Analytical Assessment of the Suitability of Multicast Communications for the SpiNNaker Neuromimetic System

Javier Navaridas [†], Mikel Luján [†], Luis A. Plana [†], Jose Miguel-Alonso [‡], Steve B. Furber [†]

[†] School of Computer Science
University of Manchester
M13 9PL, Manchester, UK.
e-mail: {javier.navaridas, mikel.lujan, plana, steve.furber}@manchester.ac.uk

[‡] Dept. of Computer Architecture and Technology
University of the Basque Country
P. Manuel de Lardizabal, 1
20018, San Sebastian, SPAIN
e-mail: j.miguel@ehu.es

***Abstract*—SpiNNaker is a custom-made architecture designed to model large-scale spiking neural nets. One of the most significant characteristics of neural nets is their extreme communication needs; each neuron propagates its activation to thousands of other neurons. This paper shows analytical proof that the novel *multicast* router in SpiNNaker is a better solution for simulating neural nets than more powerful *point-to-point* routers such as those found on datacentres or high performance computing systems, even when it has significantly lower requirements in terms of complexity, area and power. First, we characterised the utilization of resources required by both multicast and unicast networks. Then we derived the bandwidth needs of different communication architectures. Finally, we derived the amount of neurons the different networks can support. From these analyses we determined that the multicast communications adopted in SpiNNaker will be able to support the target application under the expected operation conditions.**

Keywords- Massively parallel processing, Multicast networks, Neuromimetic architecture, VLSI systems.

I. INTRODUCTION

The SpiNNaker system is a biologically-inspired massively parallel architecture designed with the aim of simulating very large-scale spiking neural networks in biological real-time with the special objective of providing low power consumption. Its design is based around bespoke multicore SoCs which are interconnected using a custom-made network. The largest configuration will house 64K nodes creating a system with over one million computing cores, which will be able to simulate spiking neural nets with more than 10^9 neurons (roughly the number of neurons in small primates' brains and 1% of a human's brain [16]). Neurons are modelled in software and their spikes generate packets that propagate through the on- and inter-chip communication fabric relying on bespoke *on-chip multicast routers*. Spike events are communicated to all connected neurons (typically in the order of thousands). The use of multicast routers helps alleviate the pressure exerted onto the interconnection network due to the high connectivity of the simulated neural models. This paper shows analytical evidence of its benefits measured both in terms of network bandwidth and number of supported neurons.

Currently, some SpiNNaker chips have been produced and successfully demonstrated running spiking neural network models [20]. SpiNNaker chips, due to their low-power design, can be used in embedded control systems such as robots, providing them with real-time stimulus-response behaviour [11, 12]. However, it is still unclear whether, once large-scale SpiNNaker systems are constructed, the communication needs of the application will prevent real-time operation with the predicted number of neurons per core. This paper provides some insights into when and how the system will be able to accommodate different application needs.

Our analysis begins with the derivation of the requirements in terms of network resources demanded by general unicast and multicast communications. From these results, we will derive the link bandwidth necessary to maintain the communication needs of neural nets for different router architectures. In this study we will consider multicast and unicast versions of the router implemented in SpiNNaker plus two state-of-the art routers used in datacentres and high performance computing systems (the SeaStar2 router in the Cray XT4 family of supercomputers [32] and the router in the Blue Gene/P family of supercomputers [2]). We close our study by deriving the number of neurons that those systems can support showing when the network may become a limiting factor.

The main conclusion of this study is that, thanks to the use of a multicast architecture, the communication infrastructure implemented in SpiNNaker can achieve the anticipated performance levels when simulating neural nets under normal operating conditions

II. OVERVIEW OF THE SPINNAKER SYSTEM

SpiNNaker is a massively parallel architecture composed of SpiNNaker chips arranged in a 2D triangular torus topology as depicted in Figure 1.

A. Supported Application

SpiNNaker is, in principle, designed to perform real-time simulation of spiking neural networks. It has been demonstrated, however, that it provides an architecture general enough to seamlessly execute a wide range of applications [30]: from the simulation of neural non-spiking models such as the Multilayer Perceptron [31] to a variety of applications completely unrelated to neural networks: discrete simulation, many-body interaction, real-time ray-tracing, finite element analysis and analogue circuit simulation.

Spiking neural systems have abundant parallelism and no explicit requirement of coherence as only local information is used by the neurons. The modelled application is in prin-

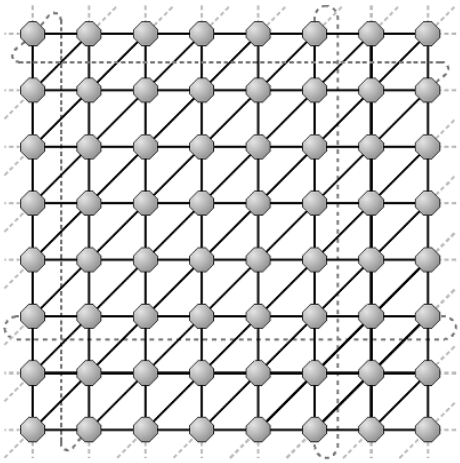

Figure 1. 8×8 SpiNNaker topology. Most peripheral wrap-around links are not shown for the sake of clarity.

ciple very simple [10]. Neurons have a membrane potential which is affected by each received signal. Whenever an excitatory signal is received the membrane potential is increased; in contrast, if the signal is inhibitory the membrane potential is reduced. After an excitatory signal, if the membrane potential exceeds a given threshold, the neuron discharges and fires a signal (a so-called *spike*). This signal propagates to all neurons sharing a synaptic connection, typically in the order of 10^3. The dynamics of the spiking neurons are emulated in SpiNNaker using two well-known models: Izhikevich [17] and Leaky Integrate and Fire [21]. In biological brains connected neurons have to come in close proximity, i.e. have a physical synapse. In SpiNNaker, however two connected neurons can be located in distant areas of the system.

Biological neurons work in a noisy environment [37] and, indeed, die during normal operation (adult humans lose about one neuron per second [8]). Thus their operation is neither perfect nor deterministic. In fact biological neural networks use *best-effort* communications in which spikes *try* to reach their intended destinations but, if this is not possible, the neural system continues its operation normally. The design of SpiNNaker reflects this behaviour. Neurons are modelled as event-driven applications executed by the processing cores. Spikes are represented by short network packets using Address-Event Representation (AER), a format widely used in neural network models [22, 23, 35]. Packets are multicast routed in hardware with the on-chip routers replicating them as necessary to reach all their destination cores through the interconnection network which is in charge of handling both intra- and inter-chip packets.

Given that digital electronics are orders of magnitude faster than the biological process – for example, biological spikes are propagated through an axon for up to 20 *milliseconds* while transmitting a packet through the SpiNNaker interconnection network should take a few *microseconds* at most – it is possible to multiplex many neurons onto a processor and many spikes onto a network.

B. System Architecture

Each SpiNNaker chip (in Figure 2) contains one multicore SoC with 18 low-power ARM968 cores. Each core has a tightly-coupled dedicated memory that can hold 32KB of instructions and 64KB of data. Each core runs an independent event-driven neural process with events generated by modules such as the timer, the vectored interrupt controller (VIC), the communication controller (CC) or the DMA controller. All the cores in a chip share an SDRAM of up to 128MB which is used to store synaptic information, among other things. Access to this shared storage space is carried out by means of a self-timed Network-on-Chip (NoC) which connects resources within the chip [28]. The router can be accessed through this NoC for configuration purposes, but during regular execution the ARM cores use the communication controller to send or receive packets. The NoC provides higher communication bandwidth (8 Gbps), lower contention and lower power consumption than any typical bus-based interconnect [29]. Detailed simulations of the chip using Verilog and SystemC showed that each *core* can model in biological real-time up to around *1000 individual neurons* [19]. This figure will be used in the analyses performed later in this paper.

The heart of SpiNNaker chip's communications is the novel multicast router that allows inter- and on-chip connectivity [14]. Its primary role is to direct neural event packets to those cores where their connected neurons are located. The router has 18 ports for internal use by the ARM cores and six ports to communicate with six adjacent chips (towards North, South, East, West, Northeast and Southwest). All ports are full-duplex and implement self-timed protocols.

To reduce area and power requirements, instead of implementing a central crossbar, the router is organized hierar-

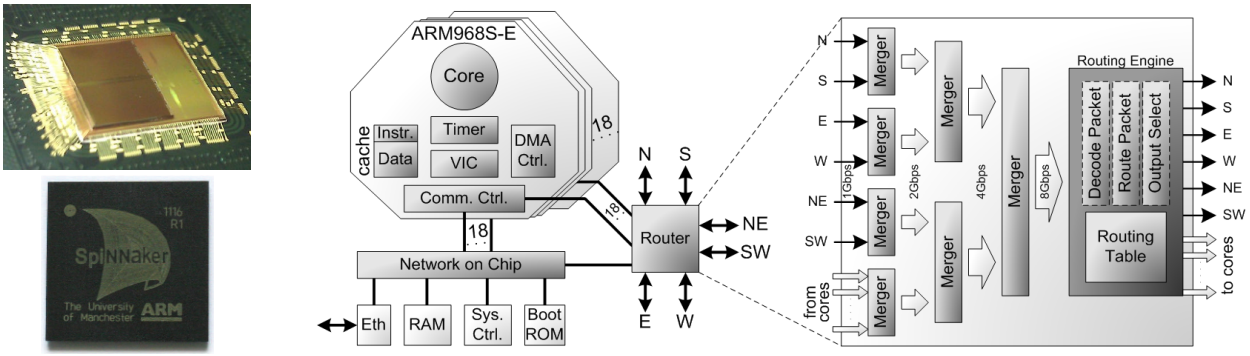

Figure 2. SpiNNaker chip before and after packaging (left) and schematic model of the SpiNNaker chip with its main components depicted (right).

chically: 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 it works faster than the transmission ports. Thus, 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 using small packets of 40 or 72 bits. Compared to a pure point-to-point alternative, the multicast engine helps reduce pressure at the injection ports, and reduces significantly the number of packets that traverse the network.

Following AER principles, routers make routing decisions based on the source address of a packet, i.e. the identifier of the firing neuron. In other words, a neural-event packet does not contain any information about its destination(s). The information necessary to deliver a neural packet to all the relevant cores and chips is compressed and distributed across the 1024-word routing tables within the routers. The routing tables have to be preloaded using application-specific information. To further compress communication data, routing tables offer a masked associative route look-up and the routers are designed to perform a default routing which needs no entry in the tables by sending the packet to the port opposite to the one the packet comes from. For example, if the packet comes from the North it will be sent to the South.

Network flow-control is very simple. When a packet arrives to the routing engine, 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 for a while and after a timeout the packet will be dropped. This has been demonstrated to be enough to guarantee deadlock-free communications [26]. 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 outdated, and it is dropped. The ages are global to the whole system and its time-span is arbitrary, a router configuration parameter. Emulating the behaviour of biological neural networks, dropped packets in SpiNNaker are not re-sent. Losing neurons or signals does not impede the normal functioning of the biological processes; nonetheless, packet dropping levels must be kept (*very*) low. A more detailed description of the features of the multicast router can be found in [39].

III. DEFINITIONS

In this section we discuss the notation used in the analytical study. The average distance from a source to a destination is defined as $\bar{d}$ and is measured in hops. Note that we consider random, uniformly distributed destinations, which means that the destinations for a given neuron do not tend to be clustered. This emulates a network configuration in which neuron placement has not been optimised, which provides a worst case scenario for the multicast router. The reader should note that if destinations were clustered, creating a multicast tree would be easier and would require less network resources. For this reason this study has to be considered as a worst-case analysis for the multicast router. In the following analyses, we will consider values of $\bar{d}$ ranging from 1 to 32 which cover reasonable configurations from very local to very distant.

The number of destinations of a multicast tree, i.e. the fan-out, is defined as F. Note that when discussing neural networks, the fan-out refers to the number of connected neurons. However in this study we will measure this figure in a slightly different way; given that several neurons can be multiplexed in the same node and that a single packet can be delivered to several neurons in the same node, it does not make sense, therefore, to send two packets to two neurons in the same node. For this reason we will consider a *node-based* fan-out and hence F will measure destination nodes per spike. Note that the actual relation between the neuron-based and node-based fan-out depends on several characteristics of the executed application mostly related to the communication patterns and the placement. A very pessimistic configuration may have every destination neuron in a different node, whereas an optimistic scenario may have most of the destination neurons in the same node. In the next section we will consider values of F ranging from 1 to 2048 and growing exponentially. This covers scenarios well beyond those expected with reasonable placements, but provides a wider view of the design space.

The number of outputs per node is identified as L (links/node) and will always be 6 as all the routers have this number of ports. n_n represents the number of neurons in each node (neurons/node). We will consider a thousand neurons running in each node – the limit imposed by the processing cores (18K neurons per node). f_s represents the frequency at which neurons generate a spike, on average, i.e. the firing rate (spikes/second). We will consider a value of f_s = 10Hz which is considered the upper limit for a very active population [10].

Finally p_s defines the packet size (bits) and the link bandwidth is represented as B_L (bits/sec). These two parameters depend on the underlying architecture. For the purpose of this paper we will consider five different configurations, three of them related to SpiNNaker and the remaining two related to state-of-the-art routers used in datacentres and high performance computing systems:

1. Worst-case multicast for the SpiNNaker router (B_L = 1Gbps, p_s = 40 bits).
2. Average case for the SpiNNaker router when using multicast.
3. Average case for the SpiNNaker router when using unicast. This configuration will show how the system would behave if a unicast approach had been implemented.
4. Average unicast for the router in the Blue Gene/P supercomputer (B_L = 3.2Gbps, p_s = 64 bits) [2]
5. Average unicast for the SeaStar2 router on Cray's XT4 (B_L = 6Gbps, p_s = 64 bits) [32].

The later two routers are beyond the design constraints of SpiNNaker both in terms of power consumption and chip area, and hence cannot be implemented in the actual system. However, from their analyses we will see how a low-power multicast design can draw reasonable performance levels when compared with unicast-based high-performance counterparts.

All the routers considered in this paper have six output ports and therefore can be arranged using the same topology. In this paper we will consider the SpiNNaker topology, but the results for this topology can be extrapolated to other mesh-like topologies such as the three-dimensional tori in both Blue Gene and Cray XT families.

IV. Performance Analysis

This section is devoted to analyze the performance of the different configurations. We will start by deriving the network requirement of general unicast and multicast approaches. After that we will derive the link bandwidth required to support the execution of an application and instantiate it in each of the router configurations. Finally we will show the amount of neurons that can be supported by each configuration to show when the network may become a limiting factor.

A. Network Resources Utilization

We have formulated the network requirements of unicast and multicast approaches to deliver the neural communications to be executed on top of SpiNNaker. The first figure to derive is the amount of network resources employed in order to communicate a spike to all connected neurons. We will represent this figure as N which is measured as the total number of traversed links (hops) and will be derived from $\overline{d}$ and F, which depend solely on the workload.

In the case of unicast, deriving the average network requirement from these two parameters is straightforward; F packets travelling an average distance of $\overline{d}$. Hence, given that unicast routes are not shared it will be:

$$N_u = \overline{d} \cdot N \tag{1}$$

In the case of multicast, it is not easy to derive an expression for the average network requirements because it strongly depends on the distribution strategy. We will derive, instead, an upper bound estimator. This estimator considers the worst configuration; this is when routes to the destinations share as few as possible of their paths. The estimator for the upper boundary can be formulated as follows:

$$N_M = \sum_{i=1}^{i=2\cdot\overline{d}-1} \min\left(6\cdot i, \frac{F}{2}, 2\,6\cdot\left(\overline{d}-i\right)\right) \tag{2}$$

Figure 3 shows N_u and N_M for the values of interest of the parameters $\overline{d}$ and F. For the sake of completeness we also show N_m, the empirical average network utilization of 10^5 random runs using a simple multicast route generation algorithm. In the figure it is clear that N_M is always lower than N_u except for very small numbers of destinations which are equal. As the fan-out and the average distance grow, the differences between unicast and multicast increase. In the most extreme configurations the *average unicast* requires over 10 times more network resources than the *worst multicast* case and over 25 times more resources than the average multicast case. More than *one order of magnitude* in terms of required network resources is a considerable difference which justifies the implementation of a multicast communications infrastructure, such as the one in SpiNNaker.

To sum up, the main conclusion of this first experiment is that implementing a multicast scheme for simulating neural networks provides substantial savings in terms of network utilization. For this reason we encourage other research

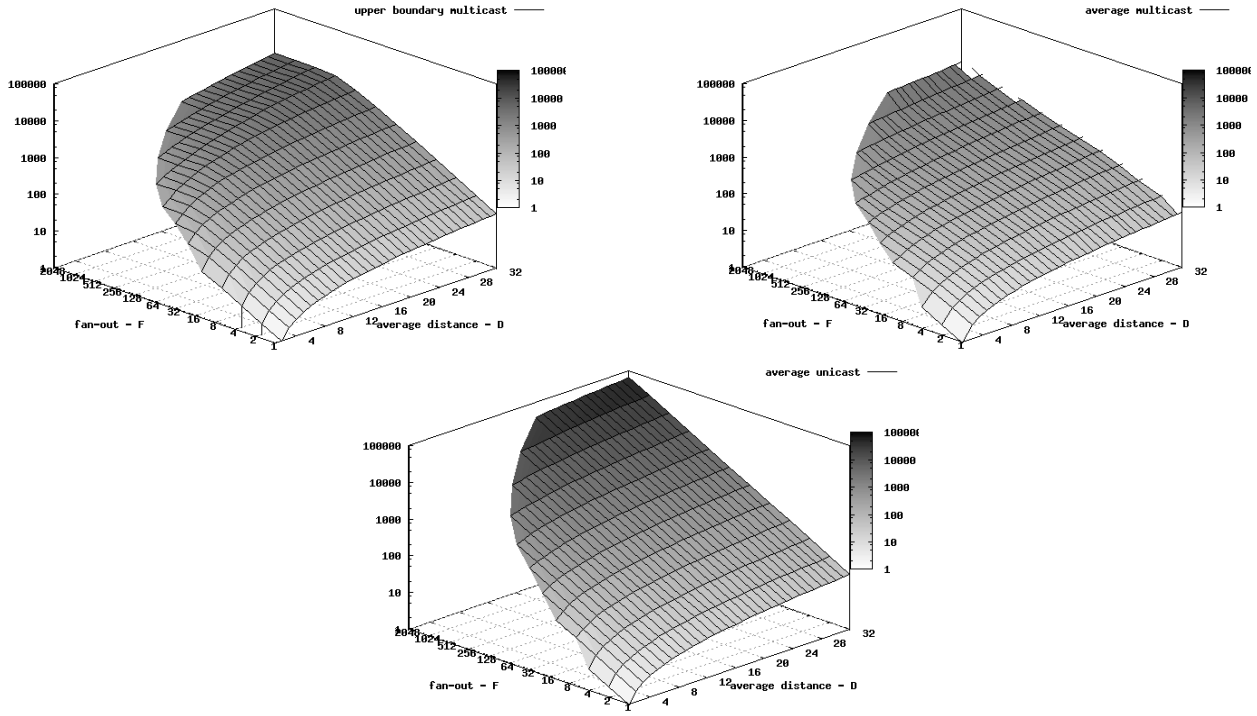

Figure 3. Required network resources. a) Multicast upper boundary - N_M. b) Average multicast - N_m (10^5 runs – ESPR). c) Average unicast – N_u.

groups in the area to not disregard multicast architectures when designing such systems.

B. Link Bandwidth

We will derive next the network bandwidth required to support a given system configuration. In the formulation below we assume that the traffic is distributed evenly among all the links of the network, which may not be the case. However as the unbalanced use of the network would affect both the unicast and the multicast to a similar extent, it is fair and safe to make such an assumption. The required link bandwidth can be computed as the total number of injected packets times the packet size times the network resources required by each configuration divided by the number of links:

$$B_L = \frac{N \cdot f_s \cdot n_n \cdot p_s}{L} \quad (3)$$

The network bandwidth required to support the neural simulation using the different system configurations and routers is plotted in Figure 4. Note how the bandwidth requirement when using any unicast router can be over *one order of magnitude* higher than the worst case for a multicast router and roughly *two orders of magnitude* when compared with the average case of multicast. In the worst cases, the unicast systems would require a sustained link bandwidth of over 100 Gbps; 100 times more than the link bandwidth provided by SpiNNaker and around 20 times the link bandwidth provided by the high performance routers considered here. The multicast results show that the maximum link bandwidth required by an extremely poorly configured application (with no exploitation of locality) could be 3 to 5 times that provided by SpiNNaker. This second experiment throws similar insights to the previous one: a multicast architecture seems to be a more sensible design than a unicast one, even if the multicast architecture has *a priori* poorer characteristics.

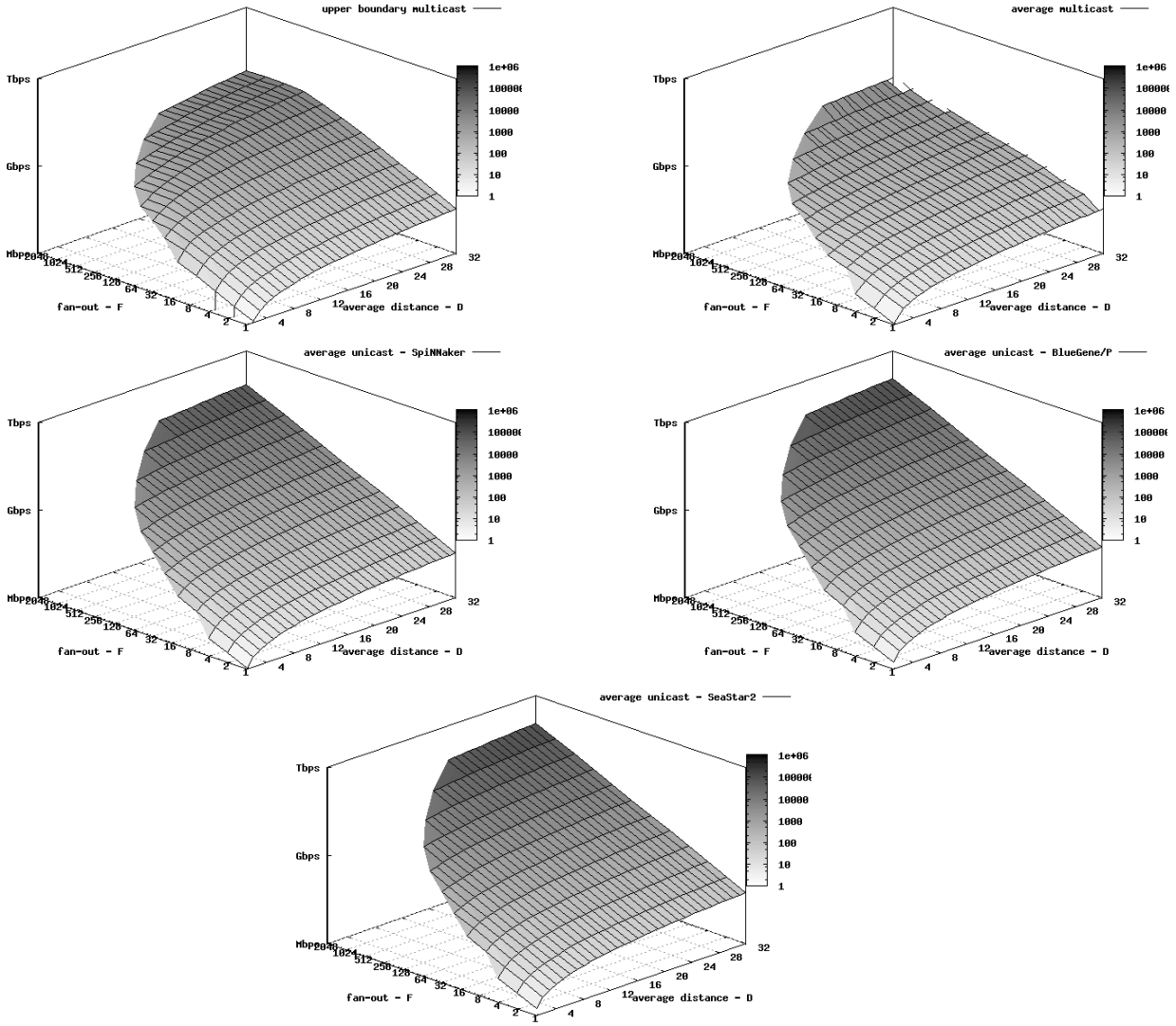

Figure 4. Network bandwidth (B_L) required to support regular operation of SpiNNaker (18000 neurons firing at 10 Hz).a) Multicast upper boundary. b) Average multicast (10^5 runs). c) Average unicast SpiNNaker router.d) Average unicast Blue Gene/P router. e) Average unicast SeaStar2 router

C. Supported Neurons

Until now we have analyzed system-level properties such as network resources or network bandwidth. However, these properties may, and indeed probably do, mean nothing to the average user of SpiNNaker, whose expertise is in a very different area of research. For this reason, we will close our analysis by deriving an important figure of merit: the number of neurons per node that the system can support during regular operation. This figure can be computed as the per-node *available* bandwidth divided by the per-neuron *required* bandwidth:

$$n_n = \frac{B_L \cdot L}{N \cdot f_s \cdot p_s} \tag{4}$$

From (1), (2), (4) and the router dependent values, we can compute the average number of neurons that each system is able to execute per node. To be able to see when the network becomes a limiting factor, we will use the limit imposed by the cores (18K) as an upper bound for the number of simulatable neurons, see Figure 5. In the plots it is evident that, in those configurations with non-demanding network utilization (low number of destinations and/or low distances) all networks can support the execution of a regular neural network simulation (*plateaus* in the plots). However as these two figures increase the networks start to become the limiting factor (*slopes*). This happens more often with the unicast routers, even when the Blue Gene and SeaStar2 have noticeably higher bandwidths than SpiNNaker's multicast router. Furthermore, in those cases in which the network becomes the limiting factor, the multicast alternative is able to support a larger number of neurons; in the worst case unicast routers fall to several hundreds of neurons, whereas multicast is able to support thousands of them, roughly one order of magnitude difference.

Reducing the number of neurons simulated in each node is one way to allow a proper operation of the system. An alternative way would be to execute the application slower

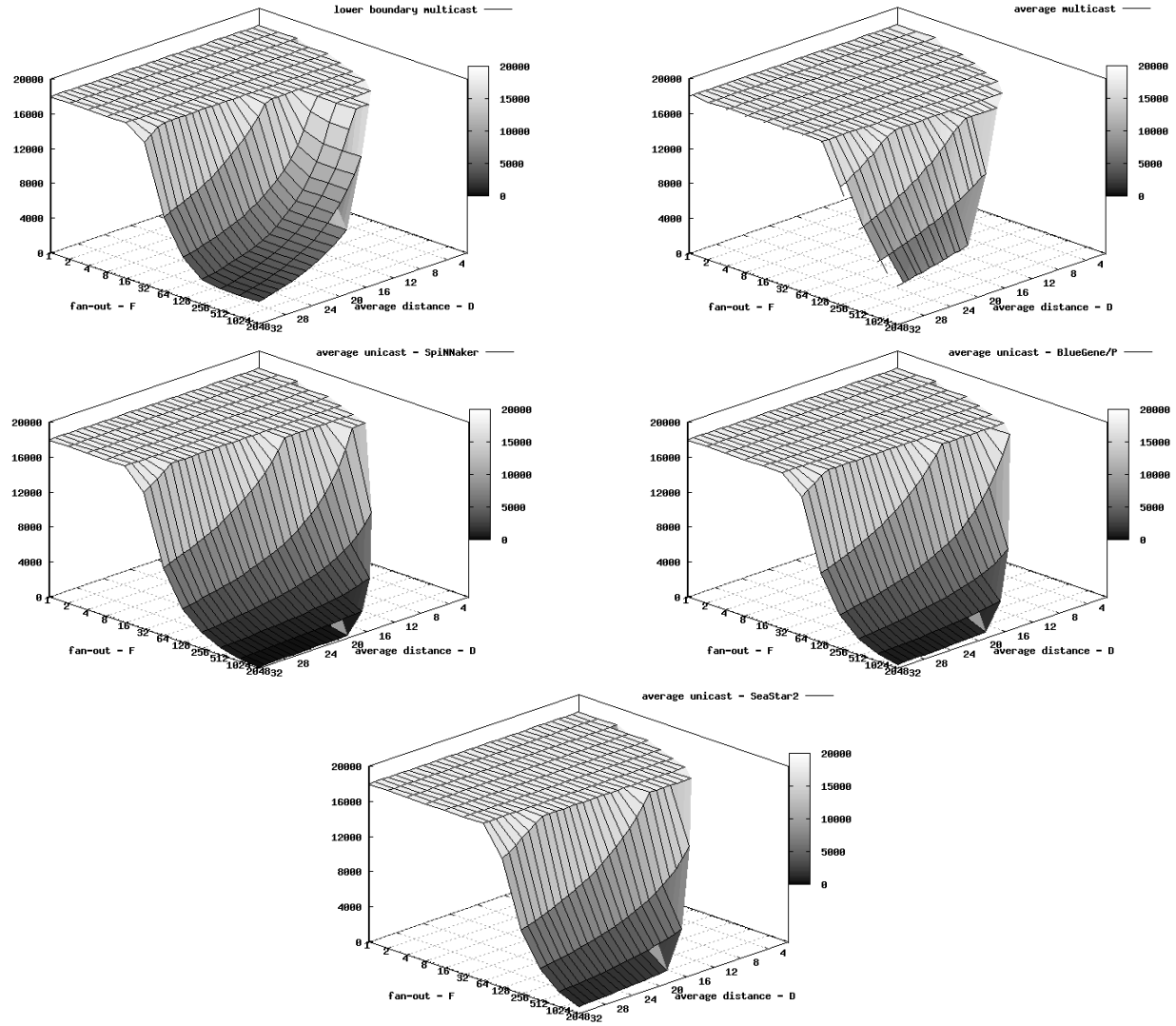

Figure 5. Number of neurons supported by the system, considering the limit imposed by the cores (18000 neurons). a) Multicast lower boundary. b) Average multicast (10^5 runs ESPR). c) Average unicast SpiNNaker router. d) Average unicast Blue Gene/P router. e) Average unicast SeaStar2 router

than real time. For example, in the worst case, the multicast architecture could simulate around 2,000 neurons per node in real-time, but it could alternatively simulate 18,000 neurons per node if executed 9 times slower than real-time. In the case of the unicast architectures, going from a few hundred neurons per node to 18,000 would require slowing down the execution by up to 30-50 times slower than real-time. The final decision on whether reducing the number of neurons per node or slowing execution ultimately depends on the purpose of the simulation. For simulating very large-scale neural networks it would be preferable to slow execution, whereas for an embedded control system as those discussed before, keeping real-time and reducing the number of neurons per node seems a more reasonable decision.

At any rate, in the real implementation we do not expect fan-outs and distances as extreme as the worst cases reported here. Hence, we can anticipate that large-scale SpiNNaker systems will be able to operate under the assumed functioning conditions.

V. Related Work

Research in simulating biologically plausible neural networks (brain-like systems) is not new and has remained a *hot topic* for the last decades. In the early nineties a team at U.C. Berkeley worked on the Connectionist Network Supercomputer [4]. 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 2D mesh, with a target size of 128 nodes (further scalable to 512). Each node would incorporate a general-purpose RISC processor plus a vector coprocessor, 16MB of RAM and a router. The router, however, did not support multicast communication, as the system was expected to rely on spatial locality to improve performance. 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 [27].

More recently, the Microelectronics Division at the T.U. of Berlin worked in a research project with objectives similar to those of SpiNNaker [25]. Part of this project was an acceleration board, called SSE, implemented with a collection of FPGAs interconnected via an on-board bus. An SEE accelerator was able to perform neural computations 30 times faster than a desktop PC [15]. Other projects used FPGAs for similar purposes, obtaining speedups of up to 50 compared to software-only implementations. Nevertheless, as these boards could not be connected to form a network, they did not have support for multicast.

As far as we know, there are only three active projects comparable to SpiNNaker in terms of simulation scale. First, the Blue Brain project [6] aims to create biologically accurate functional models of the brain; however, model complexity (far more intricate than SpiNNaker's) only allows real-time execution of roughly a neuron per node [24]. This is a low figure in comparison with the several thousand (simpler) neurons per node supported by SpiNNaker.

Secondly, the DARPA's System of Neuromorphic Adaptive Plastic Scalable Electronics (SyNAPSE) project claims that it has achieved the simulation of spiking neural networks the size of a cat's brain (10^9 neurons) using Izhikevich models like those supported by SpiNNaker [3]. Their simulations run 2-3 orders of magnitude slower than real-time.

In contrast with the biologically-inspired SpiNNaker architecture, neither Blue Brain nor SyNAPSE contemplate the construction of a custom architecture but use general-purpose supercomputers from the IBM Blue Gene family. Blue Gene systems offer native support for broadcast and multicast communications [1, 2] using a dedicated tree-like network which may not be adequate for the kind of application supported by SpiNNaker. As discussed, Blue Gene supercomputers or others, such as the Cray XT supercomputers [32], being general-purpose will provide solutions that do not match the power-efficiency of SpiNNaker.

Last but not least, the FACETS project [13] is attempting to create a faster than real-time hardware system for the simulation of networks of large but unspecified size. This architecture, while biologically inspired, uses a fixed synapse and neuron model and, therefore, is not a system as general as SpiNNaker. It employs analogue circuits to implement most of the central dynamic functions. The FACETS architecture uses wafer-scale devices [33] to achieve the necessary connectivity. It uses AER signalling (similar to SpiNNaker), but with a circuit-switched, synchronous communications subsystem which has no support for multicast. Thus the FACETS system and its associated HICANN devices once again represent a very different system designed to solve a different problem: faster than real-time neural simulation, for which power consumption is not a factor and communications merely a side effect rather than a design feature.

Regarding multicast communications, to our knowledge most of its research is on the areas of communication networks (IP networks) and multistage networks for high performance computing systems, neither of which can be applied to SpiNNaker. In IP networks, the network is completely irregular, thus topology discovery becomes a very important area of research [7, 34]. In multistage networks, the construction of multicast routes focuses on minimizing the interactions between concurrent multicast communications to increase network throughput [5, 9]. Finally, research into multicast communications for mesh-like topologies such as the one in SpiNNaker is scarce [18, 38]. The reason for this is that their main use is the implementation of collective operations, which can be easily and efficiently deployed in software over a point-to-point architecture [36]. This software-based approach of collective operations reduces the complexity of implementation and, for this reason it is more commonly adopted.

VI. Conclusions And Future Work

In this paper we have highlighted the importance of multicast routing for the kind of application executed on top of SpiNNaker in which each neuron must communicate its activation to thousands of other neurons. Our analysis shows that a multicast approach requires noticeably less network bandwidth than a unicast approach and therefore it is able to support the simulation of a larger number of neurons. More specifically we have shown how the multicast router imple-

mented in SpiNNaker can achieve better performance than more complex router designs such as Blue Gene's router or the SeaStar2 router, even though they provide noticeably higher network bandwidth.

As future work, we plan to explore different methodologies for the generation of multicast routes as it is an important, *non-trivial* aspect of using a multicast architecture. Our research will include the use of optimization-based approaches both to allocate neurons and to generate multicast routes. Optimizing the allocation of neurons will help to reduce the node-based fan-out and also the distance among communicating nodes. Optimizing route generation will provide more balanced use of network resources. These methodologies can become of exceptional importance once the large-scale SpiNNaker system is constructed.

ACKNOWLEDGMENT

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. Dr. Navaridas holds a Royal Society Newton International Fellowship. Dr Luján is supported by a Royal Society University Research Fellowship. Prof. Miguel-Alonso is supported by the Spanish Ministry of Education and Science, grant TIN2010-14931, and by Basque Government grant IT-242-07.

REFERENCES

[1] NR Adiga et al. "An Overview of the BlueGene/L Supercomputer". ACM/IEEE Supercomputing conference. 16-22 Nov 2002.

[2] S. Alam, et al. "Early evaluation of IBM BlueGene/P". ACM/IEEE Conference on Supercomputing, pp. 1-12, 15-21 Nov. 2008

[3] R Ananthanarayanan, et al. "The cat is out of the bag: cortical simulations with 10^9 neurons, 10^{13} synapses". ACM/IEEE Conference on Supercomputing. 2009. New York, NY, USA

[4] K Asanovic, et al. "A supercomputer for neural computation." Proc. 1994 Intl. Conf. on Neural Networks (ICNN94).

[5] S Bhattacharya, et al. "Multicasting in Generalized Multistage Interconnection Networks". Journal of Parallel and Distributed Computing 22(1) 1994, pp. 80-95. DOI: 10.1006/jpdc.1994.1071

[6] BlueBrain project. Available (Nov. 2011) at: http://bluebrain.epfl.ch/

[7] Y Breitbart, et al. "Topology discovery in heterogeneous IP networks: the NetInventory system". IEEE/ACM Transactions on Networking 12(3), 2004, pp. 401-414. DOI: 10.1109/TNET.2004.828963

[8] H Brody. "Cell counts in cerebral cortex and brainstem". Alzheimer Disease Senile Dementia and Related Disorders. 1978. pp. 345 – 351.

[9] S Coll, et al. "Efficient and Scalable Hardware-Based Multicast in Fat-Tree Networks". Trans. on Parallel and Distributed Systems 20(9)

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

[11] S Davies, et al. "Interfacing Real-Time Spiking I/O with the SpiNNaker neuromimetic architecture". Australian Journal of Intelligent Information Processing Systems 11(1), 2011, pp. 7-11.

[12] T Elliott, N Shadbolt, "Developmental robotics: Manifesto and application," Philosophical Trans. Royal Soc., vol. A, no. 361, 2003.

[13] J Fieres, J Schemmel, K Meier. "Realizing biological spiking neural network models in a configurable wafer-scale hardware system". Int'l Joint Conf. on Neural Networks. 2009. pp. 969–976

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

[15] HH Hellmich, et al."Emulation engine for spiking neurons and adaptive synaptic weights". In Proc. IEEE International Joint Conference on Neural Networks (IJCNN), 2005.

[16] S. Herculano-Houzel, "The human brain in numbers: a linearly scaled-up primate brain", Frontiers of Human Neuroscience 3.

[17] E Izhikevich. "Simple model of spiking neurons". IEEE Trans. on Neural Networks 14, (2003) pp. 1569–1572.

[18] NE Jerger, LS Peh ,M Lipasti. "Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support". Proceedings of the 35th Annual International Symposium on Computer Architecture

[19] X Jin, SB Furber, and JV Woods. "Efficient Modelling of Spiking Neural Networks on a Scalable Chip Multiprocessor". In Proc. of the International Joint Conference on Neural Networks, 2008.

[20] X Jin, et al. "Modelling Spiking Neural Networks on SpiNNaker". IEEE Computing in Science & Engineering 12(5), 2010, pp 91-97.

[21] C Koch, I Segev. "Methods in Neuronal Modeling". The MIT Press

[22] W Maass, CM Bishop. "Pulsed Neural Networks". The MIT Press.

[23] M Mahowald. "VLSI Analogs of Neuronal Visual Processing: a Synthesis of Form and Function". PhD Dissertation (1992), California Institute of Technology, Pasadena, CA.

[24] H Markram. "The Blue Brain Project". Nature Revs. Neuroscience 7, Feb. 2006, pp. 153-160. DOI: 10.1038/nrn1848

[25] Microelectronics Division T.U. of Berlin. "Design and implementation of spiking neural networks." Available (Feb. 2012) at: http://mikro.ee.tuberlin.de/spinn.

[26] J Navaridas, et al. Understanding the Interconnection Network of SpiNNaker. 23rd International Conference on Supercomputing (ICS'09), June 8 to 12, 2009, York Town Heights, New York, USA.

[27] P Pfaerber and K Asanovic. "Parallel neural network training on multispert". IEEE 3rd International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'97), 1997.

[28] LA Plana et al. "A GALS Infrastructure for a Massively Parallel Multiprocessor". IEEE Design & Test of Computers, Volume: 24 , Issue: 5, pp. 454 - 463, Sept.-Oct. 2007

[29] LA Plana et al. "An on-chip and inter-chip communications network for the spinnaker massively-parallel neural net simulator". Intl. Symposium on Networks-on-Chip (NoCS 2008), 2008, pp. 215 - 216

[30] AD Rast, et al. "Managing burstiness and scalability in event-driven models on the SpiNNaker neuromimetic system". International Journal of Parallel Programming (2011). In Press.

[31] F Rosenblatt. "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms". Washington, Spartan Books

[32] AR Sadaf, et al. "Cray XT4: an early evaluation for petascale scientific simulation". Proc. of the 2007 ACM/IEEE Conference on Supercomputing, pp. 1-12, 10-16 Nov. 2007, Reno, NV, USA.

[33] J Schemmel, J Fieres, K Meier. "Wafer-scale integration of analog neural networks". IEEE International Joint Conference on Neural Networks, 2008. pp. 431 – 438.

[34] JK Shapiro, et al. "Topology discovery service for router-assisted multicast transport". Procs. of IEEE Open Architectures and Network Programming Proceedings, 2002, pp. 14-24 .

[35] MA Sivilotti. "Wiring Considerations in Analog VLSI Systems, with Application to Field- Programmable Networks (VLSI)". California Institute of Technology, Pasadena, CA, Ph.D. thesis (1991)

[36] R. Thakur and W. Gropp, "Improving the Performance of Collective Operations in MPICH",

[37] P Tiesinga, T Sejnowski. "Precision of pulse-coupled networks of integrate-and-fire neurons." Network 12 (2). 2001. pp. 215–33.

[38] L Wang, et al. "Recursive partitioning multicast: A bandwidth-efficient routing for Networks-on-Chip". International Symposium on Networks-on-Chip, 2009.

[39] J Wu, S Furber. "A Multicast Routing Scheme for a Universal Spiking Neural Network Architecture". The Computer Journal 53(3)