Analysis of Multicast Traffic Induced by Spiking Neural Networks in the ANA System

Anonymous Author
Anonymous Institution

Abstract

The *Anonymous Neuromimetic Architecture* is a massively parallel system devised to simulate very large-scale spiking neural networks in biological real-time. Its largest configuration incorporates up to 64K bespoke multi-core System-on-Chips which are interconnected using a two-dimensional triangular torus network. Neural events (spikes) are modeled as packets that propagate through this fabric relying on a novel on-chip *multicast* router. This router has very modest characteristics when compared with off-chip routers found on datacenters. However, we confirm analytically that this multicast router is a *better solution* for the target application than more powerful point-to-point routers, such as Cray's SeaStar2 or those in IBM's Blue Gene/P. Our analysis shows that the resource savings obtained by the use of multicast more than compensate a lower bandwidth. Furthermore we provide and evaluate four different strategies to generate multicast trees, each of them providing different features and drawbacks.

1 Introduction

The Anonymous Neuromimetic Architecture system – hereafter, ANA – is a biologically-inspired massively parallel architecture based on custom-made multicore System-on-Chip (SoC). ANA is designed with the aim of simulating very large-scale spiking neural networks (up to 10^9 neurons, roughly a 10% of the neurons in the human cortex) in biological real-time. Currently, some *test* chips with two cores and a fully functional router have been produced and successfully demonstrated running spiking neural models [1]. ANA chips, due to their low-power design can be used as control systems for robots providing them with *real-time stimulus-response* behavior [2, 3].

Biological spiking neural networks communicate by means of spike events which occur when a neuron is stimulated beyond a given threshold and fires. Spike signals are communicated to all connected neurons, with typical *fan-outs* of the order of 1000 [4]. These applications exhibit abundant parallelism, no explicit requirement to maintain coherent shared memory and are naturally resilient to failures; neurons may die (1 per second in the human brain [5]) and spikes may be missed, but the brain continues working properly. ANA takes advantage of these characteristics to deploy a well-balanced, low-power massively parallel architecture. The largest configuration houses 64K SoCs – creating a system with over 10^6 computing cores – which are interconnected using a two-dimensional triangular torus (see Fig. 1). Neurons are modeled in software and their spikes generate packets that propagate through the on- and inter-chip communication fabric relying on custom-made on-chip *multicast* routers. The use of multicast routers alleviates the pressure suffered by the interconnection network because of the high connectivity of the simulated neural models. This paper shows analytical proof of these benefits measured both in terms of network bandwidth and number of neurons that can be supported by the system. However, constructing multicast trees from a source to a set of destinations is not *trivial*, and for this reason we describe sev-

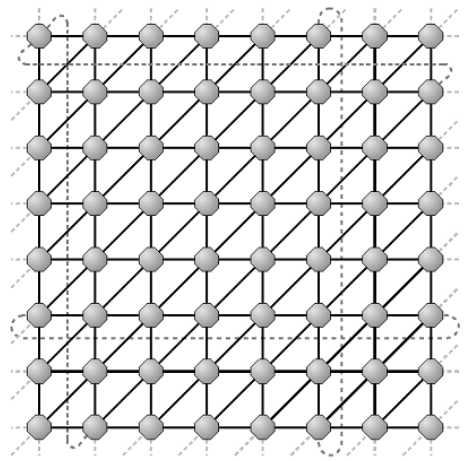


Figure 1. 8×8 triangular torus topology. Most peripheral wrap-around links not showed.

eral strategies to generate multicast trees. We also compare them considering their resource requirements.

The remaining of this paper is organized as follows. Section 2 describes the main characteristics of the system. Section 3 is devoted to define terminology and describe the four multicast strategies. Section 4 compares the proposed strategies. Section 5 provides an analytical evaluation of unicast and multicast alternatives, showing the benefits of using multicast. Finally, Section 6 closes the paper with some concluding remarks.

2 System architecture

Each ANA SoC (depicted in Fig. 2) contains 18 low-power ARM968 cores running an independent *event-driven* neural process. Events are generated by different modules: timer, vector interrupt controller (VIC), communication controller and DMA controller. Detailed simulations of the chip confirmed that each core can model in biological real-time up to around 1000 neurons [6]. The chip includes a SDRAM of 128MB mainly used to store synaptic information. Resources within the chip are connected by a custom-made self-timed Network-on-Chip (NoC) [7].

Each chip incorporates a router [8] whose primary role is to direct neural event packets to those chips and cores containing destination neurons. The router has 18 ports for the cores and six ports to communicate with six adjacent chips. Ports are hierarchically merged in three stages before using the actual routing engine. The router is able to handle a single packet at once, but works 8 times faster than transmission ports and is not expected to become a bottleneck – most of the time routers will be idle, and router delay will barely affect the pace at which packets are processed. The router supports point-to-point and multicast communications using small packets of 40 bits. We will see later how the multicast engine

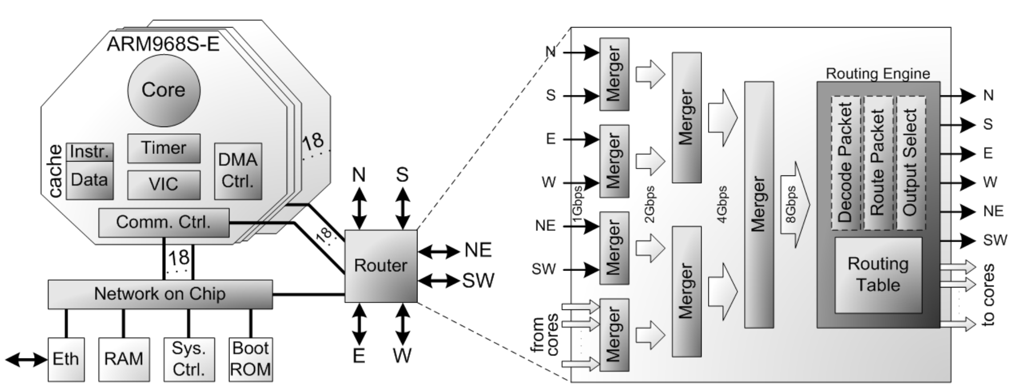


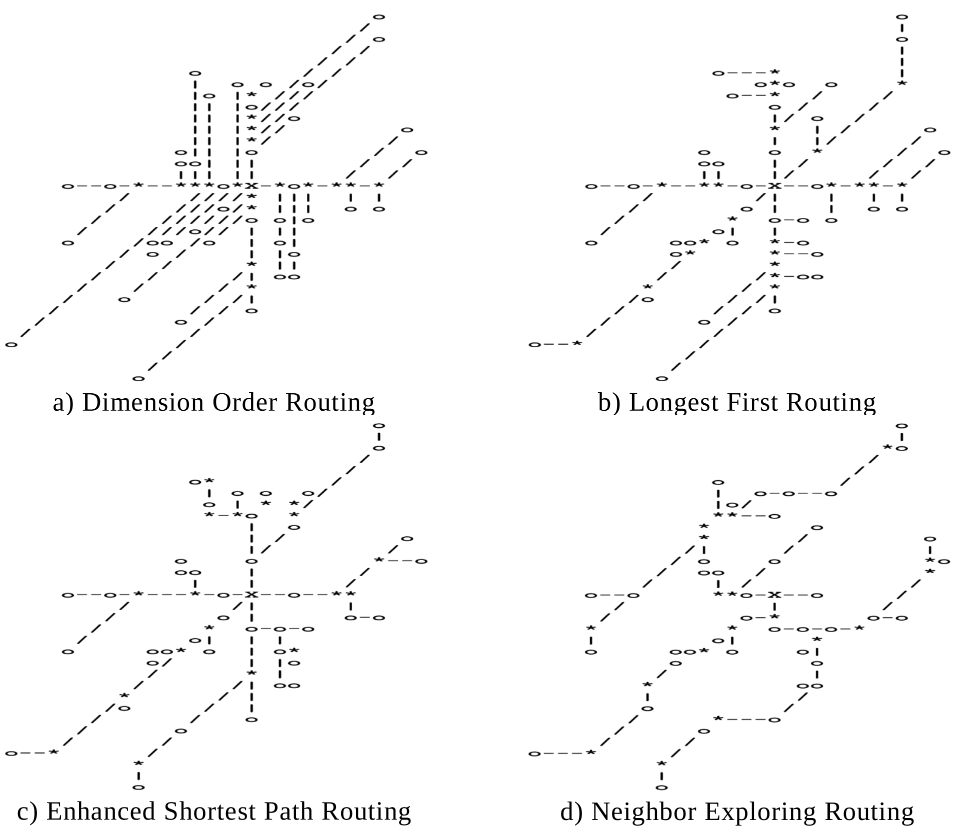
Figure 2. Schematic model of an ANA chip with all its components depicted.

reduces pressure at the injection ports, and also the number of packets traversing the network, when compared to a point-to-point alternative. Following an Address Event Representation [9] a neural-event packet does not contain any information about its destination(s), only the neuron that has fired. The information necessary to deliver a neural packet to all the relevant cores and chips is compressed and distributed across the 1024-word routing table within each router. To allow further compression, routing tables offer a *masked associative* route look-up and routers are designed to perform a *default routing* – which requires no entry in the routing table – by sending the packet to the port opposite to the one the packet comes from, i.e. if a packet comes from the North it will be sent to the South.

Network *flow-control* is very simple. When a packet arrives to the router, one or more output ports are selected and the packet is transmitted 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 to avoid deadlock. Similarly, to avoid livelock situations, packets wandering around the network for too much time are considered outdated and dropped. Emulating the behavior of biological neural networks, dropped packets in ANA are not re-sent. As discussed, losing neurons or signals does not impede the normal operation of the biological process; however dropping level must be kept (very) low.

3 Definitions

Throughout this paper we will use the following conventions to represent the variables in our study. The average distance between a source and its destinations is defined as $\bar{d}$ (measured in hops). The fan-out – number of destinations – is defined as F (nodes/spike). The link bandwidth is represented as B_L (bits/sec). The number of links per node is identified as L (links/node). n_n represents the number of neurons running in each node (neurons/node). f_s represents the frequency at which neurons generate spikes (spikes/second). p_s defines the packet size (bits).



a) Dimension Order Routing

b) Longest First Routing

c) Enhanced Shortest Path Routing

d) Neighbor Exploring Routing

Figure 3. Example of multicat trees. *legend*: ‘x’ source, ‘o’ destinations, ‘*’ a entry is required in routing table, ‘-’ ‘|’ ‘/’ default route (no entry)

Next we describe four strategies to generate multicast paths. We did not find any previous strategy in the literature that can be compared with the proposed strategies as multicast tree generation is used often with rather different objectives (avoid blocking in multistage networks [10, 11] and topology discovery in IP networks [12, 13]). In the algorithmic definitions, all the functions have comprehensible names and will not be discussed unless required for a proper explanation of the algorithm. In all cases, `source` represents the node for which the multicast tree is being generated, `dests` contains the destination set and `route` stores the tree under construction. The operation of all the proposed algorithms is similar; they start with an empty `route` and build the multicast tree by adding links as required. Examples of routing trees generated with each strategy are shown in Fig. 3.

Dimension Order Routing (**DOR**) is the simplest way of generate a multicast tree: follow an oblivious dimension order routing from the source to each destination (first move in X, then in Y and then in the diagonal). Note that only shortest paths are considered and that most communications will share the hops in dimension X or Y. An algorithmic definition of this strategy is shown in Algorithm I. The `dor` function returns the collection of links crossed when applying DOR from `source` to `d`.

Longest First Routing (**LFR**) is also oblivious using shortest paths. In this case, the routes are generated travelling first in the dimension with more hops. The definition of this strategy (Algorithm II) is similar to DOR's, but using a different routing function, `lf` which returns the links traversed using longest first routing.

Enhanced Shortest Path Routing (**ESPR**) is also based on shortest path routes but is noticeably more complex. Each destination, `d`, looks for the closest node, `c`, that it can reach using a shortest path towards the `source` and adds the route from `c` to `d` (using longest first routing) to the solution. If there is no node, then `c` will be the `source`. ESPR creates trees similar to those produced by LFR but avoids creating parallel branches when several nodes are close from each other. Algorithm III shows the pseudo-code definition of this strategy.

Finally, Neighbor Exploring Routing (**NER**) is the most complex of the presented strategies. The destinations are sorted from the closest to the furthest. Then each destination, `d`, looks for the closest route point, `p`, (the `source`, other destination or a used link) in its surroundings and adds the route between `d` and `p` to the solution. This way, as each node is connected to its closest route point, the requirements in terms of network resources are drastically reduced. Algorithm IV shows the pseudo-code definition of this strategy. Function `sort` arranges the node-list in destinations by their distance to source.

```
route=Ø
for d in dests
 r = dor(source, d)
 for l in r
  if l ∉ route
   add(l, route)
return route
```

Algorithm I. DOR strategy pseudo code

```
route=Ø
for d in dests
 r = lf(source, d)
 for l in r
  if l ∉ route
   add(l, route)
return route
```

Algorithm II. LFR strategy pseudo code

```
route=Ø
for d in dests
 c = closest(source, d)
 r = lf(c, d)
 for l in r
  if l ∉ route
   add(l, route)
return route
```

Algorithm III. ESPR strategy pseudo code

```
route=Ø
sort(dests, source)
for d in dests
 p = surroundings(d)
 r = lf(p, d)
 for l in r
  if l ∉ route
   add(l, route)
return route
```

Algorithm IV. NER strategy pseudo code

4 Evaluation of multicast strategies

We have compared the four discussed strategies bearing in mind their resource requirements. The figures of merit in the study are: (i) use of network resources, (ii) balanced use of resources, (iii) entries in the routing tables and (iv) algorithm complexity – i.e. computing time required to construct the trees. Fig. 4 shows the average resources utilization of 10^5 runs for a given configuration of $\bar{d} = 32$ and $F = 256$. Looking at the figure, it is clear that each strategy offers different characteristics.

DOR requires a reduced number of entries in the routing tables and allows constructing multicast trees very quickly but, in turn, employs lots of network resources. Another negative aspect of DOR is that the utilization of the different dimensions of the topology is very un-

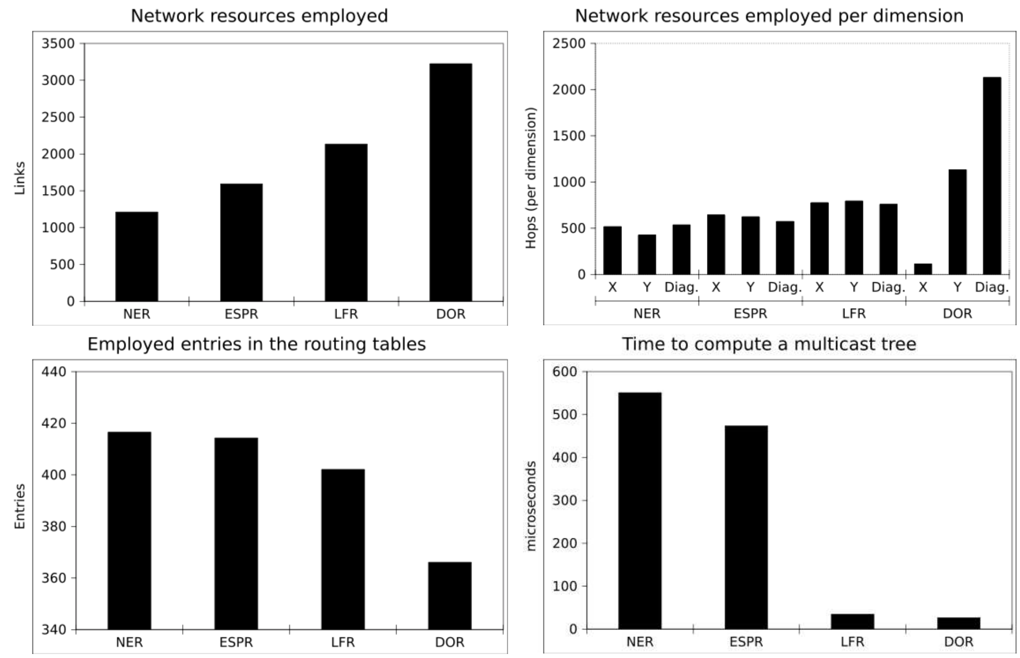


Figure 5. Per strategy resource requirements. $\bar{d}=32$, $F=256$ (average of 10^5 runs).

balanced: the diagonal may become a communication bottleneck. These two negative characteristics motivate the rejection of this strategy except in those cases in which reducing the number of entries in the routing tables is critical.

LFR is almost as fast as DOR, but drastically reduces the requirements in terms of network resources. This reduction is obtained at the cost of increasing the number of entries in the routing tables. However the main advantage of LFR is that it produces the most balanced utilization of the dimensions so none of them will become a bottleneck. In general, we can state that this is a well-rounded alternative for generating multicast routes.

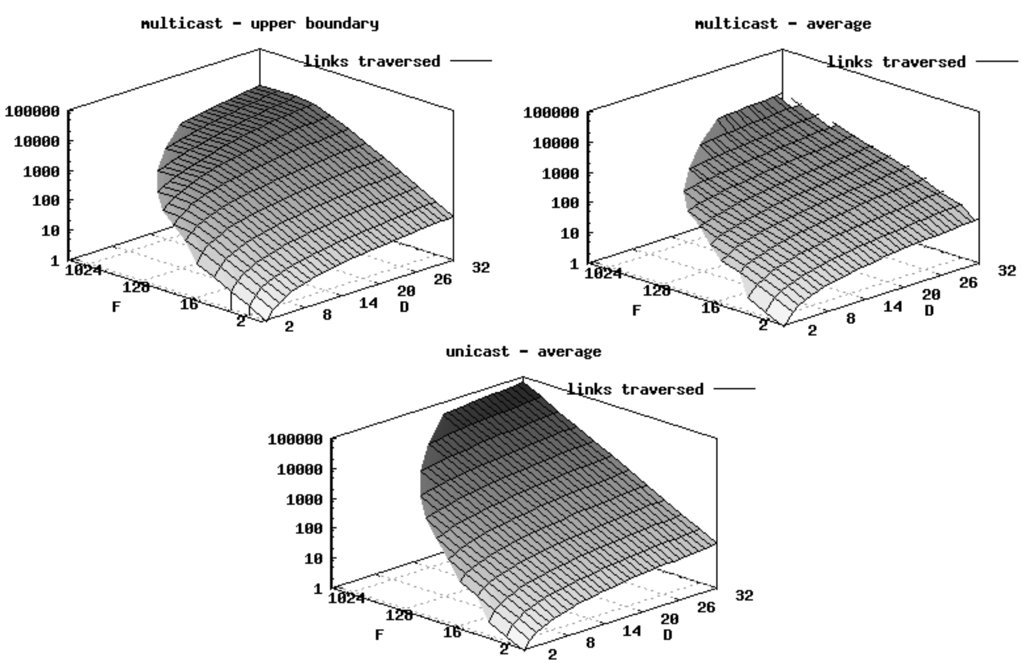


Figure 4. Network resources required to perform the communications.

ESPR reduces the network resource requirements when compared with DOR and LFR and keeps balanced the per-dimension utilization. The costs to pay are increases of the number of entries in the routing tables and of the time taken to produce the multicast trees.

Finally, NER produces the multicast trees using the lowest amount of network resources. However results are produced very slowly and require the largest amount of entries in the routing tables, a situation that can be exacerbated because routes are not straight, which would complicate route masking. For this reason NER should only be used in those cases in which reducing network utilization is *critical* and the number of table entries is not.

5 Motivation for the use of multicast

This section is devoted to formulate analytic models of network requirement to support neural communications, both for multicast and unicast approaches. We derive first the network resources employed to communicate a spike to all connected neurons (N). It is measured as the total number of traversed links (hops). We will derive it from $\bar{d}$ and F, both of them depending only on the workload, not on the actual hardware. In the case of unicast, deriving the average network requirement from these two parameters is straightforward; F packets sent to an average distance of $\bar{d}$ then, the *average* network requirement for unicast will be:

$$N_u = \bar{d} \cdot N \quad (1)$$

In the case of multicast, it is not easy to derive an expression for the average network requirement because this figure strongly depends on the employed strategy. Instead we can

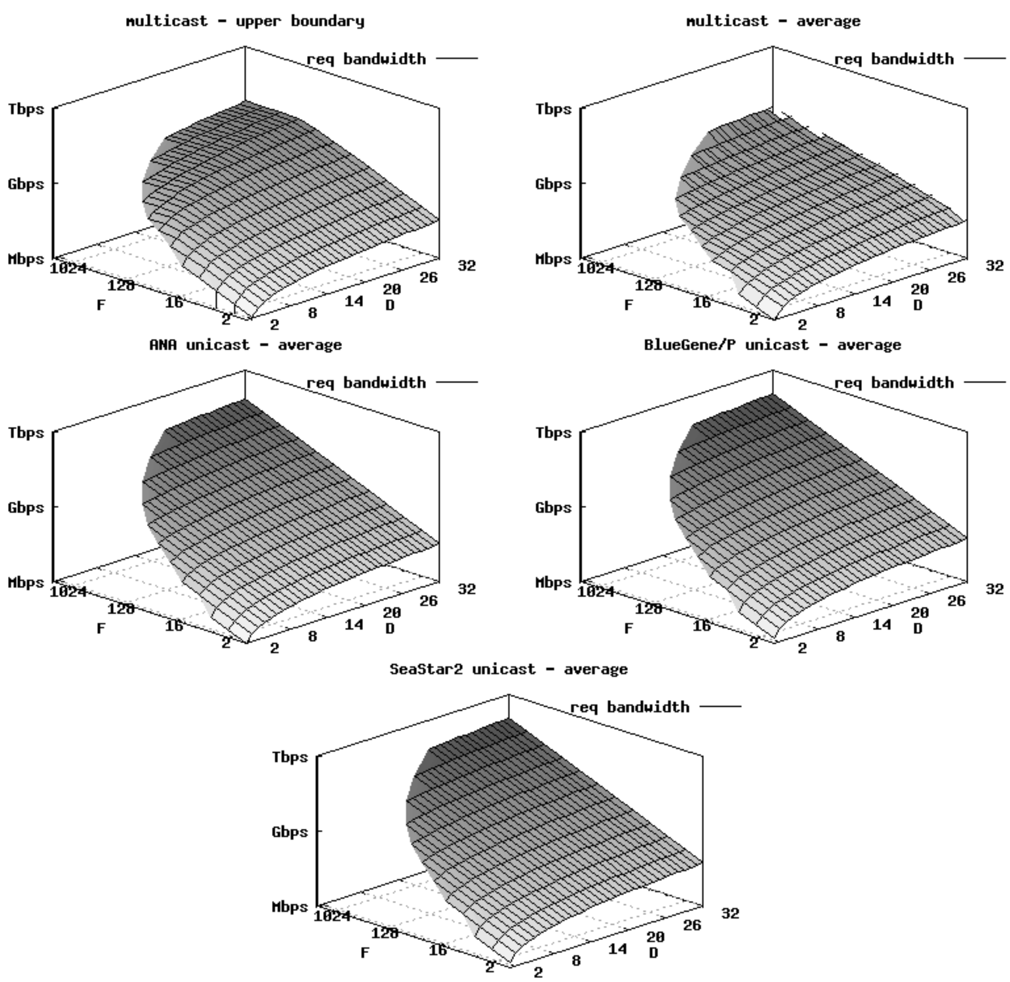


Figure 6. Network bandwidth (B_L) required to support regular system operation.

derive an upper bound estimator of network requirement when using multicast traffic. This estimator takes into account the worst configuration that could exist; this is, when routes to the destinations share as less as possible of their paths. This estimator can be formulated as:

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

Fig. 5 shows N_u and N_M with a sweep of the values of interest for the parameters $\bar{d}$ and F. For the sake of completeness we also show N_m, the average network utilization when using the previously defined ESPR strategy which has been computed as the average value of 10^5 random runs. In the figure it is clear that N_M is always lower or equal than N_u. This is obvious as the worst case for multicast is having completely disjoint routes, i.e. unicast behavior.

From (1) and (2) we can derive some other figures considering hardware characteristics. More specifically in the remaining of this section we will consider five different configurations: (i) worst-case and (ii) average multicast with ANA router (1Gbps, p_s = 40 bits), (iii) average unicast with ANA router, (iv) average unicast with the Blue Gene/P router (3.2Gbps, p_s = 64 bits) [14] and (v) average unicast with the SeaStar2 on Cray's XT4 (6Gbps, p_s = 40 bits) [15]. Unless otherwise stated, the system will be modeled in a regular operational status, i.e. n_n = 18000, f_s = 10Hz and L = 6. We will derive next the network bandwidth required to support a given system configuration, assuming evenly distributed traffic.

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

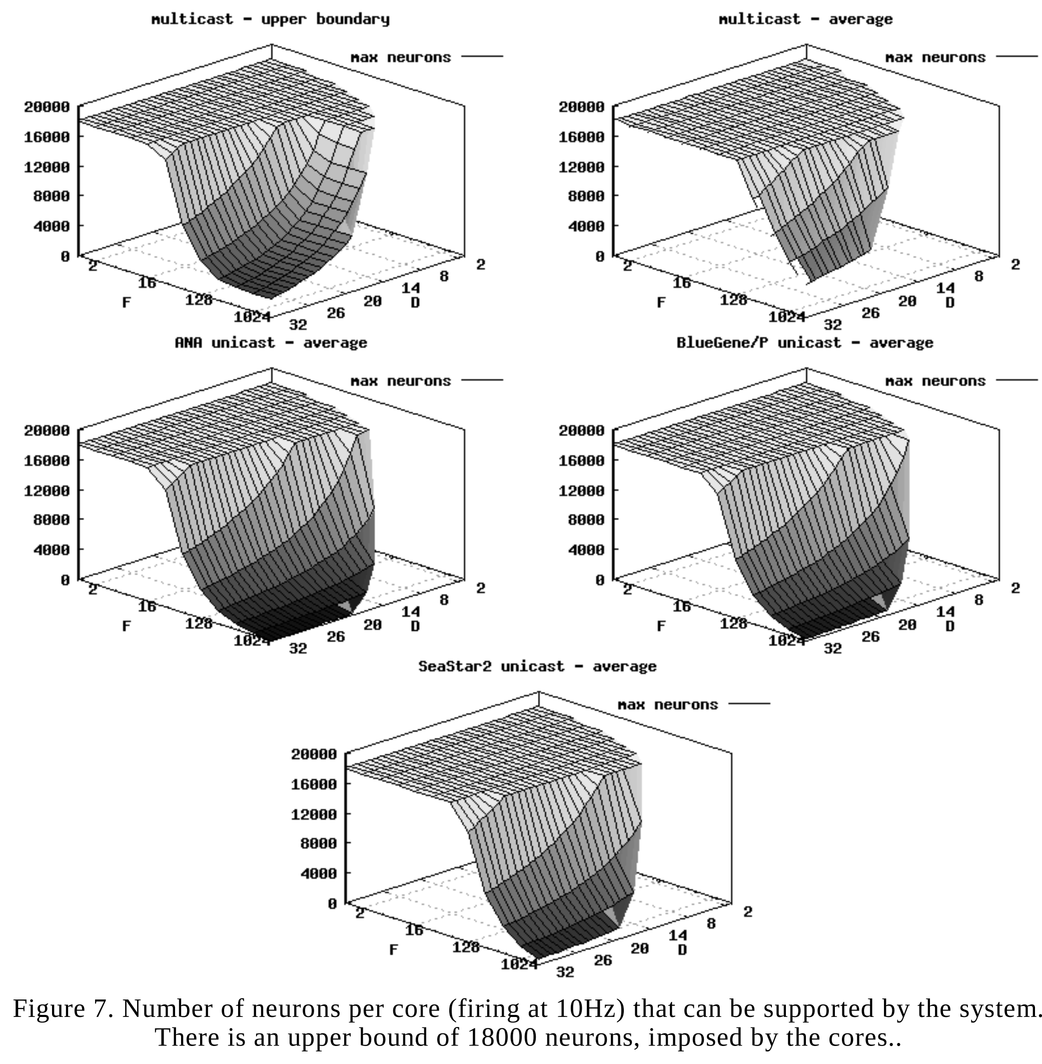


Figure 7. Number of neurons per core (firing at 10Hz) that can be supported by the system. There is an upper bound of 18000 neurons, imposed by the cores..

From (1), (2), (3) and the described values for the different systems, we plot in Fig. 6 the network bandwidth required to support the neural simulation. The *average* bandwidth required when using any *unicast* router can be more than one order of magnitude higher than the *upper bound* of *multicast* router. We will close the section deriving an important figure of merit, the number of neurons per-core that the system can support using each configuration.

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

From (1), (2), (4) and the discussed parameter values, we can compute the amount of neurons that the system is able to execute in each node. For the sake of clarity we will limit the amount of neurons that can be simulated to the maximum acceptable by the cores (18k). This way we will be able to see when the network becomes a limiting factor. This figure is shown in Fig. 7. In these plots it is evident that in the less demanding network configurations (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 figures increase the networks start becoming the limiting factor (slopes). This happens more often with the unicast routers, even when the Blue Gene and SeaStar2 have noticeably higher bandwidths than ANA'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 support only *hundreds* of neurons, while multicast is still able to support *thousands* of them.

6 Conclusions

In this paper we have highlighted the adequacy of multicast routing for the kind of applications executed on top of ANA 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 amount of neurons. More specifically we have shown how the multicast router implemented in ANA can achieve better performance than more complex router designs such as the SeaStar2 or Blue Gene's router, even though they provide noticeably higher network bandwidth.

In addition we described four different strategies to construct multicast trees and measured their resource consumption. Generating multicast trees using DOR reduces the utilization of the entries within the routing tables (a precious resource as there are only 1024 entries per chip), but demands the highest network resources. LFR is a well-rounded solution as the multicast trees can be generated very quickly while keeping resource requirements low enough. More sophisticated strategies which look for routes in their surroundings have been also considered. ESPR searches for connections using always shortest path; in contrast NER searches in all the surroundings even if no shortest path is used. These two strategies further reduce the network requirements but they are two orders of magnitude slower to generate than oblivious strategies.

These four strategies were described and implemented bearing in mind a two-dimensional triangular torus topology, but we want to remark that they are general enough to be employed in other cube-like topologies such as the 3D tori implemented in *state-of-the-art* massively parallel processors such as IBM's Blue Gene or Cray's XT families. In fact, the adaptation would be straightforward simply by using 3D torus routing functions (DOR and LF).

References

[1] *<Anonymized for blind review>*

[2] *<Anonymized for blind review>*

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

[4] P Dayan, LF Abbott. "Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems". The MIT Press. 2005.

[5] H Brody. "Cell counts in cerebral cortex and brainstem". Alzheimer disease senile dementia and related disorders 7, 1978, pp. 345–351.

[6] *<Anonymized for blind review>*

[7] *<Anonymized for blind review>*

[8] *<Anonymized for blind review>*

[9] W Maas, CM Bishop. "Pulsed neural networks". The MIT Press. 1998.

[10] S Bhattacharya, G Elsesser, WT Tsai, DZ Du. "Multicasting in Generalized Multistage Interconnection Networks". Journal of Parallel and Distributed Computing 22(1) 1994, pp. 80-95.

[11] S Coll, FJ Mora, J Duato, F Petrini. "Efficient and Scalable Hardware-Based Multicast in Fat-Tree Networks". IEEE Transactions on Parallel and Distributed Systems 20(9), September 2009

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

[13] JK Shapiro, J Kurose, D Towsley, S Zabele."Topology discovery service for router-assisted multicast transport". Procs. of IEEE Open Architectures and Network Programming, 2002, pp. 14-24 .

[14] S. Alam, et al. "Early evaluation of IBM Blue Gene/P". Proc. of the 2008 ACM/IEEE Conference on Supercomputing, pp. 1-12, 15-21 Nov. 2008, Austin, TX, USA. DOI: 10.1109/SC.2008.5214725

[15] 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.