliability concerns for such state-of-the-art systems. Aiming toward the sustainability of dependable communication among interconnected controllers in an HSF structure, the introduction of link-level fault avoidance in its underlying CN, where faulty links are bypassed by control messages using a suitably designed Fault Adaptive (FA) routing function, becomes paramount. We note that in this work we take a hard stance and assume the handling of permanent faults by our proposed routing algorithms, as detailed in Section III-B; being able to address intermittent faults is left as future work, which would lightly affect the design of our two proposed routing algorithms (see Section III.

Many Fault-Tolerant (FT) schemes target the NoC domain that our proposed HSF CN resembles. FT approaches applicable to NoCs have been inspired from macro-level interconnection networks, where Radetzkis survey [10] provides a broad coverage in the field. Essentially, as long as FT routing provides full connectivity devoid of cyclic channel dependencies in a sub-connected (i.e., faulty) topology, then the FT function crucially guarantees packet delivery devoid of deadlocks and livelocks [13]. Such deadlock-avoidance[3]

[1]We intentionally emphasize the term "most" as NoCs exhibit a plethora of different operational characteristics, and obviously not all implementations are identical.

[2]Throughout this paper the term "asynchronous" refers to a clock-less HSF control network that uses four handshaking signals between a sender-receiver controller pair, as described in Section II-C, unless otherwise stated.

[3]Throughout this paper We utilize the term "deadlock-freedom" as used in the seminal work by Duato [13]; that is, message traversals along the CN network are devoid of cyclic patterns as no channel (i.e., links in our case) dependencies are allowed, done by carefully constructing our proposed routing algorithm(s) (see Section III). As a result any messages cannot be involved in a deadlocked situation, occurring, either (1) between packetized software directives, or (2) between such directives and acknowledgement packets (ACKs), or (3) among ACK packets, all scenarios which may halt the flow of messages indefinitely and render the CN of the HSF inoperable.

mechanisms are explored by techniques such as in the Turn Model for adaptive routing [14], where certain 90 degree turns in mesh networks are prohibited so as to break the formation of said cycles and channel dependencies.

Many FT routing algorithms are build on the above principles to ensure seamless communication in NoCs, such as the FT scheme in [15] which initiates a local detour every time a faulty node is encountered. As such, new path directives are added to the header which may create a large overhead. Alternatively, the method proposed in [16] works proactively by disseminating routing data that is used to update routing tables right after a new fault is encountered, so as to bypass faulty links and routers. Another approach in [17] employs the notion of faulty blocks, where healthy links are victimized along with spatially adjacent faulty links, to design a deterministic FT method where routed packets bypass such faulty blocks. Next, the method in [18] decouples communication from computation in NoC-interconnected multicore chips, and uses stochastic communication and techniques such as branching to send multiple packets among alternative paths so as to statistically enhance the delivery of messages in networks that are marred with faulty components. Last, in an attempt to sustain good performance while avoiding deadlocks, several authors have utilized virtual channels [10]. This concept, however, is costly in terms of buffering, protocols buildup, and control resources, which cannot be afforded in the HSF CN.

C. Contributions: Fault adaptive routing in HSF CNs

In this paper, we extend our work in [19], aiming toward the development of lightweight fault adaptive routing protocols applicable to said HSF, that establish a simple, yet effective, operational mode requiring stringent network resources, as such specifications are dictated by the hardware investment restrictions imposed upon the HSF CN [7], [6]. We employ a fault adaptive methodology which attempts to deliver packets to their router node destinations in the presence of faulty network components, unlike fault tolerance methods in which packet deliveries are ensured albeit under the assumption of having bounded faulty spatial patterns in the network (i.e., by constraining both the number of faults and their distribution). Given the constraints of our topology it is very unlikely to be able to deploy classic fault-tolerant methods in our resource-constraint HSF CN, hence we direct our efforts to using fault adaptive routing that provides a best case packet delivery scenario (i.e., delivery rates equal to 100%, at best, given the density amd distribution of network component faults).

We base our fault tolerance approach on a topology-*agnostic* XY-YX routing algorithm (where oblivious XY refers to dimension ordered routing and is a routing protocol of choice for NoCs) for the HSF CN Manhattan-like topology due to its implementation simplicity (see Section III for details). Our topology agnostic XY-YX algorithm under non-fault conditions is verified to satisfy deadlock freedom [13] using model specification and model checking techniques. Specifically, a model of the HSF CN was developed using the UPPAAL model checker tool [20]. Through model checking, that exhaustively searches the state-space of the model execution, the HSF CN setting was found not to induce any deadlocks. (refer to Section III-A)

Based on said agnostic XY-YX routing, we introduce two fault adaptive routing algorithms, so as to expose different design/cost tradeoffs, dubbed as: (1) Loop-free algorithm (**LFA**), and, (2) Reliable Delivery algorithm (**RDA**) which comes in two versions: deterministic and (b) its probabilistic (see Section III-D). First, the LFA routing algorithm alternates between XY and YX routing (hence the name XY-YX), where the latter is the mirror image of XY, in order to bypass detected faulty links along routing paths, and employs a set of turn rules to maximize routing flexibility in the presence of such faults. Next, the RDA algorithm aims to achieve high reliability in routing by forming two disjoint paths between any two nodes in the network, such that when a fault occurs in the first path, the routed packetized message is forced to take an alternative path that does not overlap with the originally intended path that is found to be now faulty.

The fault-adaptiveness of the two algorithms is evaluated via simulations conducted on a custom developed simulator, with primary performance metrics considered being those of successful packet delivery ratio and the coverage ratio, i.e., the percentage of nodes nodes that can be reached via a routing algorithm. The simulation experiments also evaluate our two proposed algorithms in terms of their resource utilization, such as the average message hop count traversed by packetized messages on their way to their destination. Our results indicate that the probabilistic version of the RDA algorithm exhibits the best performance in terms of both its successful delivery ratio and network coverage, albeit at the expense of increased number of hops, while the deterministic version of RDA exhibits a good trade-off between performance and resources utilized. Finally, to mitigate the effects of very long paths which might arise, a time to live field is introduced in the message header, where design guidelines as to the suitable choice of its value is evaluated using system simulation.

The rest of the paper is organized as follows. Next, Section II discusses the considered network architecture, while in Section III the proposed routing algorithms are introduced. Section IV discusses the simulation results and finally concluding remarks and future plans are summarized in Section V.

II. HyperSurface Controller Network

A schematic overview of the HSF architecture [2] is shown in Figure I-C. It comprises a network of miniaturized controllers, each of which controls one or more metasurface switches. Depending on the switch status, different switch configurations can be dictated, which in turn change the meta-atom structure thus modifying the electromagnetic (EM) functions of the system. The input Gateway (GW), through a master-slave setup, provides connectivity between the CN and external networking devices. Although a single tile of controllers is shown in Figure I-C, provisioning is made to allow for multiple tiles, with the GW offering inter-tile communication as well.

A first full-functioning HSF prototype is targeted in [6], [12]. In the development of this prototype a three-layer Printed

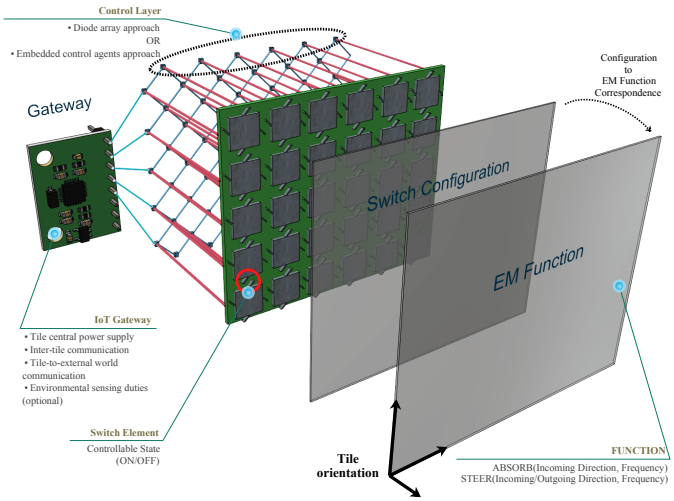
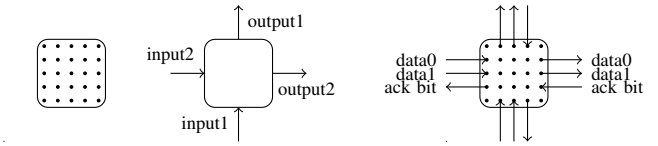


Figure 1. Decomposition of a complete HSF system: i) the metasurface layer; ii) the tile hardware layer; iii) the tile inter network Layer; and iv) the environment control layer.

Circuit Board (PCB) structure has been considered where the top layer consists of metal patches, the second layer consists of a ground plane, and the bottom layer consists of an array of Application-Specific Integrated Circuits (ASIC) in which the controller functionality is embedded. The bottom layer is connected to the top metal patch layer through vias. The role of the ASIC is to regulate the EM properties of the top layer by providing adjustable complex impedance loading as well as networking functionality. The adjustable complex impedance loading is offered by digitally-controlled varactors and varistors. A number of system requirements and constraints dictate the CN architecture, reviewed below.



(a) Chip Pins (b) Two inputs/outputs (c) Communication: Pin allocation

Figure 2. (a) controller chip underside showing pins placement, (b) single controller with its two unidirectional inputs and two unidirectional outputs, (c) CN chip pin allocation to facilitate external communication.

A. HSF Design Specifications

System design targets must be characterized by simplicity of construction, low power consumption and low implementation cost [12]. These are dictated by the need for scalability, due to the large number of meta-atoms that will be accommodated in practice, the small meta-atom size required for correct meta-material operation at small wavelengths, and the need to avoid EM interference. The meta-atom size is of critical importance to the ability of the metasurface to control EM waves. The size of the meta-atom should be comparable to the incident wave wavelength λ (in the order of $\lambda/2$), while the metasurface thickness should be much smaller than the wavelength (in the order of $\lambda/10$). Taking into consideration that at least 5 meta-atoms per wavelength are required for correct operation, to accommodate for example 60 GHz communication, sizes less than 1 mm $\times$ 1 mm are required. Such small meta-atom sizes have led to the consideration of a single chip ASIC for each meta-atom. In addition, the small meta-atom size indicates that a large number of meta-atoms and thus controllers will be involved in practical applications. This implies that the cost of each tile must be kept at a minimum and that the pursued solutions must be able to scale well with the network size. The potentially large surfaces to be covered, such as building walls, also necessitate designs of low power consumption.

EM interference avoidance with the incident waves is also of utmost importance. A large surface which is clocked can potentially radiate significant interference signals, a fact which prompts for an asynchronous digital design. An asynchronous design is also preferable in terms of scalability and cost as there will be no need for oscillators within a dense array, thus reducing space and cost. Moreover, asynchronous circuits are considered to be extremely energy-efficient. Electromagnetic interference poses constraints on the network wiring and thus the topology. Wiring should be kept at a minimum, thus favoring a grid-networked controller approach (see Section II-B).

Another significant factor to be considered is the issue tolerance to faults. The CN architecture must offer reliable data delivery even in the presence of faults, foreseen due to imminent component failure(s), external influences such as accidental/intentional damage, and loss of connectivity. It must be noted, however, that unlike in high-performing traditional NoCs, HSF applications are expected to provide somewhat relaxed performance in terms of routing latency and offered reliability levels. This implies that unsuccessful delivery of some packetized software meta-atom reconfiguration directives to the controller nodes might not be observable at the macroscopic level. Another factor to be accounted for, is that the workload that most applications are expected to incur is relatively low. Hence, the take away message is that, the CN will be characterized by asynchronous operation, simplicity of implementation, provision of fault tolerance, low power consumption, and grid-like controller interconnectivity.

B. Controller Network (CN) Topology

The current implementation technology node chosen for the controller chip has a limitation of 25 input/output signal pins [12] as shown in Figure 2-(a). The adoption of an asynchronous circuit leads to the implementation of a four-phase asynchronous handshake communication protocol [21] between two controllers. The communication protocol requires three input/output pins per controller to transmit a single bit. In accommodating these restrictions, the design chosen provides two input channel endpoints and two output channel endpoints as shown in Figure 2-(b). Such setup leads to the allocation of 12 pins for bit-by-bit communication (3 pins per channel endpoint) as shown in Figure 2-(c). The remaining 13 pins are used for configuring the meta-material and are allocated to global signals.

The interconnect geometry design of choice for the HSF CN network is a Manhattan-like topology [12] where the directionality of rows (and columns) alternates from one row (column) to the next, resembling a mesh topology where approximately half of its unidirectional links are missing. This network does not contain wraparound links that connect nodes found at the edges with their opposite topology nodes like in torus topologies; for clarification purposes, we dub such links as "end-to-end wraparound links." Figure 3 shows a 4×4 grid HSF topology, which forms a snippet of the expected 24×24 full-size HSF topology, as simulated in our experiments of Section IV-A; the full-size topology is merely a recursion of the CN interconnect shown here.

The choice of not including end-to-end wraparound links is justified by the limitations imposed by the hardware implementation; in such a scenario, the PCB would require the construction of two more layers for wraparound circuits and, moreover, controllers at the edges would require transistors with a higher signal drive to achieve the transmission of a signal through such wraparound links. Instead, in the CN topology nodes that reside on the same edge row (column) are connected between them through *edge wraparounds* as shown in Figure 3. This Manhattan-like HSF controller interconnectivity was chosen as it adheres to the constrained resource requirements of the HSF CN; at the same time it offers improved communication robustness as compared to alternative topologies such as a monotonous topology like a plain mesh or ring. This is because it offers additional connectivity via the interconnection of every pair of switch controllers at the edges of the topology using said edge wraparound links. The level of robustness attainment in the face of faulty links in our proposed HSF CN topology is further discussed in Section III.

The CN is configured via two GWs, as shown in Figure 3; the left-side GW is also responsible for packetized software directive generation. GWs are "smart" devices that connect the CN to external world communication with the possible addition of environmental sensory capabilities. The input GW forms a single point of entry into the CN and injects packetized directives to be delivered to targeted controllers, and subsequently meta-atoms, while the Acknowledgement GW (i.e., ACK-GW), connects to the opposite CN site so as to receive acknowledgements from individual controllers to confirm individual meta-atom EM setup. HSF controller addressing is implemented using Cartesian coordinates that directives use to navigate along the CN topology, as again depicted in Figure 3, where (x,y) pair values are incremented starting from the bottom left corner of the CN topology. Even rows (columns) possess an even y-coordinate (x-coordinate); the remaining rows and columns are labeled "odd."

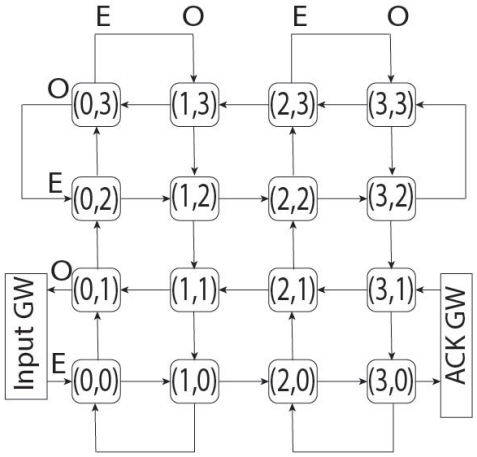


Figure 3. Manhattan-like HSF controller network with edge wrap-around links. The input gateway is connected at the south west corner of the CN, while the ACK gateway connects at the south east corner of the CN. The symbols "E" and "O" respectively denote even and odd rows or columns.

C. Inter-HSF Communicaton Signaling

The authors in [6] justify the reasoning behind choosing asynchronous circuits over synchronous ones for deployment in metasurfaces. First, while in synchronous systems components depend on clock events to exchange information, asynchronous systems rely on on-demand events, a setup which saves energy dissipated vs. observed in regular clock cycles. In addition, maintaining reliable global clock cycles becomes challenging in large surfaces, such as metasurfaces. Also, the absence of clocking saves space and is cost efficient. Instead, asynchronous circuits employ handshaking to manage data flow. In the HSF controller network, asynchronous communication among nodes is carried out through the exchange of three signals between a transmitter and a receiver.

Four-phase Handshake Communication: Figure 2-(c) depicts the three pins in each input and output port which accommodate the four-way handshake signals carried out betwen two HSF controllers (nodes), namely Data0, Data1 and ACK. First, the sender node sets one of the two data signals according to the transmission bit; Data0 signal is set if the bit to be sent is 0, otherwise Data1 signal is set if the intended transmission bit is 1. When the receiver receives this data signal it sets the ACK signal. The Sender then resets its Data signal, followed by a reset of the ACK signal by the receiver, at which point both the sender and the receiver are ready to pass the next bit.

The justification for connecting the input gateway at the [bottom left] network corner is to enable the intended deadlock-free property of our proposed XY-YX routing algorithm. After exhaustive analysis (see Section III-A) we found

out that in case the ACK gateway was connected elsewhere at the left CN boundary, then routing would become prone to deadlocks. For a complete analysis of XY deadlock freedom the reader is urged to refer to Section III-A.

III. Proposed Routing Protocols for HSF CN

As stated, most existing FT algorithms [10], [15], [22] are applicable to NoCs that provide resources and capabilities that are not available in the HSF CN, for instance, full-duplex links and ample bit-width for embedding routing data in the packet header, useful in storing info to form paths that bypass faults [15]. Taking into account the resource constraints exhibited in the HSF CN, Dimension-Order Routing (*DOR*; or "*XY*") algorithm that enjoys widespread use in 2-D mesh NoCs [23], [14] due to its inherent deadlock-avoidance property and implementation simplicity as it demands minimum route computation overheads, was chosen as the foundation protocol in developing deadlock-free routing algorithms for fully-connected (i.e., healthy) CN topology, and subsequently fault-adaptive routing functions for the same HSF CN.

DOR used "as is" would not operate correctly at all in the Manhattan-like HSF CN topology, for a simple reason: referring to Figure 4, strict XY cannot support delivery of packetized directives to controllers placed onto odd-labeled columns above the row of controllers located at the bottom-most even row despite traversing an *entirely healthy* topology, due to the reverse directionality of links residing in those odd columns. This leads us to seek XY variant routing algorithms in order to adhere correctly to the particularity of the HSF CN topology, and to later develop combined XY-XY routing to also account for the presence of possible faulty links that reduces the original path diversity offered by the base fully-connected (i.e., healthy) topology even further.

Figure 4 demonstrates two such XY variant routing algorithms: i) oblivious to the topology's interconnectivity and faults XY-YX routing that continuously alternates between routing horizontally and vertically, respectively across rows and columns; and ii) our proposed deadlock-free XY-YX variant that is odd/even column/row agnostic. We dub the first "XY-YX oblivious" and the second "XY-YX agnostic."

To explain how the two above XY-YX algorithms work, we make use of two demonstration examples as follows. We assume the presence of the HSF CN topology shown in Figure 4 where all packetized directives are inserted from the input GW at coordinate (x=0,y=0), with a node destination residing at (3,4). We begin with oblivious XY-YX where due to its topology non-agnostic nature the packet traverses the horizontal dimension rightwards (east) until node (3,0) where it discovers that it cannot go north and hence hops one more link to the right to reach (4,0) where it eventually again traverses north in an oblivious manner to reach node (4,4); we note that at (3,0) the packet could make a 180° turn, traverse a hop to the left (west) and then head up (north), but such a semi-cyclic scenario enhances the possibility of a deadlocked situation. At this point again it discovers that it cannot turn to the west direction to reach its destination and hence hops to the north, then west, then down (south) to finally reach its destination. Here, no full cycle is formed and hence it may be

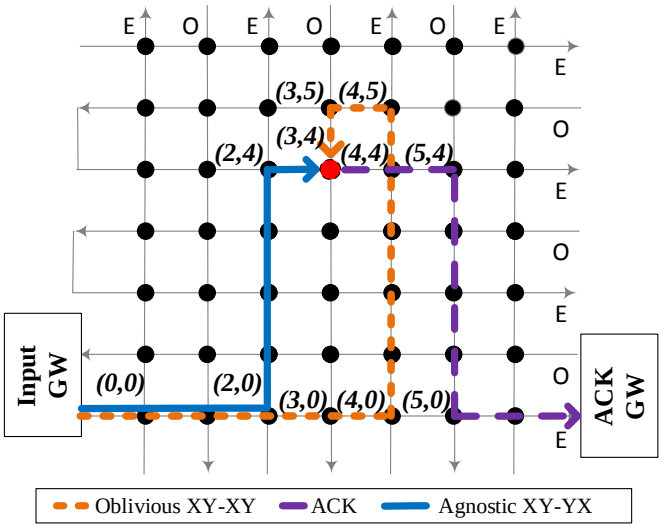


Figure 4. Paths taken by: (1) oblivious XY-YX routing (orange), (2) topology-anostic XY-YX routing (blue), and (3) ACK packet routing (purple).

considered a non-deadlocked situation; unfortunately however, the superimposition of said path with the ACK packet's path (dotted purple line in Figure 4) starting from the destination at (3,4) forms a complete 360° cycle, showing that this algorithm is prone to formations of deadlocks. This calls for a change in the XY-YX algorithm, and the solution is to alter the XY-YX algorithm to take into account the geometry of the topology, i.e., to be agnostic to it in therms of link/column-row directionality (i.e., odd/even column designation).

As such, the XY-YX agnostic algorithm makes a turn from east to north at node (2,0), instead of at node (4,0), as shown by the solid blue line in Figure 4, i.e., two hops earlier than that of the XY oblivious algorithm. The rationale for this is to allow space for maneuvering *progressively* towards the east where the destination node resides, eliminating the loop encountered earlier in XY-YX oblivious routing, which ultimately caused a full routing cycle and hence a deadlocked situation. Hence once the packet is done traversing north at node (2,4), it can now just hop to the east and reach its destination; here, there is no superimposed path with the ACK packet's route, and hence no deadlock arises, unlike in XY-YX oblivious routing. The rule formed here is to use the even CN column, one hop before the destination's odd X coordinate, to head north so as to eliminate any future deadlock. Hence considering all such routing scenarios, depending on where the destination resides and the column/row odd/even location, a complete yet uncomplicated XY-YX routing algorithm which is deadlock free is proposed and built for the HSF CN as detailed next.

A. Evaluation of Agnostic XY-XY Routing for the HSF CN: Deadlock Freedom Property

As mentioned in Section II-B, due to severe resource constraints, the interconnect geometry design of choice for the HSF CN network is a Manhattan-like topology [12] where the directionality of rows (and columns) alternates from one row (column) to the next, resembling a mesh topology where approximately half of its unidirectional links are missing, as Figure 4 shows. The geometry of a network topology and

the routing protocol that it utilizes are strongly coupled and greatly affect the deadlock-free property of the entire system, especially for the routing protocol. As such, here we utilize an analytical technique based on model checking to verify the deadlock-free property of the HSF CN system.

Said model checking is an analytical technique based on exhaustive state-space search. In contrast to traditional set theoretic techniques used to define rules for, and also validate deadlock-freedom [13], model checking: i) develops a graph/state-machine model of the system; and ii) is used for checking general properties of a system, beyond the deadlock freedom property.

In particular, specification models for candidate design alternatives using the UPPALL SMC model checking toolkit were developed, with the model against the property of deadlock-freedom in the presence of multiple packets within the HSF CN was checked. At the design phase multiple options for node connectivity and gateway placement were considered in a multidimensional design space. At this phase, applying traditional set theoretic techniques to validate each of the options would have proved an endless procedure. Model checking, however, has proved to be an efficient design tool in reaching final decisions on the chosen architecture for the HSF CN.

In contrast to most simulative approaches, model checking allows for the development of a model of the design of the system with rigorous properties. The model checker then performs an exhaustive state-space exploration of the model to check multiple properties, as expressed by the user in a formal language. This can lead to strong verification results and property guaranties. Traditional simulative approaches do not support state space exploration, neither can extract rigorous and strong verification results, a fact that may lead to the presence of bugs in a system, despite being thoroughly simulated.

Strong verification results, however, come at the cost of scalability, since formal state exploration of models suffer from the state-explosion problem. In the current analysis the scalability problems are alleviated by: i) using a tool that performs statistical reasoning on the derived state-space of a formal model, and thus allowing for statistically exploring designs at a scale; ii) gradually scaling up to a network size that reaches the performance threshold of the model checker and observing the performance trend as the network scales (since the network is symmetric our experiments in the aforementioned network size is representative of any network size); and iii) presenting the results of experiments on a 10 10 network, which is a symmetric large enough chunk of the Hypersurface and, moreover, it is a network which is larger than conventional NoC mesh networks.

We encode the HSF CN topology and the deadlock-free XY-YX agnostic routing protocol in the input language of the UPPAAL SMC model checker and its subsequent evaluation. UPPAAL SMC is the statistical extension of UPPAAL, a model checker for real-time systems represented by networks of timed automata [20]. The reasons leading to the selection of this tool to carry out the formal evaluation of the protocols considered here are threefold: First, our design is associated with dense time behaviour and requirements that can be modelled as an UPPAAL time automata model. Second, UPPAAL implements statistical reasoning about properties of timed systems. Given the large state space generated by the models, statistical model checking enables the derivation of results for larger networks than if we had used standard model checking. Third, it supports basic data structures expressed in the syntax of the C programming language, thereby allowing for concise encodings of the system's features, e.g, buffers.

The modelling presented assumes the following assumptions. First, the network is taken to be a 10×10 grid. Second, in line with the intended operation of the system, the models account only for the routing of configuration sequences and not of arbitrary sequences of packets. Finally, given that nodes are identical (thus they have the same speed) and are operating very fast, we assume the presence of a global clock and that at every tick of the clock all nodes that may fire a transition will. Following the near-future manufacturing of the first prototype chip, timing measurements (in the form of time bounds for each operation) will be provided and encoded in the model in order to obtain a more precise timing analysis.

All system variants are given by the parallel composition of 100 timed automata modelling the nodes, and a timed automaton (automata, respectively) representing the gateway (gateways, respectively). The communication between the nodes is encoded by means of four-dimensional adjacency matrices of pairwise communication channels, where item $[x][y][x'][y']$ denotes the communication channel taking input from node (x, y) and outputting to node (x', y').

Figure 5 depicts the timed automaton modelling the nodes. The automaton is composed of two states (locations) and ten transitions. Initially a node is in state *idle*. On the receipt of a message from either input *in1* or *in2* (respectively, $input_1$, $input_2$ in Figure 2-(b)), the node transitions to state *Processing*. The state models the processing of the data of the control packet before the latter is routed to its destination. While in this state, a node may perform either one of the following actions: (i) if it has not reached the destination node, then it routes the packet to one of its neighbours according to the XY-YX agnostic algorithm; (ii) if it has reached the destination node, then it will create and route an acknowledgement packet to one of its neighbouring nodes towards the ACK gateway (GW). (iii) if it is equipped with buffers, then it may receive a second packet which it enqueues in its buffer. In the Figure every transition is guarded by a boolean condition determining whether or not the transition can be fired. The condition requires from the sender-receiver pair to respect the XY-YX agnostic routing scheme and from the receiver to be in a state where the packet can be queued. Further conditions guarding the transitions enable the synchronous evolution of the system. Specifically a node can perform an action only when its local clock is equal to 1; following the action, the node resets its clock; if there is no enabled action the node simply resets its clock whenever this equals 1.

The timed automata modelling the GWs are responsible for generating configuration sequences and for receiving the acknowledgements sent by the nodes. Following the topology of the network, different configurations of the GW position and

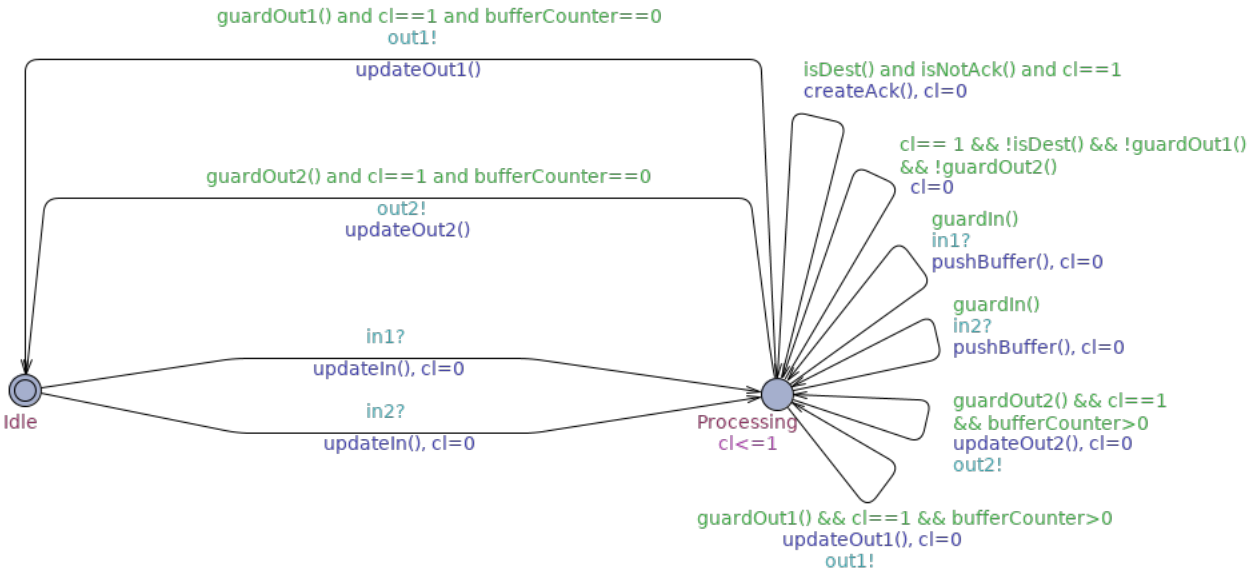


Figure 5. Timed automaton for intra-tile controller.

the orderings of the packets may induce different settings for the presence of deadlocks. We therefore consider the following configurations:

1) A GW that sends control packets towards the HSF CN is connected at the south west corner of the HSF CN and a GW that receives acknowledgement packets is connected at the south east corner of the HSF CN as depicted in Figure 3.
2) A GW that sends control packets towards the HSF CN is connected at the south west corner of the HSF CN. The same GW also acts the GW that receives acknowledgement packets, as depicted in Figure 6-(a).
3) A GW that sends control packets towards the HSF CN is connected at the south west corner of the HSF CN and a GW that receives acknowledgement packets is connected at the south east corner of the HSF CN as depicted in Figure 6-(b).

Moreover, we assume that the packets are sent row by row from south to north, and that the packets in a row are sent from west to east.

Initially, we deployed the statistical extension module of the UPPAAL toolkit that uses statistical sampling of the state space to provide results with a high degree of confidence (approx., 95%). We report the experimental results obtained by checking the system variants against specifications pertaining to deadlock-freedom, using the UPPAAL SMC query:

$$\phi_{\mathsf{ack}} \triangleq \mathsf{E}[\leq 300; 1000](\mathsf{max} : \mathsf{acks})$$

Above, acks is a variable representing the number of acknowledgements that have been received. ϕ_{ack} gives the expected maximum value of acks that are calculated on the first 300 time units, where empirical evaluation showed this to be an upper bound for the completion of the protocol, and for a sample of 1000 traces. During the lifespan of the design phase, the specifications were evaluated on progressively more complicated designs.

Evidently, UPPAAL SMC revealed that the latter two configurations present deadlocks, whereas the former configuration does not present any deadlocks. In subsequent experimentation we used the UPPAAL query

$$\mathsf{A}\langle\rangle(\mathsf{acks} == \mathsf{awaitedAcks})$$

requiring that in any execution of the model all the expected acknowledgements will be received by the ACK GW. The experimentation has been conducted on the entire state-space of the configurations and has verified that the former configuration (i.e., the configuration in Figure 3) is deadlock-free. Moreover, for the latter two configurations has revealed two traces, respectively, for which the execution deadlocks.

Figure 6 demonstrates an UPPAAL-generated simulation trace showcasing a deadlock a 4×4 size HSF CN for configuration (2). Node $(0, 2)$ is trying to route a data packet to node $(1, 3)$ through node $(0, 3)$, which in turn is trying to route an acknowledgement packet to node $(0, 1)$ through node $(0, 3)$. Consequently node $(0, 2)$ is waiting on node $(0, 3)$ and node $(0, 3)$ is waiting on node $(0, 2)$, thereby creating a deadlock. Figure 6 (down) shows a part of an UPPAAL-generated simulation trace that demonstrates the deadlock in a 4×4 size HSF CN for configuration (3). The problem arises when a configuration packet is routed towards controller $(3, 1)$, as shown with red colour. The packet necessarily needs to be routed through controller $(3, 2)$, which is connected to the ACK GW. Also, in the problematic trace it happens that the configuration packet is interleaved with acknowledgement packets, as shown with green colour, that are routed towards controller $(3, 2)$. The interleaving creates an input/output dependency between controllers $(2, 1)$, $(2, 2)$, $(3, 2)$, and $(3, 1)$.

B. Building a Fault-Tolerant XY-YX Agnostic Routing Protocol for Fault-Prone HSF Control Networks

The above XY-YX agnostic routing protocol is designed for a fully-connected HSF CN topology, and in case faulty network components are encountered it cannot offer path adaptability in bypassing such faults; as such it is not fault tolerant. Meanwhile, traditional fault-/congestion-tolerant variations of the XY-YX algorithm proposed in prior-art [24], [25], [26],

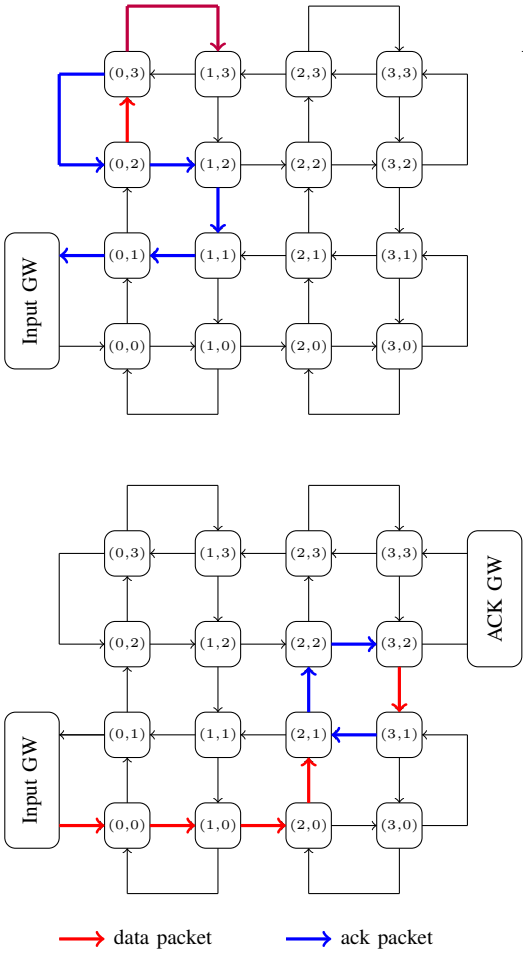


Figure 6. Deadlocked scenarios: (a) Up: paths forming a loop at the south west acknowledgement gateway, (b) Down: paths forming a loop at the north east acknowledgement gateway.

[27] are not suitable for the HSF CN because of requirements such as supporting a large packet header, duplex channels, and redundant messages, among others. While deadlock freedom is essential for well-conceived fault tolerant routing algorithms, the design of a fault-tolerant deadlock-free routing technique for the considered topology is quite challenging; especially in the presence of the aforementioned constraints. Next, the HSF applications are expected to produce moderate to low traffic rates, which can reduce the probability of induced deadlocks dramatically, as packet paths may not overlapped causing superimposed cyclic dependencies. Therefore, at this stage, we focus on the development of simple fault-adaptive routing schemes based on the agnostic XY-YX mechanism introduced earlier.

Our developed goals aim toward avoiding livelocks (i.e., endless packet cyclic routes without packets eventually being delivered to their destinations) and reliable data delivery. Our fault-adaptive routing techniques make use of the orientation and coordinates of both destination nodes and faulty nodes to provide fault adaptability; as such our routing protocols are *agnostic* of both the HSF CN topology and the positions of faulty components, such as links and routers. According to the location of a network controller, there exist four different types of topological orientations: nodes that reside on an even

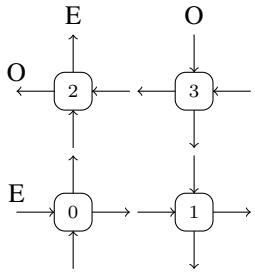


Figure 7. The four possible controller node orientations based on their odd/even row/column positioning.

row and an even column (i.e., *type* 0), nodes located on an odd row and an even column (i.e., *type* 1), nodes that reside on an even row and an odd column (*i.e., type* 2), and finally nodes located on an odd row and an odd column (i.e, *type* 3), as depicted in Figure 7.

As such, we propose two distinct fault-adaptive routing algorithms tailored for our HSF CN: Loop-Free Routing (LFA) and Reliable Delivery Algorithm (LDA). All the routing rules of the two said algorithms can be found in Appendices A and B, respectively. Each such algorithm uses a set of rules to decide the routing direction at each node. Despite these extensive sets of rules, the controller node implementation continues to be characterized by simplicity of design, low power consumption, and low hardware implementation costs. This is because the rules for route decision making are implemented by merely extending the base DOR routing algorithm, and thus operate under common principles; at the hardware level the router circuitry implements a number of conditional routing checks to establish faulty network component bypassing of routed packets, to utilize memory in terms of look-up tables, and without substantially changing the overall block architecture of the node's circuitry. Therefore, the only trade-off seen in achieving fault-adaptiveness is the fact that conditional fault-adaptive routing turn rules are more extensive vs. those required in implementing DOR routing.

In this work we take a hard stance and assume that only permanent faults are handled by our two proposed routing algorithms, although intermittent faults can also appear and subsequently be addressed [8]. For instance, intentional power gating can be considered transient, whereas physical damage can be considered permanent. Either way, we assume that the HSF CN along with its two GWs periodically checks the status of the controllers through polling, and as such detects permanent fault appearances; this is left as future work, to be implemented in the HSF CN prototype [2].

C. Loop-Free Algorithm (LFA)

As discussed earlier, despite the fact that the Turn Model for adaptive routing [14] has been adopted directly in prior-art to target routing algorithm construction in regular mesh networks [23], [17], in the case of the HSF CN under consideration such direct adoption of the Turn Model routing rules is not doable. This is because for 2-dimensional regular meshes the Turn Model assumes the existence of duplex links in all four Cartesian directions at every router, totalling eight unidirectional links for every router node. In the case of the

HSF CN however, each router supports only one direction in either the horizontal (network row) or the vertical (network column) dimensions, as it possesses half the number of links as opposed to a regular 2D mesh router; only two, perpendicular to each other, unidirectional output links exist per router.

As such, the Turn Model [14] rules cannot be effectively mapped onto such an irregular network that the HSF CN dictates, as by eliminating turns at each router in the quest of achieving deadlock-freedom in the presence of faulty network links, many of the routers will be directed to route packets along just one routing direction, a limiting fact which is guaranteed to cause a significant drop in the delivery ratios of routed packets in the presence of faulty network components. Next, oblivious XY-YX routing is, however, not adaptive to faults and is hence not fault-tolerant. Therefore, to achieve adaptivity to faults in an HSF CN, our Loop-Free routing Algorithm (LFA) combines 1) agnosticism of the topology connectivity status (i.e., the locations of faulty components and destination node) to achieve adaptability to faults by circumventing faulty links and router nodes, and 2) turn prevention in routing so as to avoid livelocks by proposing a variant scheme of our proposed XY-YX agnostic routing algorithm (see Section III-A). As such, livelock-inducing routing loops are avoided while faults are bypassed, simultaneously.

XY-YX agnostic routing employed by LFA is continuously aware of the status of the output channels[4], either healthy or faulty, at each router node so as to alternate between using XY and YX routing, accordingly. An output channel is considered to be faulty if it is defective with no packets being able to traverse it, or it leads to a non-functioning node despite the outgoing link from an operational router being non-faulty. Under LFA, a packet is initially forwarded using XY routing (i.e. horizontally then vertically) until a faulty channel is detected in its path; in such case, the packet is then directed to utilize the alternative (i.e., second, since each router contains just two output links) healthy output channel of the same router node. At this point, the header of the packet is altered to reflect upon the usage of YX routing (i.e. forwarded vertically then horizontally) to be employed starting from the next visited node along its routing path. In case another fault is later detected along the packet's path, the same swapping process is repeated. This technique is referred to as fault adaptive, FA XY-YX for short.

Simulation experiments carried out using MATLAB (details are not reported here) consistently produced considerable improvements in successful delivery rates of our proposed FA XY-YX as compared to other XY routing variants. However, blindly applying XY routing followed by YX routing prescribes a routing scheme that is prone to livelocks in the presence of faulty links. This is because the combined XY-YX algorithm can sometimes forward packets back to the same original faulty path, hence forming an infinite number of routing loops with packets never being delivered to their destinations, i.e., livelocks. Thus, to avoid such adverse phenomena while maintaining routing flexibility in the presence of faults, we utilized appropriate turn rules and adopted them to agnostic FA XY-YX routing that accounts for 1) the presence of faulty links in the vicinity of a packet's route, 2) the packet's destination node location, and, 3) the current horizontal-vertical topology orientation of the packet. We note that not all current faulty link-destination node pairs scenarios induce a livelock, hence we limit the usage of our proposed turn prevention policy to the locations where faults are expected to create loops. As such, the routing mode in which turn prevention is employed is referred to as ***abnormal*** routing, while when FA XY-YX is used is dubbed ***normal*** routing.

Our proposed routing protocols work as follows; the list of all routing rules and pseudo-code is provided in the Appendix A, and the reader is urged to refer to them for complete coverage: A packet is routed in the *normal* routing mode using agnostic XY routing until a fault is encountered, at which point (based on the horizontal-vertical Cartesian location of the fault) the packet's header is altered such that the routing algorithm is set to agnostic YX routing, and depending on the prevailing network condition, the routing mode can also be switched to *abnormal* routing mode (this is controlled by the rules in item b) in Appendix A). This abnormal routing mode restricts certain routing turns so as to break potential cycles that would cause livelocks, and delivers the packet to the next router via its input channel that was not targeted in the original routing mode. In other words, if XY routing is to deliver a packet through the first input of a certain destination node, YX routing (with the abnormal mode if necessary) now targets the second (the only alternative) input port of the same destination router. This scenario aims to route the packet around the faulty node, hence bypassing the current fault. However, there exist special cases where a fault may render the destination node physically disconnected from the network due to the spatial placement of faults in the network, in which case routing to that destination becomes impossible; for example when node (2,2) in Figure 3 is faulty, nodes (2,3), (3,2) and (3,3) automatically become disconnected. To avoid localized deadlocks and livelocks, the proposed routing scheme forbids self-looping 180° turns (i.e., going back to the same edge node where the packet had departed from using its wraparound link) at the edge wraparounds.

LFA requires the use of two bits in the packet header, one to indicate the routing technique, either XY or YX, and the other to designate the routing mode. The node prior to the location of a fault selects the appropriate routing mode which is accomplished by comparing the faulty node's location and orientation with those of the destination node. Next, having an extra bit in the header to indicate the routing mode enables the proposed method to terminate when a loop cannot be avoided. Hence, if abnormal mode is employed and the packet encounters an additional fault that forces a packet in taking a blocked turn, the routing algorithm terminates to avoid livelock induction; as such, the packet is dropped.

Walk-through Example Figure 8 shows two distinct paths taken by an equivalent number of routing algorithms to route a packet to the destination node indicated as a red circle. The destination node category is of "type 3" meaning that it lies on an odd network column and an odd network row. LFA starts

[4]The terms "link" and "channel" are used interchangeably in Section III. The same applies for the terms "router" and "node."

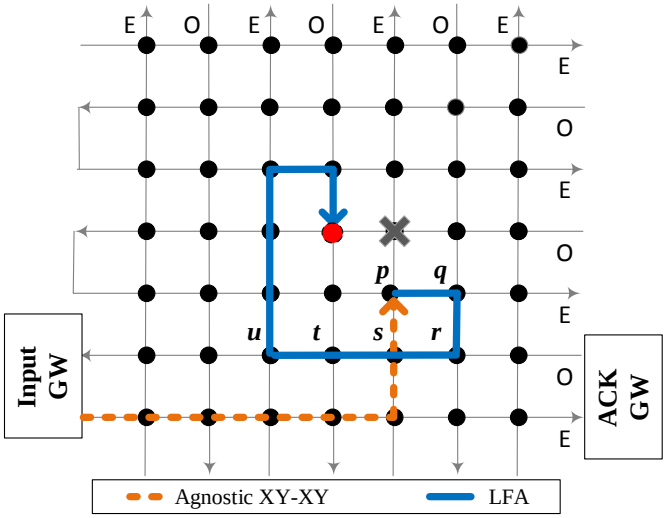


Figure 8. Routes traversed by: 1) the agnostic FA XY-YX routing shown as a dotted orange line, and 2) the LFA routing algorithm shown as a combination of said orange dotted line followed by the solid blue line, in reaching the destination node denoted by a red circle. The faulty node depicted with a cross causes XY-YX to fail, while LFA bypasses the fault a node earlier (i.e., at node p) and succeeds in its delivery.

routing the packet from the input GW using XY routing toward the destination node colored in red. When the faulty link is detected at the visited neighboring node located prior to the fault, each algorithm takes a different action to bypass it; XY routing fails to direct the packet towards its destination and hence terminates, while LFA forwards the packet to the healthy output of the current node (to the east direction), from where the packet then re-routes starting from the node labeled q. At node q, LFA switches to YX and the routing mode to abnormal (based on the rules in item b) in Appendix A). YX routing would normally forward the packet to the north direction to eliminate the spatial offset created in the vertical dimension, and then westwards (left) to eliminate the horizontal offset created. However, this leads the packet back to the same faulty node which hence creating a cycle, i.e., livelock. Therefore, in LFA the only turn allowed at node q in this case is towards east-south (i.e., to the south direction), which is found following the set of rules in item c) of Appendix A, since neither the current node or the destination are on the edge of the network (which would have required using the boundary rules in item d) in Appendix A). At node r no turns are restricted, and as such YX routing sends the packet to node s, at which point if no forbidden turns were applicable, it would direct the packet upwards (north) leading to the same cycle. Therefore, the west-north turn at node s is blocked (following the rules in item c in Appendix A), and hence from node s the packet is forwarded westwards towards node t and then u, since t routes non-progressively toward the south. From node u YX routing is used to route the packet to its final destination without requiring further turns to be blocked. As such the packet is routed to the even row which resides one row to the north of the destination node, the packet then makes a turn and travels one hop east, and finally one hop south to reach its destination.

D. Reliable Delivery Algorithm (RDA)

The Reliable Delivery Algorithm (RDA) is developed based on the principle that each source node can choose between two alternative, *disjoint paths*[5], dubbed Path_1 and Path_2, to route a packet towards a target node in the presence of faulty components. Path_1 initiates at one of the two output links (i.e., edges or channels) at the source node, and ends at one of the two input links of the destination node. Equivalently, Path_2 initiates at the alternative output channel of the source node, and terminates at the alternative input channel of the target node. Note that, in a similar manner, the oblivious XY routing algorithm along with its symmetric oblivious YX algorithm also comprise disjoint paths onto the proposed topology. An additional property of any of these route-traversing path pairs is that the first path spans the topology in a clockwise mode, while the second path spans it in a counter-clockwise mode.

Essentially, the RDA algorithm employs both the topology-agnostic XY-YX algorithm, presented as an example using the blue-colored path in Figure 4, as well as the oblivious XY-YX algorithm, presented as an example using the orange-colored path in the same figure. One of the two spans the topology in a clockwise manner, while the other acts as a symmetric counterpart and spans the topology in an anti-clockwise manner; each route accordingly follows either the routing turn rules in Table I or those in Table II of Appendix B. Having the option between alternative disjoint routing paths towards a target node increases the likelihood of delivering a packet, since path disjointnes enhances the avoidance of possible faults in one of the two paths.

RDA utilizes a single bit in the packet header to indicate whether a node will route a packet either along Path_1 or along Path_2. In the case where a fault is next encountered along its path (i.e., the next connected node in the originally spanned path possesses a fault), say Path_1, the controller node then flips said RDA packet header bit to designate that the same packet will start traversing the alternative path, say Path_2. The disjointness of the two paths implies that each path utilizes a distinct routing node output channel, and, that also each such path terminates at a unique input channel at that target node.

Using two disjoint paths to reach the destination node may result in one of the offered paths not being minimum in length. The send Gateway sets the RDA packet header bit, according to which of the two paths (Path_1 or Path_2) has a minimal length; if no faults are encountered the packet will reach the destination with the minimal number of steps.

The gist behind RDA's workings, is that it pre-targets one of the two input channels of the destination node based on both the packet header bit and its agnosticism concerning the odd/even row/column orientation of the destination node. For example, as seen in Table I in Appendix B, which provides details of Path_1, if the packet header bit is equal to logic "0" corresponding to Path_1 (computed using the rules of Table I in Appendix B) and the destination node orientation type is 0, then RDA aims for the input channel of a target node residing on the vertical CN axis (i.e., column). If a fault is detected

[5]The two paths are edge independent, and to the most part node independent, with only the first and the last nodes being common.

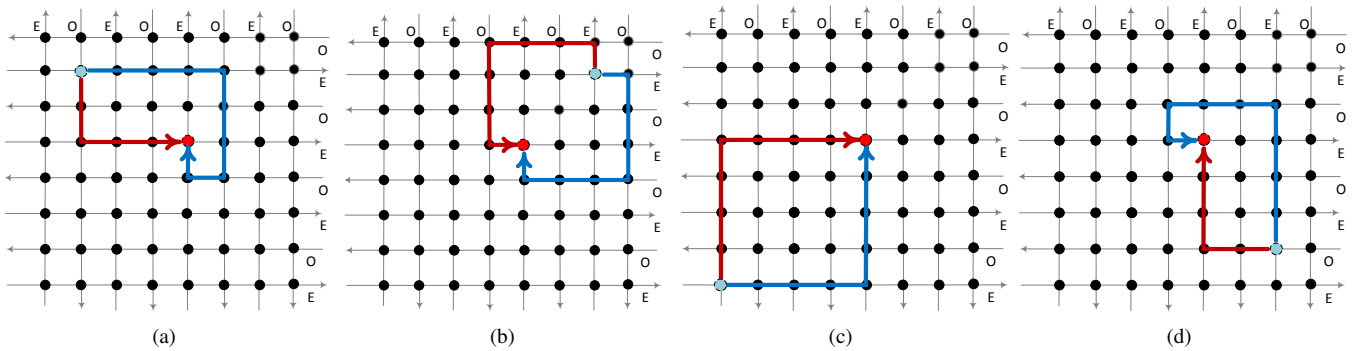


Figure 9. Routing examples for the Reliable Delivery Algorithm.

along this path, then the packet header bit is flipped to 1, and is then routed towards the alternative path that aims for the alternative input channel of the destination node that resides on the horizontal CN axis (i.e., row).

After the source node decides on path towards target, the RDA compares the target node location with the source node location to choose the appropriate topology-agnostic XY-YX algorithm to route the configuration packet. However, in some cases topology-agnostic XY-YX algorithm and its symmetric ones are not sufficient to establish two disjoint paths. In this case, the RDA proposes a set of turn rules, based on the Turn Model for adaptive routing [14], to achieve path disjointedness. The turn rules are defined in the Turn Rules sections of Table I and Table II of Appendix B. These conditions identify the cases where path disjointness cannot be achieved using topology-agnostic XY-YX algorithm and describe turn rules for forbidding turns.

We provide the following summary of RDA steps for clarity:

i. RDA works by first using its packet header bit to identify the appropriate routing rules (either Table I or Table II in Appendix B);
ii. Next, RDA checks its turn rules, and upon their satisfaction avoids the forbidden routing turn;
iii. Otherwise, on a failure of the above it checks and implements the XY routing variant conditions.

Figure 9 illustrates the disjoint paths taken by the RDA algorithm for some main routing cases. Source nodes are shown in red color while target nodes in turquoise color. Disjoint paths (coloured in red and blue, respectively) start from each source node and end at different input channels on the target node. The two alternative paths can be computed with the help of Table I and Table II in Appendix B. Table I corresponds to Path_1 when the packet header bit equals to logic "0" and Table II symmetrically corresponds to Path_2 and the RDA packet header bit being equal to logic "1."

The case shown in the diagram (c) of Figure 9 is demonstrated; other cases are similar. The routing node (type 0) has address (0, 0) (denoted as a light blue circle in the diagram) and the destination node (also of type 0) has address (4, 4) (denoted as a red circle in the diagram). According to the state of the RDA packet header bit, it decides which of the two decision Tables to use. For Path_1 the routing node (0, 0) uses the topology-agnostic XY-YX algorithms section of Table I to compare its own coordinates with the target node coordinates (4, 4) and decide on the routing algorithm suitable to reach the target from the horizontal input. In this case, condition $x \leq a$ and $y < b$ (under D=0) is true, resulting in choosing the symmetric Oblivious XY-YX algorithm that follows an anti-clockwise direction. Respectively, if the RDA packet header bit is 1 and Path_2 is enabled, the same conditions will result in using the Oblivious XY-YX routing algorithm (under D=0 in the topology-agnostic XY-YX algorithms section of Table II) and the destination will be reached from the south input using a clockwise direction to the target.

A case where turn rules need to be used is also explained. Assume that the packet is routed from node (0, 0) (type 0) towards the target node (4, 0) (also of type 0). If Path_1 is decided for routing, then the Oblivious XY-YX algorithm (Table I, under D=0, first condition) will be used. Similarly, if Path_2 is decided for routing, then also the Oblivious XY-YX algorithm will be used (Table II is used, under D=0, the first condition). For both paths the same algorithm will be used. To achieve path disjointness for Path_1 and Path_2 the Turn Rule section of Tables I (respectively, Table II) must be used. In this case, the second rule on Table I is used, which forbids routing west when the routing node is on a y-coordinate equal to 0.

A variant of the RDA is the probabilistic RDA. In the probabilistic RDA when a fault is next encountered along its path the routing node will route the packet to the alternative output but will flip the RDA packet header bit based on a probability. This implies that the routing path might not change upon encountering a fault. Intuitively, the probabilistic RDA can offer more alternative paths to reach a target node; the non-deterministic nature of the probabilistic RDA can be used to escape a livelock that would otherwise appear if the non-probabilistic RDA version was used. However, more alternative paths and livelock escape implies additional number of hops towards reaching the target node. Non-probabilistic RDA is also referred to as probabilistic RDA with probability 1, RDA-1.

As explained above, RDA rules ensure that Path_1 and Path_2 are disjoint towards the destination node. However RDA rules do not exclude the presence of livelocks, that may appear given the position of faults in the network. In the presence of a livelock the RDA requires a Time-to-Live (TTL) field in

the packet header that is reduced each time a packet makes a routing hop. If the TTL value reaches 0, then the routing node assumes that the packet cannot reach its destination and terminates the routing process. It is important to make appropriate choices for the TTL value that achieve a good trade-off between successful delivery and waste of resources. A further analysis of importance of the TTL variable is shown on the Evaluation section.

IV. Performance Evaluation

Routers in the HSF CN are not clocked and thus employ asynchronous communication. Most available NoC simulators, however, are based on scheduling and do not offer a ready-to-use clock-less communication operating mode option. As such, we built a custom-made simulator, dubbed HSF CN Asynchronous Simulator, or HCNAS for short, that employs a four-phase asynchronous handshake communication protocol (see Section II-C), which allows routers to communicate asynchronously, using the AnyLogic multimethod simulation modeling tool [28].

HCNAS relies on conditional events to achieve asynchronicity. The network is connected to two "smart" gateways which are equipped with clocks and are responsible for generating data traffic and handling ACKs, one placed at the lower left side of the network, while the second gateway is placed at the lower right side of the HSF CN as depicted in Figure 3. A router node comprises of two output ports and two input ports, which connect each router to its neighbors, as shown in Figure 3. Edge router nodes are connected to each other through edge wraparounds, with each creating a bidirectional channel between each two edge neighbors. In addition, a router node incorporates enough buffering space to receive an entire packet before forwarding it to the next neighboring router.

Faulty nodes are determined and designated at the initialization time before any packets containing data enter the network. Each router node performs a four-way hand shake to receive and transmit every bit, for a bit-by-bit serial transmission, until the whole packet is sent. Once an output port (input port at the intermediate, or neighboring, router receiver) is selected for transmission (reception at the neighboring router) it cannot be changed until the current transmission (reception at the neighboring router) of the packet is completed. Next, a router node is never in a position to receive and transmit at the same time, nor receive from or transmit to two different nodes simultaneously; hence, one operation is carried out at a time, and this is done to simplify the hardware design of each HSF CN router node.

Depending on the control data contained in the packet header, nodes locally decide the direction of the next hop. Additionally, each router node is assumed to possess local knowledge of its faulty neighbors, and thus disables its output ports that are connected to such faulty nodes. Thus, to avoid transmitting a packet towards a faulty node which would evidently cause an unacceptable transmission error, the packet is either redirected to the healthy channel (ideal scenario), or is stalled, or as a last resort, it is dropped; the choice depends on the routing algorithm being employed. If both outputs at a router node are disabled (or, blocked) packets are either stalled or dropped altogether. RDA and LFA ensure that packets will either be routed or dropped, but they can never be stalled, since stalling might result in the entire network getting stalled due to back-pressure. This is achieved by not sending packets to nodes that are healthy but have both their outputs blocked due to faulty neighbors.

A. Simulation Results

The simulation experiments aim at comparing the performance of the fault-adaptive routing algorithms proposed in Section III. The evaluation is conducted with respect to the ability of the algorithms to successfully deliver packets to their destinations in the presence of faults taking into account the resources utilized as these are quantified by the number of hops. A time to live field (TTL) is also introduced to prevent overlength paths, and simulations are conducted to provide guidelines for the selection of a suitable value of the TTL which would yield good performance. In order to evaluate the performance of the proposed routing algorithms in the presence of faults, simulation experiments are conducted for a 24×24 Manhattan-like network with edge wraparounds; this network resembles the alternating row and column directionality of the network shown in Figure 3, albeit being a 4×4-sized network. The input gateway and acknowledgement gateway are assumed to be at the bottom left corner and bottom right corner of the network, respectively. To emulate a HSF CN with faulty components, each router node is assigned a probability of failure equal to P_f, where P_f assumes values in the range $0.0 - 0.08$ in increments of 0.02; such a span of failure values establishes reasonable failure patterns so as to evaluate the performance of a HSF CN under realistic scenarios of network defects. A single packet is sent from the input GW to its destination node, while in the event of successful delivery, an acknowledgement packet is forwarded from this destination router node to the acknowledgement gateway.

The first experiment evaluates the fault-adaptiveness of the agnostic XY-YX (non fault adaptive), LFA, RDA (probability 1 and probability 0.7), and the probabilistic agnostic FA XY-YX (probability 1 and probability 0.7) fault adaptive algorithms. The non fault adaptive agnostic XY-YX is used as a measure of rating the fault adaptability of each algorithm. In order to evaluate the performance of our proposed routing algorithms and to aid the understanding of their behavior in bypassing network faults, the network is segmented into four equal-sized quadrants. A destination in each quarter of the network is selected. Each of the four destinations has a different orientation. For each destination node the experiment is repeated $5,000$ times for a total of $20,000$ repetitions, for each P_f value.

Figure 10 demonstrates the average percentage of data packets successfully delivered to their respective destination nodes when the probability of faults, P_f, increases. All algorithms, except the agnostic XY-YX, achieve a percentage of successful delivery of more than 93% for $P_f = 0.02$ with the best results achieved by the RDA-0.7 algorithm that has a 97% successful delivery ratio. As the P_f value increases to 0.08 the successful

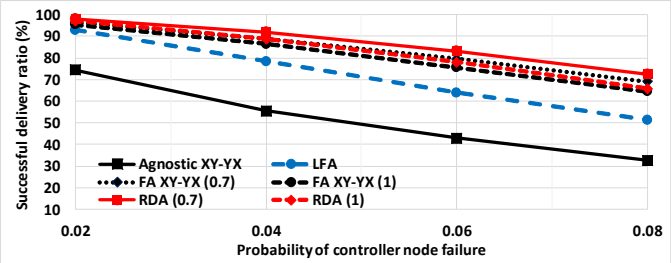


Figure 10. Percentage of packetized directives successfully delivered to their destination controller nodes vs. per controller node failure probability.

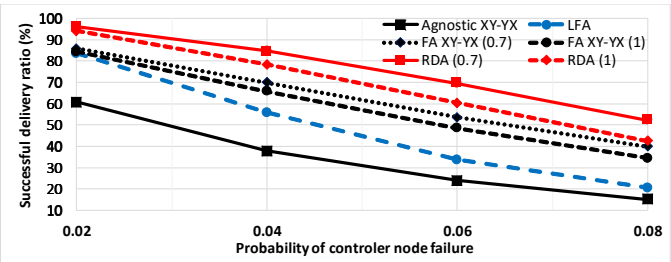


Figure 11. Percentage of ACK packets successfully delivered to the ACK gateway vs. per controller node failure probability.

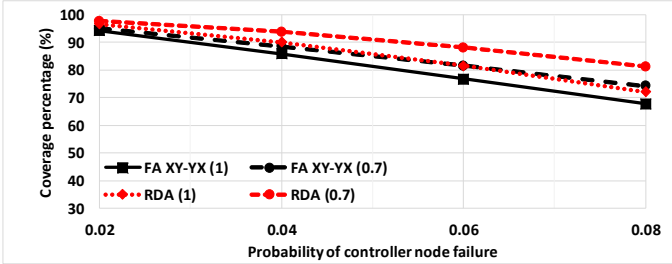


Figure 12. Coverage percentage as a function of the node failure probability: Successful delivery of packets to the target node.

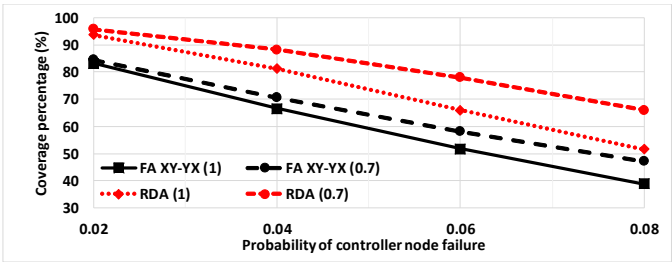


Figure 13. Coverage percentage as a function of the node failure probability: Successful delivery of acknowledgement packets to the Acknowledgement Gateway.

delivery metric drops in a linear fashion in the range 64% and 72% for the RDA variants and the probabilistic FA XY-YX cases, whereas the LFA has a lower successful packet delivery percentage of 51%. All fault adaptive algorithms deliver 18% to 40% more packets than the non fault adaptive agnostic XY-YX algorithm. Moreover, it is observed that the RDA-0.7 achieves better results overall, whereas the LFA achieves the worst successful delivery results in all cases. Interestingly, the probabilistic algorithms of RDA and FA XY-YX that have 0.7 probability to change the routing path upon encountering a fault, have higher percentage of successful delivery.

The next experiment aims at evaluating the successful delivery of the acknowledgement packets. Figure 11 shows the average percentage of successfully delivered acknowledgement packets to the acknowledgement gateway, as the probability of faults, P_f, increases. As expected, the successful delivery drops compared to the successful deliveries to destination nodes. The reason behind this is the fact that the ACK delivery rates are calculated taking into account the total number of data packets, regardless of their delivery status. We can observe, a linear drop of the delivery percentage as P_f increases. In fact, as P_f increases the delivery percentage approaches the results of the non fault adaptive agnostic XY-YX algorithm. The RDA-0.7 algorithm yields the best results, whereas the LFA algorithm yields the worst. Overall, successful delivery ratios decline when the destination is further away from the input gateway, which is reasonable as the random fault model used implies that longer paths are more likely to have faults.

Next, we evaluate the best performing algorithms of the previous set of experiments with respect to a slightly different performance metric which characterizes successful message delivery, namely the coverage percentage. The coverage is a measure of the average success probability over a number of destination nodes and is defined as the number of nodes that successfully receive a packet when a packet is sent to each node in the network. We examine the coverage achieved by RDA-1, RDA-0.7, XY-YX-1 and XY-YX-0.7, which are the best performing algorithms of the previous experiments whose results are shown in Figures 10 and 11. Specifically, the coverage percentage of these algorithms is evaluated; a packet is sent to each node that is connected to the gateway, measuring the percentage of nodes that successfully receive the packet. In practice however, the covered nodes can only be detected through the received respective acknowledgement packet. Thus, we consider the percentage of acknowledgement packets received by the gateway in order to measure the percentage of "reported" covered nodes.

Figure 12 presents the coverage percentage corresponding to the successful delivery of data packets to the destination nodes as P_f increases from 0.02 to 0.08. The results validate the fact that RDA-0.7 exhibits the best performance (as in Figure 10) and algorithm FA XY-YX-1 exhibits the poorest performance of the four algorithms. However, the coverage percentage is higher compared to the successful delivery percentage in Figure 10.

Similarly, Figure 13 presents the reported coverage through the percentage of successfully received acknowledgement packets at the acknowledgement gateway, as P_f increases from 0.02 to 0.08. Figure 13 shows a significant drop in reported coverage percentage compared to Figure 12. The results continue to validate RDA-0.7 as the best-performing and XY-YX-1 as the worst, similar to what is reported in Figure 11. Moreover, we can see that RDA-0.7 has a respectable coverage percentage of 67% for $P_f = 0.08$ in contrast to Figure 11 that reports a performance degradation of the order of 50% or lower.

RDA-0.7 and RDA-1 achieve better results due to the increased adaptability complemented by more complex decisioning and intelligence. RDA-0.7 performs better than RDA-1 in all cases, and in some cases it reports significantly better results, e.g., in Figure 13 RDA-0.7 has 65% coverage when $P_f = 0.08$, while RDA-1 has 52%, a notable difference of 12%. This fact shows the advantage of adding a factor of non-determinism to the fault adaptive algorithm.

Moreover, in order to investigate the scalability of the proposed algorithms with respect to changing the network size, we repeat the experiment for two additional sizes. We consider network sizes of 24×24, 20×20 and 14×14, denoted by Size 24, Size 20 and Size 14, respectively. In Figure 14, we plot the percentage of successful deliveries of data packets to destination node for RDA-1, RDA-0.7, FA XY-YX-1, and FA XY-YX-0.7. It can be observed that the performance of all four schemes deteriorates with the increase in the network size. However, RDA-0.7 achieves the slightest deterioration with a difference of less than 5% between the successful deliveries of Size 14 and those of Size 20 for $P_f = 0.08$. It can also be seen that as the probability of failure increases the decrease in the performance is more evident at larger sizes. The same applies for the percentage of deliveries of ACK packets, where RDA-0.7 outperforms the rest of the algorithms as shown in Figure 15. The decline in the performance reported at larger sizes can be attributed to the fact that longer paths, as a result of larger networks, are more likely to have faults than shorter paths. The independent random faults model which we use, implies that as the network size increases the absolute number of fault increases. Thus, packets may encounter more faults in larger networks which negatively affects the successful delivery percentage of the proposed routing techniques in larger networks.

In the second set of experiments we investigate the resources utilized by the different algorithms in order to realize their routing decisions, by observing the average number of hops of the routing paths. Higher number of hops is undesirable as it implies higher resource utilization which degrades performance in cases of heavy load. Our results indicate that higher coverage percentage comes at the cost of additional resources utilized by the network. This is demonstrated in Figure 16 which shows the average number of hops per packet required to successfully deliver a packet to its destination and subsequently its acknowledgement to the acknowledgement gateway, in the network coverage scenario. As expected, the most successful algorithms, e.g. RDA-0.7, require more hops to successfully deliver a packet. This is due to the fact that deterministic approaches cannot break encountered livelocks, thus delivering less packets with a limited number of hops. On the other hand, probabilistic methods are more likely to break livelocks which can increase the hops taken by a packet to achieve a successful delivery.

An interesting observation in Figure 16, is that the average number of hops per packet does not change much for algorithms RDA-1 and XY-1, i.e. the non-probabilistic versions, whereas it steadily increases for algorithms RDA-0.7 and XY-0.7, i.e. the probabilistic versions of the algorithms. This can be explained by the fact that algorithms with probability 1 create deterministic combinations of 2 paths in the case of faults, whereas algorithms with a probability lower than 1 create more alternative paths in a non-deterministic fashion. As the fault percentage increases, the need for more alternative paths becomes essential to achieve fault adaptiveness. This, however, comes at the cost of extra resources, i.e. hops.

The results in Figures 10, 11, 12 and 13 demonstrate that the probabilistic RDA-0.7 achieves higher successful packet delivery ratios and higher coverage ratios in all cases. Comparable performance is also demonstrated by RDA-1 and probabilistic LFA XY-YX-0.7. However, as indicated in Figure 16 RDA-0.7 achieves better ratios at the cost of higher number of hops.

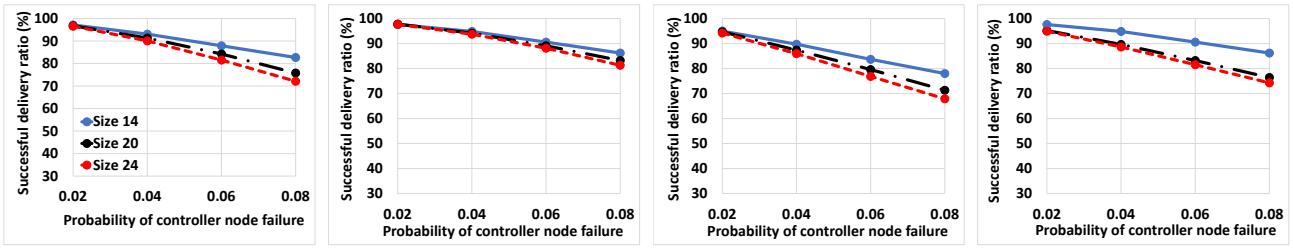
A further study on the number of hops required to achieve coverage is demonstrated in Figure 17, and Figure 18. The scenario involves packets being sent to each of the controller nodes and the graphs show the percentage of successful deliveries which have required a particular number of hops to be realized. Figure 17 shows the case of RDA-1. Figure 18 shows the case of RDA-0.7. This can be used as a guideline for the optimal choice of the TTL field so that resources are not wasted.

For RDA-1 all packets reach the target in no more than 70 hops, while RDA-0.7 requires up to 200 hops to deliver packets to the destination. Specifically, for $P_f = 0.02$, RDA-1 delivers 96.6% of the total number of packets with less than 66 hops, while RDA-0.7 delivers 97.3% of the total number of packets with less than 66 hops and 0.2% additional packets with more hops (up to the TTL = 200 hops). Therefore, for a low probability of faults RDA-0.7 outperforms the rest of the algorithms even when the TTL value is limited to 66 hops. However, for higher P_f values, e.g. $P_f = 0.08$, RDA-1 delivers 72.1% of the total number of packets with the TTL field settled to 71 hops, while RDA-0.7 delivers 78.6% of the total number of packets with the number of hops being lower than 71 and 3% more packets with a number of hops higher than 71.

The TTL value is a parameter that affects the overall performance of the RDA; setting the TTL value to a high value may result in waste of resources, i.e number of hops, without any significant gain as in the case of RDA-0.7 for $P_f = 0.02$. The results in Figures 17, and 18 can be used as a guideline to set the TTL field for the RDA variants, especially when the trade-off between performance and resource is critical.

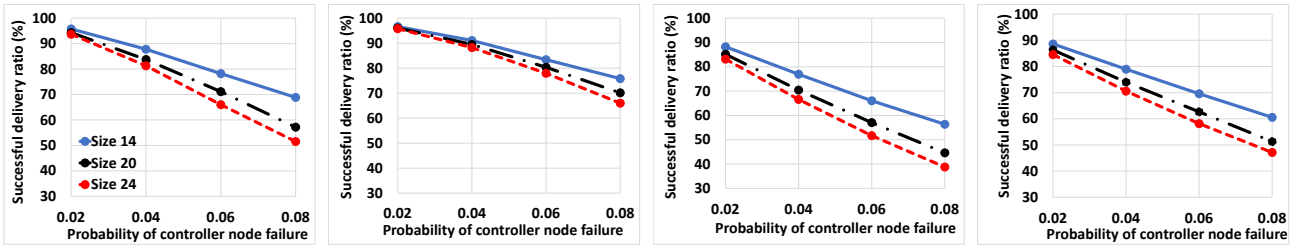
V. Conclusions

HyperSurfaces are novel planar devices that offer customizable interaction with electromagnetic waves. A core component is an embedded Network of miniaturized Controllers (CN) that resembles NoC architectures, albeit with unique restrictions in terms of available hardware resources and operational mode. The reconfigurability of the HSF meta-material relies on disseminating relevant configuration information to the controller switches which control the behavior of the meta-surface cells. With such HSF components prone to permanent faults, it is vital to employ an efficient fault-adaptive routing technique in the HSF CN so as to establish reliable delivery of directives to CN controllers to enable said reconfiguration.



(a) (b) (c) (d)

Figure 14. Successful delivery ratio of data packets of the proposed algorithms for different sizes of the network, namely, 14×14 (solid blue), 20×20 (dotted dashed), and 24×24 (dashed) : (a) RDA (1) (b) RDA (0.7) (c) XY-YX (1) (d) XY-YX (0.7).



(a) (b) (c) (d)

Figure 15. Successful delivery ratio of ACK packets of the proposed algorithms for different sizes of the network, namely, 14×14 (solid blue), 20×20 (dotted dashed), and 24×24 (dashed): (a) RDA (1) (b) RDA (0.7) (c) XY-YX (1) (d) XY-YX (0.7).

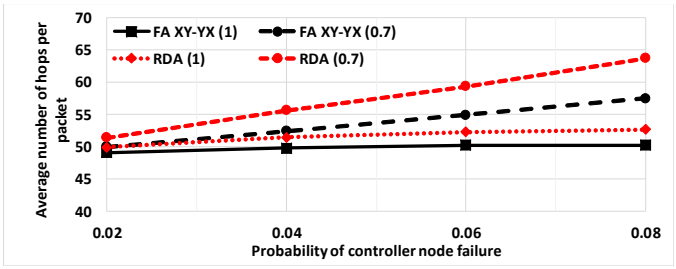


Figure 16. Average number of hops for successfully delivered packets as a function of the controller failure probability.

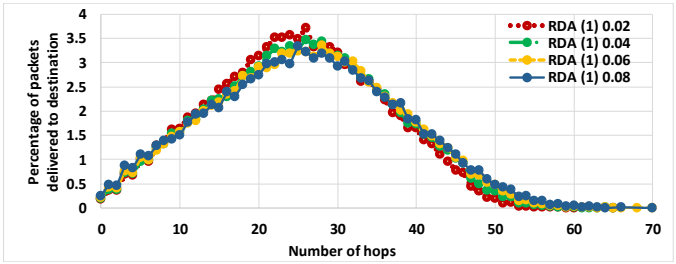


Figure 17. Percentage of successfully packet deliveries corresponding to a particular number of hops.

As such, in this work we proposed HyperSurface-specific fault adaptive techniques for said CN, based on XY-YX routing algorithm variants and an appropriate adoption of routing turn rules. The effectiveness of the proposed schemes was demonstrated through extensive HSF CN simulations, including directives successful delivery ratios as well as a network scalatbility analysis.

Future goals include the development of a fault-tolerant

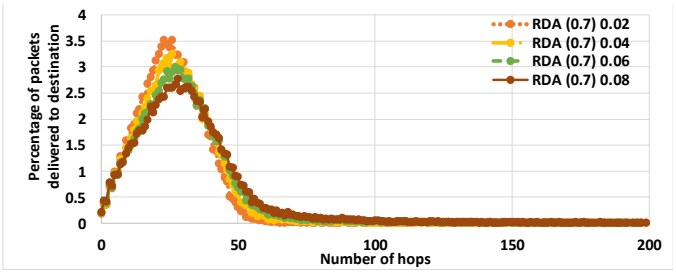


Figure 18. Percentage of successfully packet deliveries corresponding to a particular number of hops.

routing algorithm which in tandem ensures freedom from deadlocks and livelocks. Possible techniques may involve node victimization to ensure well-formed faulty regions so as to restrict turns and hence eliminate any cyclic dependencies that are precursor to undesirable deadlocked routing, as well as time-division multiplexing techniques to avoid path superimposition that would produce such routing cycles. Further future works in the field will also focus on developing validated fault models as HSF prototypes become available, which will help in better determining the impact of the fault-tolerant routing algorithms. Finally, power-gating mechanisms will be also developed aiming to leverage fault tolerance provided by the network in the pursuit of very low power consumption.

REFERENCES

[1] Christos Liaskos, Ageliki Tsioliaridou, Andreas Pitsillides, Ian F Akyildiz, Nikolaos V Kantartzis, Antonios X Lalas, Xenofontas Dimitropoulos, Sotiris Ioannidis, Maria Kafesaki, and CM Soukoulis. Design and development of software defined metamaterials for nanonetworks. *IEEE Circuits and Systems Magazine*, 15(4):12–25, 2015.

[2] C. Liaskos, S. Nie, A.i Tsioliaridou, A. Pitsillides, S. Ioannidis, and I. Akyildiz. A new wireless communication paradigm through software-controlled metasurfaces. *IEEE Communications Magazine*, 56(9):162–169, 2018.

[3] H. Yang, X. Cao, F. Yang, et al. A programmable metasurface with dynamic polarization, scattering and focusing control. *Scientific reports*, 6:35692, 2016.

[4] Christos Liaskos, Shuai Nie, Ageliki Tsioliaridou, Andreas Pitsillides, Sotiris Ioannidis, and Ian Akyildiz. Realizing wireless communication through software-defined hypersurface environments. IEEE, 2018.

[5] S. Abadal, C. Liaskos, A. Tsioliaridou, et al. Computing and communications for the software-defined metamaterial paradigm: A context analysis. *IEEE access*, 5:6225–6235, 2017.

[6] L. Petrou, P. Karousios, and J. Georgiou. Asynchronous circuits as an enabler of scalable and programmable metasurfaces. In *proceedings of the ISCAS*, pages 1–5. IEEE, 2018.

[7] P. Kouvaros, D. Kouzapas, A. Philippou, et al. Formal verification of a programmable hypersurface - work in progress. In *Proceedings of the 23rd FMICS*, pages 83–97, 2018.

[8] H. Taghvaee, S. Abadal, J. Georgiou, A. Cabellos-Aparicio, and E. Alarcon. Fault tolerance in programmable metasurfaces: The beam steering case. In *Proceedings of the ISCAS '19*, pages 1–5, 2019.

[9] S. R. Vangal, J. Howard, G. Ruhl, et al. An 80-tile sub-100-w teraflops processor in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, 43(1):29–41, 2008.

[10] M. Radetzki, C. Feng, X. Zhao, et al. Methods for fault tolerance in networks-on-chip. *ACM Computing Surveys*, 46(1):1–38, 2013.

[11] J. Duato, S. Yalamanchili, and N. Lionel. *Interconnection Networks: An Engineering Approach*, volume ISBN: 1558608524. Morgan Kaufmann Publishers Inc., 2002.

[12] Loukas Petrou and Julius Georgiou. Hardware design and manufacturing of the Hypersurface control nodes and hardware interface. VISORSURF Second Project Meeting, Berlin, July 2017.

[13] J. Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(10):1055–1067, 1995.

[14] C. Glass and L. Ni. The turn model for adaptive routing. *ACM SIGARCH Computer Architecture News*, 20(2):278–287, 1992.

[15] A. Vitkovskiy, V. Soteriou, and C. Nicopoulos. Dynamic fault-tolerant routing algorithm for networks-on-chip based on localised detouring paths. *IET Computers & Digital Techniques*, 7(2):93–103, 2013.

[16] K. Aisopos, A. DeOrio, L. Peh, et al. Ariadne: Agnostic reconfiguration in a disconnected network environment. In *Proceedings of the PACT*, pages 298–309. IEEE, 2011.

[17] J. Wu. A fault-tolerant and deadlock-free routing protocol in 2d meshes based on odd-even turn model. *IEEE Transactions on Computers*, 52(9):1154–1169, 2003.

[18] P. Bogdan, T. Dumitra, and Marculescu R. Stochastic communication: A new paradigm for fault-tolerant networks-on-chip. *VLSI Design*, 2007(95348):1–17, 2007.

[19] T. Saeed, C. Skitsas, D. Kouzapas, et al. Fault adaptive routing in metasurface controller networks. In *2018 11th International Workshop on Network on Chip Architectures (NoCArc)*, pages 1–6. IEEE, 2018.

[20] Peter E. Bulychev, Alexandre David, Kim Guldstrand Larsen, Marius Mikucionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. UPPAAL-SMC: statistical model checking for priced timed automata. In *Proceedings of QAPL 2012*, volume 85 of *EPTCS*, pages 1–16, 2012.

[21] S. Nowick and S. Montek. Asynchronous designpart 1: Overview and recent advances. *IEEE Design & Test*, 32(3):5–18, 2015.

[22] Konstantinos Aisopos, Chia-Hsin Owen Chen, and Li-Shiuan Peh. Enabling system-level modeling of variation-induced faults in networks-on-chips. In *Proceedings of the 48th Design Automation Conference*, pages 930–935. ACM, 2011.

[23] G. Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on parallel and distributed systems*, 11(7):729–738, 2000.

[24] Shubhangi D Chawade, Mahendra A Gaikwad, and Rajendra M Patrikar. Review of xy routing algorithm for network-on-chip architecture. *International Journal of Computer Applications*, 43(21):48–52, 2012.

[25] A. Patooghy and S. Miremadi. Xyx: A power & performance efficient fault-tolerant routing algorithm for network on chip. In *Proceedings of 27th the Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pages 245–251, 2009.

[26] A. Shafiee, M. Montazeri, and M. Nikdast. An innovational intermittent algorithm in networks-on-chip (noc). *World Academy of Science, Engineering and Technology*, 45:145–147, 2008.

[27] E. Pilakoutas. Development implementation and experimental evaluation of a fault torelant algorhitm in a manhattan topology. Master's thesis, Department of Computer Science, University of Cyprus, Nicosia, Cyprus, 2018.

[28] A. Borshchev. *The big book of simulation modeling: multimethod modeling with AnyLogic 6*. AnyLogic North America Chicago, 2013.

APPENDIX A
LFA RULES AND PSEUDOCODE

a) LFA Rules: :

- LFA has three sets of rules:

1) Rules to define the routing mode (i.e. normal or abnormal). Abnormal mode is when turn prevention is employed.
2) Abnormal routing non-boundary rules.
3) Abnormal routing boundary rules.
4) In both normal and abnormal routing 180 degree turns at the edge wraparounds are forbidden.

- These rules only apply when a fault is encountered.
- The cases that are not included in the abnormal routing rules are when simply XY-YX is employed.
- When a blocked turn is forced because of a fault while abnormal mode routing is activated, the algorithm terminates.

b) Routing mode selection: :

- If one of the following cases is true, abnormal mode is activated, otherwise only the routing algorithm is switched (XY_YX)

1) The destination is on a boundary row or column.
2) The destination is type 1, faulty node is type 3 and targets x-coordinate is 1.
3) The destination is type 2, faulty node is type 0 and targets x-coordinate is 22.
4) The destination is type 3, faulty node is type 2 and targets x-coordinate is 21
5) The destination is type 1 and faulty node is type 2 or type 3.
6) The destination is type 3 and faulty node is type 2.

c) Abnormal non-boundary rules: :

- At type 2 nodes to the East or at the same column of destination turns to North are forbidden.
- At type 3 nodes:
  - At nodes to the East or at the same column as the destination, WS turns are forbidden.
  - At nodes to the North or at the same row as the destination, SW turns are forbidden.
- When the destination is type 3:
  - At type 1 nodes: only turns to the South are allowed.
  - At type 2 nodes to the East of the destination, WN turns are forbidden.
  - At type 2 nodes to the West of the destination only turns to the North are allowed.

d) Abnormal boundary rules: :

- The destination is at the South boundary:
  - The destination is type 0: at nodes to the West from the destination forbid East_South turns and North_West turns.
  - The destination is type 0: at nodes to the East from the destination forbid South_West turns.
  - The destination is type 1: at nodes to the West from the destination forbid East_South turns.
- The destination is at the East boundary:
  - The destination is type 1: at nodes to the South or at the same row of the destination forbid North_East turns.
  - The destination is type 1: at nodes to the South and at the same column of the destination forbid East_East.
  - The destination is type 1: at nodes at the coordinate (destination.x-1) forbid West_North turns.
  - The destination is type 3: at type 1 nodes on the same column of and to the South from the destination by more than one node forbid East_East.
  - The destination is type 3: at nodes to the West North from the destination forbid South_East turns.
  - The destination is type 3: at type 1 nodes to the West North from the destination forbid East_East.
  - The destination is type 3: at type 0 nodes at y coordinate equal (destination.y - 1) forbid East_North.
  - The destination is type 3: at type 3 nodes to the South from the destination forbid West_South turns.
  - The destination is type 3: at nodes to the South from the destination and at the column before last forbid West_North turns.
- The destination is at the North boundary:
  - The destination is type 3: at nodes to the East from the destination forbid East_East. At nodes at x-coordinate equal (destination.x +1) forbid East _North and at nodes at x coordinate equal destination.x-1 forbid North_East.
  - The destination is type 2: forbid East_North turns, and West_North unless current node is at x coordinate equal (destination.x-1).

```
1 Routing mode selection
2 If the next node is faulty
```

```
  if destination is on a boundary row or column
    switch to abnormal routing
  elseif destination is type 1 && faulty node is type 3 && destination.x == min.x + 1
    switch to abnormal routing
  elseif destination is type 2 && faulty node is type 0 && destination.x == max.x - 1
    switch to abnormal routing
  elseif destination is type 3 && faulty node is type 2 && destination.x = max.x - 2
    switch to abnormal routing
  elseif destination is type 1 && faulty node is type 2 || faulty node type 3
    switch to abnormal routing
  elseif destination is type 3 && faulty node is type 2
    switch to abnormal routing
  else
    continue with normal routing
    switch routing from XY to YX or from YX to XY
  end
end

while (abnormal routing = 1)
  if destination node is on a boundary row or column
    if destination.y = 0  % south boundary
      if destination is type 0
        if source.x < destination.x
          block east south & north west turns
        elseif source.x > destination.x
          block south west turns
        end
      elseif destination is type 1 && source.x < destination.x
        block east south turns
      end
    elseif destination.x = n     % east boundary (max.x)
      if destination is type 1
        if source.y < destination.y && source.x = destination.x
          block east east routes
        elseif source.y <= destination.y
          block north east turns
        elseif source.x = (destination.x-1)
          block west north turns
        end
      elseif destination is type 3
        if source is type 1 && source.x = destination.x && source.y<destination.y-1
          block east east routes
        elseif source is type 1 && source.x < destination.x && source.y > destination.y
    block east east routes
        elseif source.x < destination.x && source.y > destination.y
    block south east turns
        elseif source is type 0 && source.y = destination.y-1
    block east north turns
        elseif source is type 3 && source.y < destination.y
    block west south turns
        elseif source.y < destination.y && source.x== n-1
    block west north turns
        end
      elseif destination.y = m      % north boundary max.y
        if destination is type 3
          if source.x > destination.x
            block east east routes
          elseif source.x = destination.x+1
            block east north
          elseif source.x = destination.x-1
            block north east
          end
        elseif destination is type 2
          if source.x != destination.x-1
            block east north & west north
          end
        end
      end
    else
```

```
      if source is type 2  &&  source.x >= destination.x
        block turns to north
      elseif source is type 3
        if source.x >= destination.x
          block west south turns
        elseif source.y >= destination.y
          block south west turns
        elseif destination is type 3
          if source is type 1
            only route to south
          elseif source is type 2 && source.x > destination.x
            block west north turns
          esleif source is type 2 && source.x < destination.x
            only route to north
          end
        end
      end
    end
  end
end
```

APPENDIX B
RDA RULES

RDA-Path1	
Turn Rules	
Condition	Decision
$c = 3$ and $x = 0$	Forbid north
$c \neq 3$ and $y = 0$	Forbid west
$c \neq 0$ and $y = 23$	Forbid east
$c = 0$ and $x = 23$	Forbid south
$D = 0$ and $x < a$ and $y < b$ and $c = 1$	Forbid west
$D = 0$ and $x = a$ and $y > b$	Forbid west
$x < a$ and $y > b$ and $c = 0$ and $D = 1$	Forbid west
$x > a$ and $y > b$ and $c = 2$ and $D = 1$	Forbid east
$x < a$ and $y < b$ and $c = 1$ and $D = 2$	Forbid west
$x = a$ and $y > b$ and $D = 2$	Forbid east
$x > a$ and $y < b$ and $c = 3$ and $D = 2$	Forbid east
$x < a$ and $y \geq b$ and $D = 3$	Forbid west
$x > a$ and $y > b$ and $D = 3$	Forbid west

Topology-agnostic XY-YX algorithms							
D=0		D=1		D=2		D=3	
Condition	Decision	Condition	Decision	Condition	Decision	Condition	Decision
$x < a$ and $y \leq b$	Obl. YX-XY	$x < a$ and $y \leq b$	Obl. YX-XY	$x \leq a$ and $y < b$	Obl. YX-XY	$x \leq a$ and $y < b$	Obl. XY-YX
$x < a$ and $y > b$	Obl. YX-XY	$x < a$ and $y > b$	Obl. YX-XY	$x < a$ and $y > b$	Obl. XY-YX	$x < a$ and $y \geq b$	Agn. YX-XY
$x > a$ and $y > b$	Agn. XY-YX	$x > a$ and $y \geq b$	Agn. YX-XY	$x > a$ and $y > b$	Obl. YX-XY	$x > a$ and $y > b$	Obl. YX-XY
$x > a$ and $y \leq b$	Obl. YX-XY	$x > a$ and $y < b$	Agn. XY-YX	$x > a$ and $y < b$	Obl. YX-XY	$x > a$ and $y < b$	Obl. YX-XY

Table I

RDA PATH 1 DECISION LOGIC. D: TARGET NODE TYPE , C: CURRENT NODE TYPE, X: CURRENT NODE X COORDINATE, Y: CURRENT NODE Y COORDINATE, A: DESTINATION NODE X COORDINATE, B: DESTINATION NODE Y COORDINATE.

RDA-Path2	
Turn Rules	
Condition	Decision
$c \neq 3$ and $x = 0$	Forbid north
$c = 3$ and $y = 0$	Forbid west
$c = 0$ and $y = 23$	Forbid east
$c \neq 0$ and $x = 23$	Forbid south
$D = 0$ and $x < a$ and $y < b$ and $c = 2$	Forbid north
$D = 0$ and $x < a$ and $y > b$ and $c = 3$	Forbid south
$x < a$ and $y < b$ and $c = 2$ and $D = 1$	Forbid north
$x < a$ and $y > b$ and $c = 3$ and $D = 1$	Forbid south
$x > a$ and $y > b$ and $c = 1$ and $D = 2$	Forbid south
$x > a$ and $y = b - 1$ and $D = 2$ and $c = 0$	Forbid north
$x \leq a$ and $y < b$ and $D = 3$	Forbid west
$x > a$ and $y >= b$ and $D = 3$	Forbid south
$x > a$ and $y < b$ and $c = 0$ and $y \neq 0$ and $D = 3$	Forbid west

Topology-agnostic XY-YX algorithms							
D=0		D=1		D=2		D=3	
Condition	Decision	Condition	Decision	Condition	Decision	Condition	Decision
$x \leq a$ and $y < b$	Obl. XY-YX	$x \leq a$ and $y < b$	Obl. XY-YX	$x \leq a$ and $y < b$	Obl. XY-YX	$x < a$ and $y < b$	Obl. YX-XY
$x \leq a$ and $y > b$	Obl. XY-YX	$x < a$ and $y > b$	Obl. XY-YX	$x < a$ and $y > b$	Agn. YX-XY	$x < a$ and $y > b$	Obl. XY-YX
$x > a$ and $y \geq b$	Agn. YX-XY	$x > a$ and $y \geq b$	Obl. XY-YX	$x \geq a$ and $y > b$	Agn. XY-YX	$x > a$ and $y \geq b$	Obl. XY-YX
$x > a$ and $y < b$	Obl. XY-YX	$x > a$ and $y < b$	Obl. YX-XY	$x > a$ and $y < b$	Obl. XY-YX	$x > a$ and $y < b$	Agn. XY-YX

Table II

RDA PATH 2 DECISION LOGIC. D: TARGET NODE TYPE , C: CURRENT NODE TYPE, X: CURRENT NODE X COORDINATE, Y: CURRENT NODE Y COORDINATE, A: DESTINATION NODE X COORDINATE, B: DESTINATION NODE Y COORDINATE.

Dr. Dimitrios Kouzapas is a Research Associate at the Department of Computer Science University of Cyprus. His conducting research for the purposes of the research programme ”VISORSURF: A Hardware Platform for Software-driven Functional Metasurfaces” He obtained his PhD at the Department of Computing, Imperial College London under the supervision of Prof. Nobuko Yoshida. His researh interests include theoretical computer science, formal methods, programming languages, and modelling and verification.

Taqwa Saeed is a PhD candidate at Frederick University, Cyprus. Her research work focuses on the development of information dissemination solutions for emerging network technologies, as for example, VANETs, molecular networks, and nano-networks using probabilistic and optimization methods.

Constantinos Skitsas is a Research Assistant at the University of Cyprus, employed by the research project VISORSURF: A Hardware Platform for Software-driven Functional Metasurfaces. He obtained his Bachelor’s degree in Computer Science from the University of Cyprus, Nicosia in 2018.

Marios Lestas, received the B.A and M.Eng degrees in Electrical and Information Engineering from the University of Cambridge U.K and the PhD degree in Electrical Engineering from the University of Southern California in 2000 and 2006 respectively. He is currently an Associate Professor at Frederick University. His research interests include application of non-linear control theory and optimization methods in Complex Networks such as Computer Networks, Transportation Networks, Power Networks, Molecular Nano-networks and Metasurfaces. In the aforementioned networks he has investigated issues pertinent to information dissemination, congestion control, network vulnerability, demand response and more recently privacy and security.

Vassos Soteriou received the B.S. and Ph.D. degrees in electrical engineering from Rice University, Houston, TX, in 2001, and Princeton University, Princeton, NJ, in 2006, respectively. He is currently an Associate Professor at the Department of Electrical Engineering, Computer Engineering and Informatics at the Cyprus University of Technology. He is a recipient of a Best Paper Award at the 2004 IEEE International Conference on Computer Design. His research interests lie in multicore computer architectures, and on-chip networks.

Anna Philippou is an Associate Professor at the Department of Computer Science, University of Cyprus, where she co-founded and co-directs the Laboratory on Foundations of Computing Systems and Theoretical Computer Science. She completed her undergraduate studies at the University of Oxford, UK (B.A. in Mathematics and Computation, 1992) and her graduate studies at the University of Warwick, UK (M.Sc. in Parallel Computers and Computation, 1993; PhD in Computer Science, 1997). Her research interests include Concurrency Theory and its Applications, Formal Methods for Safety-Critical Systems, Type Systems, and Algorithmic Game Theory.

Sergi Abadal is Project Director at the NaNoNetworking Center in Catalonia, Universitat Politecnica de Catalunya, where he also obtained his PhD in computer science engineering (2016). He has co-authored more than 50 research papers. In 2013, he was awarded by INTEL within his Doctoral Student Honor Program. He is Associate Editor of the Nano Communication Networks (Elsevier) Journal. His research interests include on-chip networking, many-core architectures, and graphene-based wireless communications.

Christos Liaskos received the Diploma in Electrical Engineering from the Aristotle University of Thessaloniki (AUTH), Greece in 2004, the MSc degree in Medical Informatics in 2008 from the Medical School, AUTH and the PhD degree in Computer Networking from the Dept. of Informatics, AUTH in 2014. He is currently a researcher at the Foundation of Research and Technology, Hellas (FORTH). His research interests include computer networks, traffic engineering and novel control schemes for wireless communications.

Loukas Petrou received his Bachelor Degree in Electrical Engineering from the University of Cyprus in 2016. He holds a Master's Degree in Analogue and Digital Integrated Circuit Design from Imperial College London. He is currently pursuing a PhD degree in Electrical Engineering at the University of Cyprus. His research interests include integrated circuit design and asynchronous digital circuits.

Julius Georgiou (IEEE M98-SM08) is an Associate Professor at the University of Cyprus. He received his M.Eng degree in Electrical and Electronic Engineering and Ph.D. degree from Imperial College London in 1998 and 2003 respectively. For two years he worked as Head of Micropower Design in a technology start-up company, Toumaz Technology. In 2004 he joined the Johns Hopkins University as a Postdoctoral Fellow, before becoming a faculty member at the University of Cyprus from 2005 onwards. Prof. Georgiou is a member of the IEEE Circuits and Systems Society, was the Chair of the IEEE Biomedical and Life Science Circuits and Systems (BioCAS) Technical Committee, as well as a member of the IEEE Circuits and Systems Society Analog Signal Processing Technical Committee. He served as the General Chair of the 2010 IEEE Biomedical Circuits and Systems Conference and is the Action Chair of the EU COST Action ICT-1401 on Memristors-Devices, Models, Circuits, Systems and Applications - MemoCIS. Prof. Georgiou was an IEEE Circuits and Systems Society Distinguished Lecturer for 2016-2017. He is also is an Associate Editor of the IEEE Transactions on Biomedical Circuits and Systems and Associate Editor of the Frontiers in Neuromorphic Engineering Journal. He is a recipient of a best paper award at the IEEE ISCAS 2011 International Symposium and at the IEEE BioDevices 2008 Conference. In 2016 he received the 2015 ONE Award from the President of the Republic of Cyprus for his research accomplishments. His research interests include Low-power analog and digital ASICs, implantable biomedical devices, bioinspired electronic systems, electronics for space, brain-computer-interfaces (BCIs), memristive devices, inertial and optical sensors and related systems.

Andreas Pitsillides is a Professor in the Department of Computer Science, University of Cyprus, co-director of the Networks Research Laboratory (NetRL, http://www.NetRL.cs.ucy.ac.cy), and appointed Visiting Professor at the University of the Witwatersrand (Wits), School of Electrical and Information engineering, Johannesburg, South Africa. His broad research interests include communication networks, Software Defined Metamaterials (including Hypersurfaces and intelligent communication surfaces), Nanonetworks, Internet- and Web- of Things, and Smart Systems (e.g. Smart Grid) and Smart Spaces (e.g. Home, City). He has a particular interest in adapting tools from various fields of applied mathematics such as adaptive non-linear control theory, computational intelligence, game theory, and complex systems and nature inspired techniques, to solve problems in communication networks. Published over 270 refereed papers in flagship journals (e.g. IEEE, Elsevier, IFAC, Springer), international conferences, and book chapters, co-authored 2 books (1 edited), participated as principal or co-principal investigator in over 40 European Commission and locally funded research projects with over 6.6 million Euro, received several awards, including best paper, presented several keynotes, invited lectures at major research organisations, short courses at international conferences and short courses to industry.