SpiNNaker: Impact of Traffic Locality, Causality and Burstiness on the Performance of the Interconnection Network

Javier Navaridas†, Luis A. Plana*, Jose Miguel-Alonso†, Mikel Luján*, Steve Furber*

† Dpt. of Computer Architecture and Technology
The University of the Basque Country, Spain
Manuel de Lardizabal, 1. 20018 San Sebastian

* School of Computer Science
The University of Manchester
Oxford Road, Manchester M13 9PL, UK.

{javier.navaridas, j.miguel}@ehu.es {plana, mikel.lujan, steve.furber}@manchester.ac.uk

ABSTRACT

The SpiNNaker system is a biologically-inspired massively parallel architecture of bespoke multi-core System-on-Chips. The aim of its design is to simulate up to a billion spiking neurons in (biological) real-time. Packets, in SpiNNaker, represent neural spikes and these travel through the two-dimensional triangular torus network that connects the over 65 thousand nodes housed in the largest size of SpiNNaker.

The research question that we explore is the impact that spatial locality, temporal causality and burstiness of the traffic have on the performance of such interconnection network. Given the limited knowledge of neuron activity patterns, we propose and use synthetic traffic patterns which resemble biological neural traffic and allow tuning of spatial locality. Causality is explored by means of temporal patterns that maintain a specified overall network load while allowing at the node level autonomous causal traffic generation. Part of the traffic is generated automatically, but the remaining traffic is triggered by a spike arrival in the form of a packet or a burst of packets; as neural stimuli do. In this way, we generate non-uniform traffic patterns with an evolving concentration of activity at nodes which contain more active parts of the spiking neural network.

Given the application domain, the simulation-based study focuses on the real-time behavior of the system rather than focusing on standard HPC network metrics. The results show that the interconnection network of SpiNNaker can operate without dropping packets with traffic loads that exceed more than 3.5 times those required to simulate 10^9 spiking neurons, despite using non-local traffic. We also find that increments in the degree of traffic causality do not affect the performance of the system, but burstiness in the traffic can hurt performance.

CF'10, May 17–19, 2010, Bertinoro, Italy.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *Network topology.*

C.3 [**Special-Purpose and Application-Based Systems**]: – *Real-time and embedded systems.*

C.4 [**Performance of Systems**]: – *Modeling Techniques, Performance attributes.*

General Terms

Performance, Design, Verification.

Keywords

Interconnection Networks, Massively Parallel Systems, Performance Evaluation, Power-Efficient Architectures, Real-time Applications, Spiking Neural Networks, System-on-Chip, Traffic Characterization.

1 INTRODUCTION

SpiNNaker is a massively parallel architecture designed to model large-scale spiking neural networks in biological real-time. Its design is based around *bespoke* multi-core System-on-Chips which are interconnected using a two-dimensional triangular torus. Neural models running in the system communicate by means of spike events that occur when a neuron is stimulated beyond a given threshold and then fires. Spike events are communicated to all connected neurons, with typical fan-outs on the order of 10^3. Applications such as these have abundant parallelism and no explicit requirement to maintain consistency in shared memories. Another characteristic of the biological process is its natural resilience to failures: neurons may die, spikes may be missed, but the brain remains functioning appropriately. Furthermore, the biological process advances at very low pace when compared to standard electronic components: milliseconds versus microseconds [4]. The design of SpiNNaker takes advantage of these characteristics to deploy a well-balanced, low-power massively parallel architecture. The largest configuration (to be deployed by 2012) houses 2^{16} nodes creating a system with over one million computing cores capable of simulating spiking neural networks with up to one billion (10^9) neurons. To put this number into perspective, a human brain contains approximately 10^{11} neurons.

In a previous paper [13], we justified the interconnection network, characterized analytically some of its topological properties, and

investigated appropriate values for certain timeout parameters of the packet dropping mechanisms used to avoid *deadlock* and *livelock*. We also investigated the temporal evolution of the system under different levels of degradation due to faults, showing the suitability of a novel mechanism to keep the system working stably: the *emergency routing*. A limitation of that study was its reliance on plain uniform, point-to-point traffic from independent traffic sources.

This paper overcomes this and carries out a more extensive evaluation using more complex and realistic traffic patterns. In particular, we explore how traffic locality, causality and burstiness may affect the performance figures of the interconnection network.

Given the limited scientific knowledge of activity patterns in large biological spiking neural networks [5][18], we use synthetic traffic patterns which resemble biological neural traffic while allowing tuning spatial locality: from very local traffic with packets traveling small distances, to very distant ones in which packets are likely to travel across the whole network. Moreover we explore causality by means of temporal patterns that use a two-folded traffic generation scheme: independent traffic generation plus causal traffic generation. In other words, part of the traffic is generated automatically, but the remaining traffic is triggered by the arrival of a packet (as neural stimuli do) in the form of a packet or a burst of packets.

The structure of this paper is organized as follows: Section 2 describes the architecture of the SpiNNaker system. The experimental set-up of the simulation-based environment is discussed in Section 3. Section 4 presents and discusses the results of the experimental evaluation. Section 5 is devoted to discuss some related projects. Finally, Section 6 closes this paper with the main conclusions of this research work.

2 SPINNAKER ARCHITECTURE

SpiNNaker is a system-on-chip (SoC)-based architecture designed to support the real-time simulation of large networks of spiking neurons (up to 10^9). To emulate the very high connectivity of biological systems, SpiNNaker uses a self-timed, packet-switched interconnection network which gives support to efficient multicast, with high bandwidth and low-delay. The heart of the communication infrastructure is an on-chip router and the self-timed implementation of the fabric that allows the seamless extension of the on-chip communications to include the inter-chip links. We encourage the interested reader to check [15] for a more detailed description of the system at hardware level and [7] for an engineering-oriented overview of biologically plausible neural networks applications and systems.

2.1 SpiNNaker Node

The basic block of the interconnection network is the SpiNNaker chip. It contains one multi-core SoC with 20 low-power ARM968S-E processing cores and one SDRAM chip. On-chip, each ARM core has a tightly-coupled dedicated memory that can hold 32KBytes of instructions and 64Kbytes of data. Processing units are provided with other useful modules such as the timer, the vector interrupt controller (VIC in the figure), the communication controller and the DMA controller. A conceptual representation of the SpiNNaker chip is depicted in Figure 1. Detailed low-level simulations of the chip using Verilog and SystemC confirmed that each of the cores is able to simulate up to around 1000 individual neurons [9].

All the cores in a chip share a SDRAM in which synaptic connection information is stored. Access to this shared storage space is carried out by means of a self-timed NoC [6] which is used to connect resources in the chip. This NoC provides higher communication bandwidth (8 Gbps), lower contention and lower consumption than any typical bus-based architecture [16]. Each chip is also provided with other subsystems used for booting and managing purposes, which are also accessed through the NoC. The Boot ROM stores the minimal code to boot and verify the chip. The system controller is in charge of performing management functions at the chip level. Finally, every chip has an Ethernet connection that can be used for loading the application data [11], managing the system from a console and also for fault-tolerance purposes. However, given that this sub-module requires a dedicated core, only a few chips of the system will make use of this connection, being them in charge of distributing through the SpiNNaker interconnection network the necessary Ethernet received traffic.

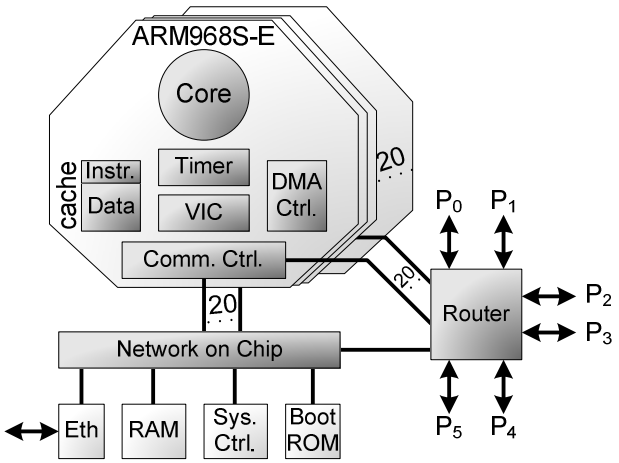

Figure 1. Schematic model of the SpiNNaker chip with all its components depicted.

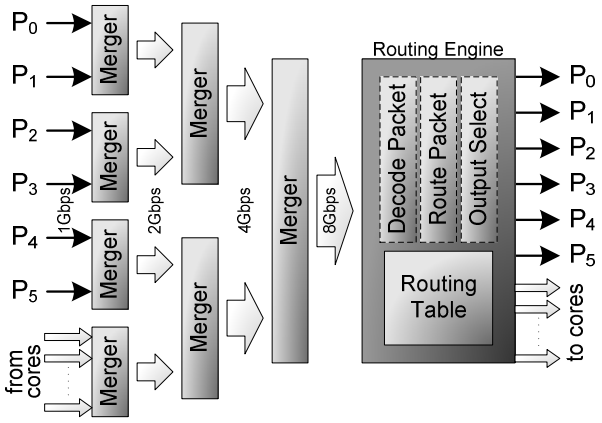

Figure 2. Architecture of the router. Black arrows represent links outwards from the chip. White arrows represent hard-wired links within the chip.

2.2 SpiNNaker Router

A depiction of the router is shown in Figure 2. It has 20 ports for internal use of the ARM cores and six ports to communicate with six adjacent chips. All ports are full-duplex and implement self-timed protocols. The organization within the router is hierarchical; ports are merged in three stages before using the actual routing engine. Note that the router is able to forward a single packet at once, but works faster than transmission ports. Therefore, most of the time, routers will be idle, and router delay barely affects the pace at which packets are processed.

The router is designed to support point-to-point and multicast communications, as required by the target applications. The multicast engine helps reducing pressure at the injection ports, and, compared to a pure point-to-point alternative, it reduces significantly the number of packets that traverse the network. Information interchange is performed using small, 40-bit packets. It is important to indicate that routers make routing decisions based on the source address (neuron identifier) of the packets. Hence, packets do not contain any information about its destination(s), only the neuron that fired it. The network itself will deliver the packets to all chips containing neurons that have synaptic connections with the source neuron. These connections are embedded in the 1024-word routing tables inside the routers, and must be preloaded using application-specific information. To minimize the space pressure on the routing tables, these offer a masked associative route look-up. In addition the routers are designed to perform a default routing that sends the packet following a straight line, a process that avoids using extra entries in the routing tables. For example, if the packet comes from the North it will be sent to the South. The expected shape of the routes among chips is formed by two straight lines connected at a single inflection point to keep the number of entries in each table as low as possible [10].

The network topology allows two-hop routes to go from a chip to each one of its neighbors—see Figure 3. These two-hop paths between neighbor nodes are known as *emergency routes* and may be invoked to bypass problematic links due to transient congestion states or link failures. In order to minimize chip area, the *emergency routing mechanism* implements only one of these turns. Our previous study [13] showed the potential of such mechanism for keeping the system operating stably despite considering scenarios in which the interconnection network suffered of high number of link failures (up to 1024).

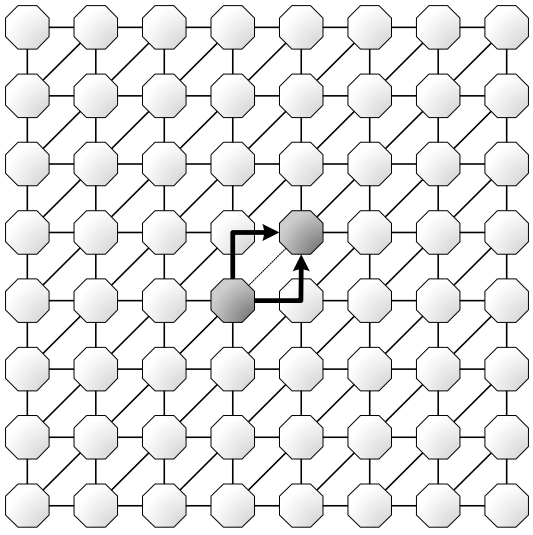

Figure 3. Example of an 8×8 SpiNNaker topology. Peripheral connections are not depicted for the sake of clarity. The regular route (slashed line) and the two emergency routes (thick arrows) between the shaded nodes are shown.

The SpiNNaker flow-control is straightforward. When a packet arrives to an input port, one or more output ports are selected and the router tries to transmit the packet through them. If the packet cannot be forwarded, the router will keep trying, and after a given amount of time it will also test the clockwise emergency route—both the regular and the emergency route will be checked. Finally, if a packet stays in the router for longer than a given threshold—a router parameter: *waiting time*—the packet will be dropped to avoid deadlock scenarios. To avoid livelock situations, packets have an *age* field in their header. When two ages pass and the packet is still in transit, it is considered as *outdated* and dropped. The ages are global to the whole system and its time-span is arbitrary, a system configuration parameter. Appropriate values for these parameters were provided in our previous study [13].

Emulating the behavior of biological neural networks, dropped packets in SpiNNaker are not re-sent. Losing neurons (one per second in human brains) or signals does not impede the normal functioning of the biological processes; although, the dropping level must be kept (*very*) low. We consider acceptable any packet-dropping level below one dropped packet every 10^6 injected.

2.3 Interconnection Network Topology

SpiNNaker chips are arranged in a two-dimensional triangular torus topology with links to the neighbors in North, South, East, West, Southwest and Northeast. An 8×8 instance of this topology is depicted in Figure 3. Note that chips at the network boundaries are connected by means of peripheral, wrap-around links that are not shown in the figure for the sake of clarity. The topological characteristics of the SpiNNaker interconnection network were analytically derived in [13], and validated by means of simulation. Furthermore, in that work we computed the expected network utilization during regular operation of the system. This value was roughly a packet generation rate of 0.01 packets/cycle/node. In the following experiments, we represent this generation rate as **RO**, standing for Regular Operation.

Using the 6-port router within the SpiNNaker chip, the system could be arranged as a three-dimensional (3D) torus, which has theoretically superior topological properties (*e.g.*, bisection bandwidth and distance-related characteristics) than those of the topology of SpiNNaker. However the topology of SpiNNaker has some advantageous properties: a two-dimensional system is easier to deploy and the diagonal links add redundancy to the design, a redundancy that can be exploited using the previously described emergency routing mechanism. Note that a three-hop emergency routing could be implemented in a 3D torus, but the extra chip area required makes it unaffordable in the context of SpiNNaker. It is also noticeable that routing in a 3D torus requires more entries in the routing tables, as regular routes are composed by three straight lines instead of two. This would increase the entries in the routing tables roughly by a 25% which may force to increase the number of entries in each table and, therefore, the chip area. In our previous paper [13], we compared the behavior

of the two topologies, showing how the availability of the emergency routing tipped the scale in favor of the SpiNNaker topology as it provided a more stable behavior across different scenarios of network degradation.

3 EXPERIMENTAL SET-UP

We perform a simulation-based evaluation in which the main figure of merit is the packet dropped ratio. As explained previously, the application modeled by SpiNNaker tolerates some degree of packet loss, but it must be kept low. Any packet dropped ratio below 10^{-6} is considered acceptable. This section describes the environment used to collect the results.

3.1 Model of the System

A detailed model of the SpiNNaker interconnection network is implemented in INSEE, a fast, flexible and mature simulation environment [17] for interconnection networks. The developed node model contains most of the features of the router, as well as the topological arrangement. In order to be able to confront simulations of large-scale systems, some simplifications are taken. We model a cycle as the time to route and forward a packet. Since routing is faster than transmission, the router can process several packets in a single cycle, provided that all the involved input and output ports are different.

This study evaluates the largest configuration of the system, which is composed by 2^{16} nodes arranged on a 256×256 layout. The model of the router includes the emergency routing and deadlock avoidance mechanisms (packet-dropping). The clockwise emergency routing is checked during the last 3 cycles before dropping the packet. As suggested in [13], the *waiting time* parameter of the deadlock avoidance mechanism has been fixed to 5 cycles.

Regarding the routing tables, the actual system will configure them on a per biological network basis. As our evaluation should not be tied to any particular biological network, the table-based routing is not used, a simplification that significantly reduces the computing resources required to perform simulations. As the regular routes between chips in the actual system will attempt to use a minimal path with a single inflection point [10], packets are sent through minimal routes using Dimension Order Routing (DOR). When applying DOR, the diagonal links are considered a third dimension (Z), therefore the routes followed by packets were always XY, XZ or YZ—note that a XYZ route can not be a minimal path.

The nodes are modeled as independent traffic sources that inject packets following a Bernoulli temporal distribution, in which the packet injection rate (packets/cycle/node) can be tuned to any desired value. Furthermore we provide them with the capability to react to receiving a packet by generating a new packet or a burst of packets. We model this reactive traffic with two parameters, the probability to trigger a new packet (p) and the number of packets that are triggered (n). Given that all the ports from the cores inside a chip are merged, we model the whole set of cores as a single injection queue with room for up to four packets. If this queue is full and a core tries to inject, the packet is dropped because there is no room to store it.

3.2 Workloads

We propose workloads that introduce the possibility to modulate the locality and causality of the traffic while resembling the kind of workloads that will execute on SpiNNaker. In previous work [10], the neuron-to-core mapping was explored with the purpose of increasing locality, *i.e.* reducing the distance among communicating nodes. In this paper we want to explore how critical the mapping of the neurons onto the system may become.

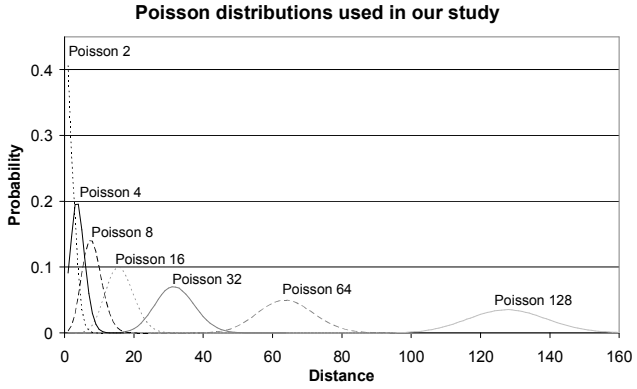

Figure 4. The Poisson distributions used to model different degrees of locality.

Despitve the application model being known, there is no detailed biological expression describing concrete and detailed brain activity. For this reason, this evaluation uses Poisson distributions with different values of its *lambda* (λ) parameter to simulate the way packets spread through the network. Note that this is a comprehensible model of the kind of traffic that will be executed over SpiNNaker. A combination of Poisson distributions with different lambdas could be a more accurate model but, for the sake of simplicity, we will restrict the study to a single Poisson.

For every injected packet, we first select, using the given distribution, the distance (number of hops) that the packet will traverse and then randomly select a destination node located at this distance. We have used seven different values for the lambda parameter of the Poisson distribution with the purpose of modeling different degrees of locality, from very local (λ=2) to very distant (λ=128). Figure 4 shows the distance distributions generated by each of the used values of λ. The greater the value of λ the more distant is the generated traffic.

Uniform traffic, as the used in [13], has no implicit bottleneck in its definition and, consequently, produces a balanced use of network resources. In contrast, the Poisson-based spatial patterns used in this paper do not guarantee balanced network usage and, therefore, bottlenecks may appear. Capturing this effect is desirable, as the traffic generated by the applications running on SpiNNaker may not (and probably will not) exhibit a balanced utilization of network resources.

3.3 Experimental Methodology

The experiments are carried out with the following methodology. We start with an empty network that is fed by the selected workload. A warm-up phase of 25K cycles is executed, after which a convergence phase starts in which the figures of interest are measured and collected every 10^3 cycles. Once three consecutive intervals are within a range of ±5% difference, the system is considered stable, and a statistics collection phase starts. In this phase the statistics are collected for 10 intervals of 10^4 cycles each. In the plots we present the average values of these 10

intervals. As the figures of merit are captured once the system has converged, the standard deviations are negligible and therefore are not plotted. In the graphs illustrating system evolution, the average of each figure of merit is captured every 10 cycles, which allows capturing the dynamic behavior of the interconnection network.

The first set of experiments uses independent traffic sources (non-causal traffic) with a wide range of traffic generation rates;from 0.1·**RO** (0.001 packets/cycle/node) to 10·**RO** (0.1 packets/cycle/node). In this way, we can observe the relation between the degree of locality and the ratio of dropped packets.

In the second set of experiments, we model traffic causality using the trigger mechanism explained previously: the reception of a packet can generate a single packet or a burst of them. To make fair comparisons we test different configurations that manage the same overall amount of traffic. To do so, we have to solve the following equation,

$$G = i + i \cdot \sum_{k=1}^{\infty} n \cdot p^k$$

in which G is the desired total traffic generation rate, i is the independent traffic generation rate, p is the probability to trigger a new burst and n is the amount of packets generated in each burst.

In this set of experiments, we fix the generation rate (G) to **RO** (0.01 packets/cycle/node) and consider different levels of causal generation of packets. The values of p are deliberately selected to generate constant values, across configurations, for the independent generation rate: i=0.0099, i=0.009, i=0.0075 and i=0.005 which correspond to 99%, 90%, 75% and 50% of the desired G. Thus, for the causal generation of a single packet (n=1) we use four different probabilities of triggering a packet: p=0.01, p=0.1, p=0.25 and p=0.5. Similarly, for the causal generation of a burst of 5 packets (n=5) we use four different probabilities of triggering a burst: p=0.002, p=0.02, p=0.05 and p=0.1. For the causal generation of a burst of 10 packets (n=10) we use: p=0.001, p=0.01, p=0.025 and p=0.05. Finally, for the causal generation of a burst of 20 packets (n=20) we use p=0.0005, p=0.005, p=0.0125 and p=0.025. Notice that a burst of 20 packets corresponds to a scenario where all the cores in the chip respond to a packet reception.

These experiments study whether the causality and burstiness implicit in spiking neural networks can generate excessive injection-level contention and hurt the performance of the system. However, as we will see later, the system can handle the **RO** and loads that are considerably higher. Thus, to learn more about the behavior of the interconnection network, we also consider scenarios with much higher loads, which will be described later.

4 ANALYSIS OF RESULTS

4.1 Locality Study

The first set of experiments in our study revolves around the impact that different degrees of traffic locality can have on the performance of the interconnection network, measured as the amount of packets dropped and the lowest generation rate at which the system is forced to drop packets. As discussed before, this study will provide some insights about the importance of neuron mapping onto the SpiNNaker system.

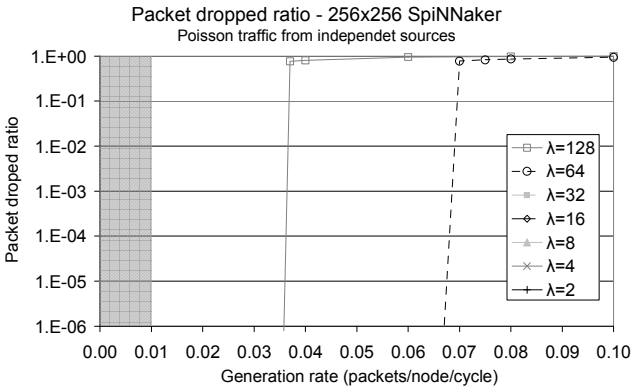

Figure 5. Packet dropped ratio per configuration using Poisson-spatial traffic from independent sources. Shaded area amid 0.00 and 0.01 represents SpiNNaker's regular operation.

Figure 5 shows the ratio of packets dropped for each configuration. The X-axis shows the traffic generation rate (using independent sources) and the Y-axis the measured ratio of dropped/injected packets. Values not plotted are equal to zero, which means that no packet is dropped. The shaded area at the left delimits the expected injection rates during **RO**. The results show that traffic locality has a great impact on the behavior of the SpiNNaker interconnection network as the load at which it is forced to drop packets is inversely proportional to the λ parameter. The system can handle all the injected traffic, regardless of their degree of locality, for injection rates up to 3.5·**RO**. For those distributions that do not expose very distant spatial distributions (λ≤32), we can see that even with a network pressure in excess of ten times **RO,** the system is able to manage the traffic without dropping any packet. This behavior reinforces the impression of robustness encountered in [13].

In the experiments, we also observed that, when the network operates with a load either well below or well above saturation, it reaches the steady-state after a short transient period. When the load is low, the steady-state is reached after a few hundred cycles. Note that in such state, the system behaves perfectly: no packet is dropped and latencies are low. Similarly, when the network is working under high pressure the steady-state is reached after one thousand cycles. In this case, the system behavior is unacceptable: most of the packets are dropped and those packets that are *lucky* enough to reach their destination suffer from severe latencies (more than 5 times those experienced in non-saturated scenarios).

In contrast with the rapid convergence observed in the previous scenarios, when the system is working at injection levels close to its saturation point, the temporal evolution of performance indicators is somewhat different. They rapidly progress, in just a few hundred cycles, to a phase in which the system behaves as non-saturated. During this phase the network operates correctly and is able to deliver all the injected traffic. However, after a long interval of several thousand cycles, the network enters into a long transient phase (that spans a few thousand cycles) in which the network starts to collapse, leading to the steady-state phase in which the network is severely affected by saturation. This behavior was not found when working with uniform traffic. Any unbalance introduced by the Poisson traffic generates network bottlenecks. The contention around these bottlenecks is eventually

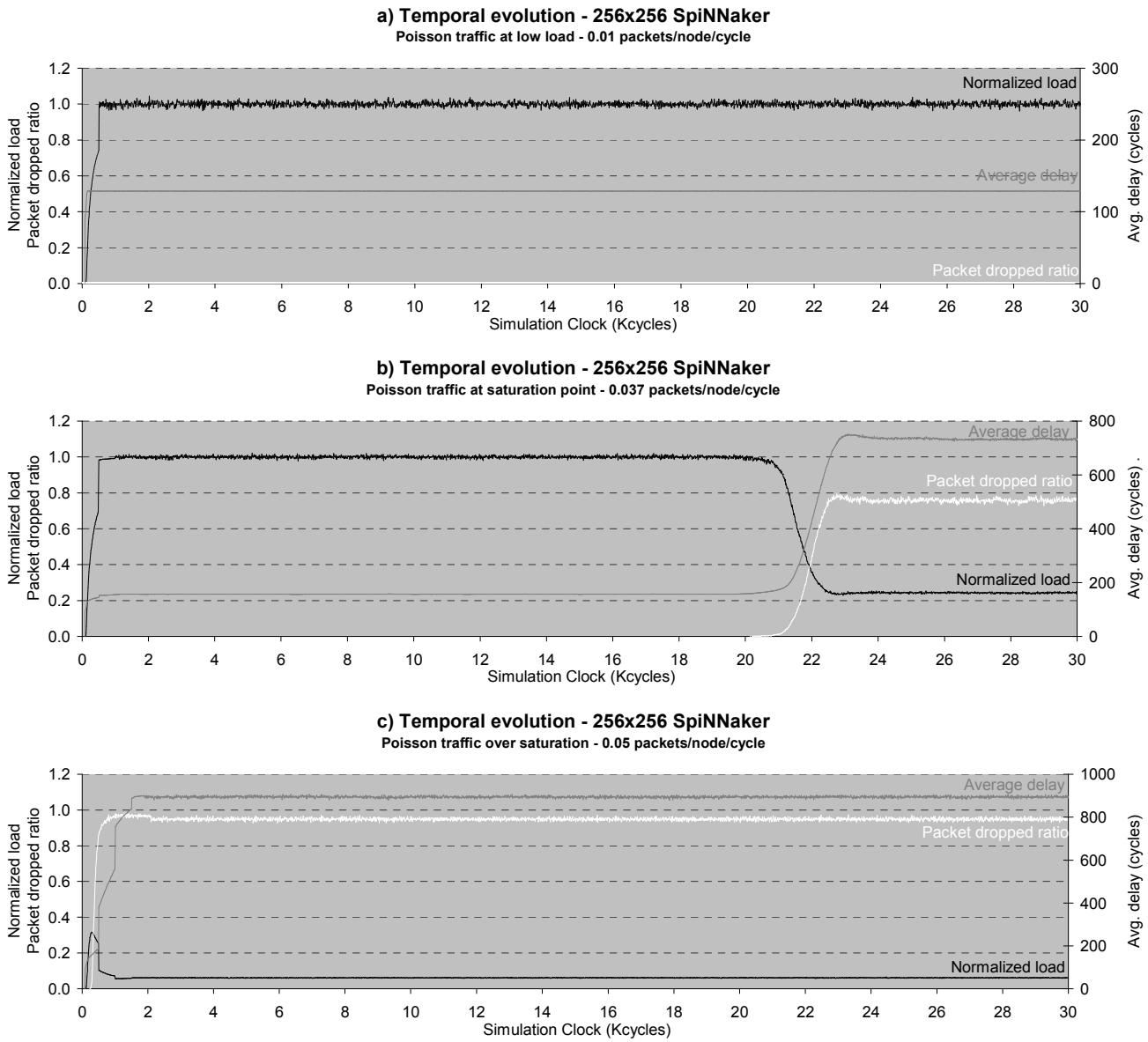

Figure 6. Temporal evolution using Poisson-spatial traffic at different loads. a) Non-saturated at RO load (0.01 packets/cycle/node). b) Saturation point (0.037 packets/cycle/node). c) Over saturation (0.05 packets/cycle/node).

spread to the whole network, in which case the network operates as severely saturated.

The temporal evolution of the network when fed with **RO** load (non-saturated), with a load close to the previously observed saturation point (at 0.037 packets/cycle/node) and with a load well beyond the saturation point (at 0.05 packets/cycle/node) are plotted in Figure 6a, b and c, respectively. We limited the plot to the first 30 Kcycles of the simulation, with the purpose of allowing a clear visualization of the transient states. Three important performance metrics are plotted. The first one is the load accepted by the system, which is normalized to the provided load in order to allow an easy comparison of system behavior. The packet dropped ratio and the average delay suffered by the packets are also plotted. The plotted figures were captured every 10 simulation cycles.

Another interesting finding from these experiments is that, in those cases in which the network reaches saturation, the distance distribution computed at injection and that measured at consumption (considering those packets that are actually consumed) are noticeably different. Figure 7 shows the cumulative distance distribution of the system when being fed by the most distant traffic (λ=128), at loads below and over the saturation point. Three figures of merit are plotted: the first one is the distance distribution at injection (*Di*), computed as the number of hops in the shortest path between source and destination. The second is the distance distribution at consumption (*Dc*), also computed as the shortest path. Finally, the distribution of the

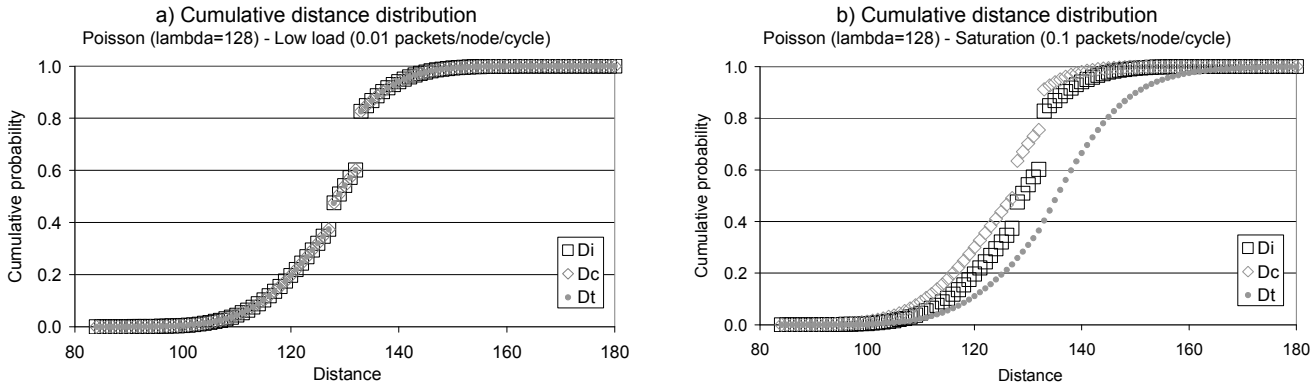

Figure 7. Cumulative distance distribution functions measured at injection and at consumption. a) Network fed at RO level (non-saturated). b) Saturated network at 10·RO load.

distance actually traveled by the packets, measured as the actual number of hops the packet traveled, is also plotted (*Dt*). We want to remark that the utilization of the emergency routing mechanism only affects *Dt.*

Note how these three distributions are almost identical when the system is operating under a low load, **RO** in Figure 7a. In saturated scenarios, 10 **RO** in Figure 7b, the distance distribution at injection does not experience any change. However the distance distributions at consumption are noticeably different. *Dc* is shifted to the left (shorter distances) meaning that those packets that have to travel longer distances are more likely to be dropped. In contrast, *Dt* is shifted to the right (longer distances). This is because the frequent activation of the emergency routing mechanism, due to the saturation scenario, is reflected as an increase in the number of hops actually traveled by the packets.

4.2 Causality and Burstiness Study

The second set of experiments focuses on measuring the impact that traffic causality and burstiness have on the performance of the interconnection network. We simulate the network with causal traffic at **RO** level (0.01 packets/cycle/node). For that level of utilization none of the workloads forced the system to drop any packet. Consequently, we need to put additional stress on the network in order to capture the impact that the traffic properties have on the performance of the network. For this reason, we use the previously defined burst lengths (n) and probabilities of triggering a burst (p), and use total generation rates (G) that are around the point at which the traffic from independent sources forced the system to drop packets (0.037 packets/cycle/node). We only show the results for the most distant traffic (λ=128), but the conclusions also hold for other λ values when managing loads close to their corresponding saturation points. Values λ=64 and λ=32 were checked, but not plotted for the sake of brevity.

Figure 8 shows the packet dropped ratio for each burst length. In all cases, the higher degree of causality in the traffic the lower the packet dropped ratio is once the system reaches saturation. This is inherent to the causality of the traffic because, when packets are dropped they do not reach their destination and, therefore, they do not trigger other packets. For this reason the actual generation rate is not as high as expected, which can be seen as a form of self-throttling of the workload. Another important discovery is that traffic burstiness affects the injection rate at which the network is forced to drop packets. The larger the burst length and the probability to trigger a burst are, the lower the generation rate at which the network starts dropping packets. We want to remark, however, that the used loads are more than three and a half times the required one during the regular operation of the system.

We found that traffic locality and burstiness have a noticeable impact on the performance of the interconnection network of SpiNNaker. Another interesting finding is that the causal traffic shelf-throttles: an increase in the number of dropped packets leads to the decrease of the injection rate. To conclude, we have found that the system is able to manage workloads that are significantly more demanding than those expected during the regular operation of the system, even in those scenarios in which the traffic exhibited undesirable characteristics such as a low degree of locality and large bursts. We can derive, hence, that the neuron-to-node mapping while being important is not going to become a critical issue when simulating actual neural activity with SpiNNaker.

5 RELATED WORK

Research in simulating biologically plausible neural networks (*brain-like* systems) has remained a *hot topic* for the last decades. In the early nineties a team at U.C. Berkeley worked in the Connectionist Network Supercomputer [1]. This project aimed to build a supercomputer specifically tailored for neural computation as a tool for connectionist research. The system was designed to be implemented as a two-dimensional mesh, with a target size of 128 nodes (scalable to 512). Each node would incorporate a general-purpose RISC processor plus a vector coprocessor, 16MB of RAM and a router. To our knowledge, a prototype of the node was built (under the codename T0), but the system never operated as a network. Experiments using up to five nodes in a bus configuration were discussed in [14].

More recently, the Microelectronics Division at the T.U. of Berlin worked in a project [12] whose objectives were similar to those of SpiNNaker. Part of this project is an acceleration board, called SSE, implemented with a collection of FPGAs interconnected via an on-board bus. An SEE accelerator is able to perform neural computations 30 times faster than a desktop PC [8]. Other projects used FPGAs for similar purposes, obtaining speedups of up to 50 compared to software-only implementations. However, as these boards cannot be connected to form a network, they are not able to scale to the magnitudes of SpiNNaker.

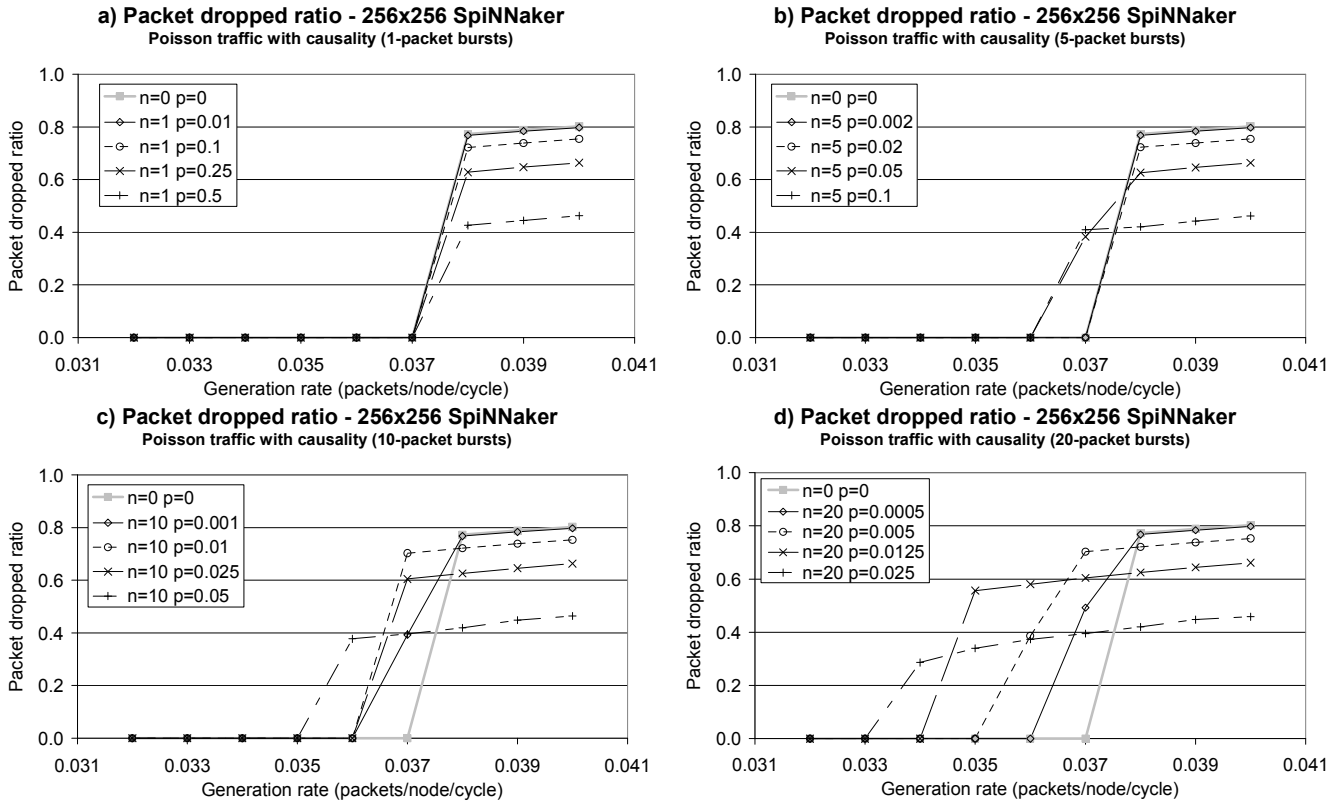

Figure 8. Packet dropped ratio per configuration using Poisson-spatial traffic from bursty causal sources. As reference, the thick grey line represents this figure when independent traffic sources are used. a) 1-packet bursts. b) 5-packet bursts. c) 10-packet bursts. d) 20-packet bursts.

As far as we know, the only active project comparable to SpiNNaker in terms of simulation scale is the BlueBrain project [2] which aims to create a biologically accurate functional model of the brain. However, the high complexity of its neuronal model does not allow it to work in real-time. In contrast with the biologically-inspired SpiNNaker architecture, the BlueBrain project does not contemplate the construction of any specific computing system but uses a general-purpose supercomputer, an IBM BlueGene [3].

6 CONCLUSIONS

This paper investigates the performance of the interconnection network of SpiNNaker, a system designed for real-time simulation of biologically plausible spiking neural networks. In particular, for the first time we have analyzed the impact that traffic locality and causality have on the real-time behavior. The system has been evaluated under several different scenarios using Poisson spatial distributions that allow tuning the degree of traffic locality, while being a reasonable model of the actual traffic that has to be supported by the system.

Similarly to other interconnection architectures, traffic locality has a strong influence on the performance of SpiNNaker interconnection network. However, even in the case of very distant patterns, the network is able to manage the traffic without dropping packets with injection rates well above three times the required during regular system operation. This means that the neuron mapping will not become a crucial issue. Moreover, for those distributions that do not impose very distant communication the system can manage the traffic for injection times up to 10 times the required during regular operation, which remarks the system robustness found in previous evaluations [13].

Regarding the causality and burstiness of the traffic we have found that the burstiness in the generation of traffic may generate contention around the node that is injecting. This contention may lead to dropping packets at loads at which the network would operate flawlessly with non-causal traffic, but still significantly higher than the load required during the regular operation of the system. Another interesting property that we found is that causal traffic manages to self-throttle, because the dropping of packets leads to the reduction of the packet generation rate.

7 ACKNOWLEDGMENTS

The SpiNNaker project is supported by the Engineering and Physical Sciences Research Council, through Grants EP/D07908X/1 and GR/S61270/01, and also by ARM and Silistix. Dr. Mikel Luján holds a Royal Society University Research Fellowship. Authors from the University of the Basque Country are supported by the Spanish Ministry of Education and Science, grant TIN2007-68023-C02-02, and by Basque Government grant IT-242-07. Dr. Javier Navaridas is supported by a post-doctoral grant of the University of the Basque Country.

8 REFERENCES

[1] K Asanovic, J Beck, J Feldman, N Morgan, J Wawrzynek. "A supercomputer for neural computation." In Proc. 1994 Intl. Conf. on Neural Networks.

[2] BlueBrain project. Available (September 2009) at: http://bluebrain.epfl.ch/.

[3] M. Blumrich, et al. "Design and Analysis of the BlueGene/L Torus Interconnection Network". IBM Research Report RC23025 Dec. 2003.

[4] P Dayan and L Abbott, "Theoretical Neuroscience". Cambridge: MIT Press, 2001.

[5] S.A. Knock, A.R. McIntosh, O. Sporns, R. Kotter, P. Hagmann, V.K. Jirsa, "The effects of physiologically plausible connectivity structure on local and global dynamics in large scale brain models" Journal of Neurosicence Methods, 183(1), pp 86-94, 2009.

[6] S Furber, S Temple, and A Brown, "On-chip and inter-chip networks for modelling large-scale neural systems," in Proc. International Symposium on Circuits and Systems, ISCAS-2006, Kos, Greece, May 2006.

[7] S Furber, S Temple, "Neural Systems Engineering". Journal of The Royal Society Interface 4(13), pp 193-206, April 2007

[8] H Hellmich, M Geike, P Griep, P Mahr, M Rafanelli, H Klar.. "Emulation engine for spiking neurons and adaptive synaptic weights". In Proc. IEEE International Joint Conference on Neural Networks (IJCNN), 2005.

[9] X. Jin, S.B. Furber, and J.V. Woods. "Efficient Modelling of Spiking Neural Networks on a Scalable Chip Multiprocessor". In Proc. of the International Joint Conference on Neural Networks, 2008.

[10] MM Khan, DR Lester, LA Plana, A Rast, X Jin, E Painkras and SB Furber. "SpiNNaker: Mapping Neural Networks onto a Massively-Parallel Chip Multiprocessor". Proc. 2008 International Joint Conference on Neural Networks (IJCNN2008).

[11] MM Khan, J Navaridas Palma, AD Rast, X Jin, LA Plana, M Luján, JV Woods, J Miguel-Alonso, and SB Furber. "Event-Driven Configuration of a Neural Network CMP System over a Homogeneous Interconnect Fabric". 8th Intl Symposium on Parallel and Distributed Computing. July 2009. Lisbon, Portugal.

[12] Microelectronics Division T.U. of Berlin. "Design and implementation of spiking neural networks". Available (September 2009) at: http://mikro.ee.tu-berlin.de/spinn.

[13] J Navaridas, M Luján, J Miguel-Alonso, LA Plana, SB Furber. "Understanding the Interconnection Network of SpiNNaker". 23rd International Conference on Supercomputing (ICS'09), June 8-12, 2009, York Town Heights, New York, USA.

[14] P Pfaerber and K Asanovic. "Parallel neural network training on multispert". In Proc. IEEE Third International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'97), 1997.

[15] LA Plana, SB Furber, S Temple, MM Khan, Y Shi, J Wu, and S Yang. "A GALS Infrastructure for a Massively Parallel Multiprocessor". IEEE Design & Test of Computers, Volume: 24 , Issue: 5, pp. 454 - 463, Sept.-Oct. 2007

[16] LA Plana, J Bainbridge, SB Furber, S Salisbury, Y Shi and J Wu. "An on-chip and inter-chip communications network for the spinnaker massively-parallel neural net simulator," in Proc. 2nd ACM/IEEE Intl. Symposium on Networks-on-Chip, 2008, pp. 215 – 216.

[17] FJ Ridruejo, J Miguel-Alonso. "INSEE: an Interconnection Network Simulation and Evaluation Environment". Lecture Notes in Computer Science, Volume 3648 / 2005 (Proc. Euro-Par 2005).

[18] P. Hagmann, L. Cammoun, X. Gigandet, R. Meuli, C.J. Honey, et al., "Mapping the Structural Core of Human Cerebral Cortex", PloS. Biol. 6(7), 2008.