Social-Insect-Inspired Networking for Autonomous Load Optimisation

Matthew Rowlings*, Andy Tyrrell, Martin Trefzer

Intelligent Systems Group, Department of Electronics, University of York, York, YO10 5DD, UK

* Corresponding author. E-mail address: mr589@york.ac.uk

1. Introduction

The extremely high logic density of modern Very-Large-Scale Integration platforms has lead to an adoption of Many-Core systems for embedded system design, relying on a Network on Chip (NoC) [1] [2] to interconnect the processing elements of the many-core array. Whilst the NoC shares many properties with conventional computer networking, the application to embedded systems means that the network should be designed to conform to typical embedded system constraints such as power efficiency, compact resource requirements and effective fault tolerance. Therefore an inclination to simpler networking capabilities is seen with NoCs when compared to conventional networking, a caveat of this is that performance suffers and so a trade-off between node router complexity and performance has to be made. An alternative is to perform off-line analysis [3][4][5] and optimisation of the task and network model, however the resulting strategy is generally fixed and so does not support runtime dynamic reconfiguration of the system structure; a key requirement for supporting future many-core system design paradigms such as dynamic task allocation and autonomous online optimisation [6].

Therefore we need a network that can self-organise and self-optimise without the need for off-line analysis. To support both good scalability and dynamic network topology reconfiguration, an ideal routing algorithm should therefore not rely on global knowledge of the network layout; indeed if many-core systems do scale into the hundreds and thousands of cores as suggested in [6] then any online analysis will be computationally infeasible within an embedded system. Therefore the network will have to take a decentralised approach to routing, whereby each node in the network is responsible for its own routing behaviour. An extremely simple example of this is the Round Robin algorithm [7]. By servicing each port in turn and only allowing each port to be serviced once in a round, round robin provides a decentralised and fair routing strategy that does not rely on any global coordination. Whilst the authors appreciate that round robin is a very simple case in a field full of more capable algorithms for specific applications, its simplicity means that not only is it suitable for implementation of a NoC in an embedded system but it also serves as a good baseline to compare the self-organising algorithms proposed in this paper to.

When researchers consider self-optimising systems, many have looked towards Nature for inspiration. Life has provided a host of examples of decentralised self-organising systems at all ranges of abstraction: from the chemical networks used for Gene Regulation, to the cellular growth and development in multicellular organisms, up to the social networks required for survival of insect (and other) colonies. However when these models are applied to engineering problems we often see significant overhead requirements due to the extra resources required...
to fit the engineering model to the biological metaphor. The UNITRONICS project for example [8] uses multi-cellular development as an inspiration for building fault tolerant systems but a simple 4x4 hardware multiplier required 40 cells, where each cell requires significant hardware resources to support all of the cell development model; an arguably large amount of unnecessary overhead for a simple circuit that makes scalability across a whole many-core system infeasible. Therefore when considering inspiration from Nature, it is important to find good links between the metaphor and the target problem. For many-core networking we can break our problem down into a decentralised model with simple communication between neighbouring nodes, in addition to local knowledge at each node. To implement this on chip requires as efficient a model as possible, whilst bearing in mind that although Nature produces efficient solutions they are not guaranteed to be optimal! Considering this, we argue that Social Insects are a suitable inspiration for many-core networking as their communication structures fit the decentralised routing model well: simple communications between members result in self-organising behaviours emerging when observed globally at colony level; examples include nest building, self-replication and food scouting, gathering and dispersal. Their typical habitat has resulted in Nature to produce behaviours that are very food (i.e. energy) efficient, fitting the embedded system application case well; indeed successful use of social insect models in other problem areas have succeeded because of the efficient emergent high level behaviour (Ant Colony Optimisation applied to the Travelling Salesman Problem for example [9] [10]). This paper describes a series of simulations exploring the application of social insect inspired network routing schemes to a NoC. First each of the routing algorithms and the NoC are described in terms of the local sensory inputs available to each node. Section 4 then presents a statistical analysis of the performance of each routing scheme on two representative NoC applications. Finally the results presented here inform discussion in Section 6 on how the model will be elaborated to exhibit autonomous NoC topology optimisation, whilst also touching on aspects of a future hardware implementation. Despite fault tolerance not being an explicit focus of this investigation, the decentralised sense-threshold model provides the emergent adaptive behaviours required for a significant first step towards a fully autonomous fault tolerant system; as is also described in this section.

2. Intelligent Many-Core Routing

To investigate which behaviours of the social insects can be applied to hardware systems, biological observations of the social insects are considered. Those of particular interest include [11] wherein the authors argue that a honey bee can be described in terms of a behavioural repertoire of 59 distinct behaviour patterns and [12] which explores several models of agent task allocation and how this is determined at the single organism level to result in emergent task allocation pattern at colony level. These distributed models have a clear analogy to the desired properties of many-core systems; each node in such a system could be modelled with a distinct behavioural repertoire depending on its function and then the repertoire of each node (or agent) is exploited at the local level to provide a scalable, distributed system with the desired emergent properties demonstrated by social insect colonies, i.e. scalability, adaptability to new environments and colony fault tolerance. This is explored by simulating social insect inspired routing algorithms on a many-core system with 36 nodes, as described in Section 3. Instead of the more traditional design-space analysis approach to Network-on-Chip routing, we considered several agent level behaviours that are implemented heterogeneously at each node in the many-core grid. Taking inspiration from [12] we used a simple threshold intelligence model at each node which takes data from sensors local to the node and performs a routing behaviour depending on the input to the sensors and the threshold function applied. In each investigation step different sensors are incorporated into the threshold model; with the aim to capture the following social insect inspired features within the many-core system:

(a) Simple sense/act behaviour of insects
(b) Efficient communication strategies between agents
(c) Emergent overall behaviour of the colony

This section describes the sensory capabilities investigated, with each capability striving to add more local information to each node with the ambition to improving the effectiveness of the emergent routing behaviour of the many-core system. All schemes aside from the first scheme (used as a benchmark case) employ a Response-Threshold intelligence strategy loosely based on the models presented in [12] and the final scheme (Neighbour Hunger Model) is also directly inspired from the food distribution behaviours of ant colonies. All the routing schemes operate on a 5-port router with cardinal inputs and an application node attached, as shown in Figure 1.

2.1. Round Robin Routing

To provide a suitable baseline to evaluate the proposed algorithms against, a decentralised routing algorithm that does not use any local node knowledge is required. Such a port servicing algorithm is the simple, but well used, Round Robin arbitrator[7]. In this routing scheme each port is checked in turn for data. If a port has data to route then a single packet is routed and then the next port in the router is checked for data. This continues serving ports in a circular fashion (in our case N,E,S,W,I and back to N), providing a fair and balanced routing
scheme without utilising any of the sensory information available to the node.

2.2. FIFO Fill Threshold Model

The first investigation architecture looked at the possibility of balancing the network’s overall load by balancing the traffic at each node. To achieve this each node was given knowledge of the fill levels of each of its North, East, South, West and Internal FIFO buffers. A simple threshold model then chooses the router port to service based on which node has the most traffic waiting in its buffers, i.e:

\[
\text{Port} = \max(\text{fill}(x)) \quad \forall x \in \{N, E, S, W, I\}
\]

where \( \text{fill}(x) \) is the FIFO fill level of port \( x \)

2.3. Packet Priority Threshold Model

An important part of a providing Quality of Service (QoS) within a network is a notion of priority. This allows a subset of packets (possibly system control or feedback data) to have routing preference regardless of the order they are sent. Due to the use of FIFOs in our system it is not possible for such packets to overtake packets already in the buffer, but we can give preference to buffers that contain high priority packets. This could be seen as analogous to fear in organisms; the more high priority packets that are in a buffer, the more an intelligent node “wants” to service that port to remove the higher priority packets. To achieve this we add a sensor that detects how many high priority packets are within a FIFO and add this to the threshold model. Thus our selection algorithm becomes:

\[
\text{Port} = \max(\text{fill}(x) + \lambda \text{HP} x) \quad \forall x \in \{N, E, S, W, I\}
\]

where \( \lambda \) is the “fear factor” controlling how much preference should be given to servicing high priority packets and \( \text{HP} x \) is the number of high priority packets in a buffer. \( \text{fill}(x) \) is the FIFO fill level of port \( x \) as with the first model.

2.4. Neighbour Hunger Model

The previous models have considered how sensory inputs can be used to decide which port to service, this model however considers the output port that a packet should be routed to. As introduced in Section 3, each node has an ordered preference list of output ports that it should route a packet to dependent on its task. This ensures that short paths are taken through the network, but it can also mean that certain nodes are put under lots of pressure in congested scenarios. Sometimes a longer path through the network will result in a more balanced traffic flow, potentially improving throughput over the entire network. To investigate this we have considered the food distribution networks of ant colonies [13]. The nature of task polyethism in the colony means that food must pass from returning foragers at the front of a nest all the way through the entire colony to keep, all members well fed. From a simplistic point of view this is achieved by individual members taking more food than required for themselves when offered and then sharing it with other members as they pass them in the nest. If the recipient already has a plentiful amount of food then she refuses the offer and the supplier tries other members until her food excess is removed. This self-organising, distributed methodology successfully balances food distribution across the colony; this investigation explores if we can use such a scheme to balance packet distribution across the NoC.

As a node’s buffers fill up, their fill is accumulated and compared to a “hunger” threshold, the output of which is shared with each of a node’s neighbours. When this threshold is passed then a node is no longer “hungry” and its neighbours will not send it any more packets until it becomes “hungry” again. To achieve this the ordered preference list of output ports is used to select an increasingly less optimal output port should the neighbour on the optimal output port not be hungry, if all neighbours are not hungry then the router does not route any packets until one of them is.

Formally, for each node:

\[
\text{Hungry} = \begin{cases} \text{‘Yes’} & \text{if } \sum(\text{fill} x) < \theta \\ \text{‘No’} & \text{else} \end{cases} \quad \forall x \in \{N, E, S, W, I\}
\]

And when servicing a port:

\[
\text{Output Port} = \text{first}(y) \quad \text{if Hungry}(y) = \text{‘Yes’}
\]

where \( \theta \) is the total FIFO fill level at which a node is no longer hungry and \( \text{fill}(x) \) is the FIFO fill level of port \( x \) as with the first model. \( y \) represents a list of the the node’s N,S,E,W neighbours (which may not exist if a node is located on the edge)

3. Network on Chip Architecture

To evaluate the above routing schemes we need a many-core system to investigate with, which for this paper is a simulated 6x6 grid of interconnected nodes; the choice of a 6x6 NoC has been made with regards to the hardware system available for later experiments with a hardware platform (as introduced in Section 6). Each node is given a task that represents the operation that the Processing Element at the node carries out. Application graphs are simulated by generating packets from one task with other tasks as their destination. This task orientated routing scheme allows packets to be routed based on the task they must reach as opposed to a specific node performing the desired task. This allows more flexibility in the network as any node of the correct target task can accept the packet, as well as being more suited to a decentralised many-core system should other distributed behaviours such as dynamic task reallocation be explored in the future. Figure 1 illustrates the router design used for the investigations. Each router can route packets to and from their North, East, South and West neighbours (unless they are on the boundary of the grid), as well as routing packets internally to simulate acceptance of a packet by a node of the target task. The routers work in a blocking fashion: they can only route one packet at a time and are blocked for the duration of the routing operation. As a “store and forward” [14] routing style was used, input First In, First Out (FIFO) buffers on each port ensure that a packet can be routed to a router even if it is blocked. If a packet is accepted by a node then a “CPU time” emulates the time that a node is blocked whilst it carries out an operation, this is representative of our target hardware system introduced in Section 6. Each node is given a ordered...
preference list of destination ports it should route specific tasks to; for example if a packet destined for a Task 2 node arrives on the North port, then a router can check the destination list to see that it should route these packets West for the shortest path to the nearest Task 2 node. This ensures that efficient paths are taken through the network.

4. Experimental Results

This section presents the results of simulation of a many-core system exploiting each of the following routing schemes. Two application scenarios are considered to show the advantages and disadvantages of each scheme under different operating conditions. Each simulation records the time it takes each packet to traverse the network from the node it is generated at (the source node) and the node it is finally accepted at (the target node). The simulation is run for 10,000 timesteps, after which no more packets are generated and it continues running until all packets finally reach their task target node. The mean packet traversal time is then calculated for each task and recorded. Next the simulation is rerun 1000 times in total to permit a statistical outline of the performance across many variations in network node topology, capturing the mean performance as well the worst and best case outliers. This is repeated for the following routing schemes with each task scenario: (1) Round Robin, (2) FIFO Fill Threshold, (3) FIFO Fill Threshold with Packet Priority and (4) FIFO Fill Threshold with Packet Priority and Neighbour Hunger.

4.1. Application Scenarios

The application scenarios simulated aim to show the abilities of each routing scheme whilst capturing a sense of realism in their application. To help limit the number of parameters to consider when simulating many-core task traffic, three tasks were used in each scenario with their allocation to nodes in the network being entirely random resulting in vastly different network topologies for each run. Scenario 1 reflects the case where the tasks are distributed uniformly across the many-core array, each task has the same packet generation parameters and the same number of instance of each node; the only difference between tasks is that Task 3 sends out high priority packets. This is illustrated in Figure 2.

Fig. 2. Application task graph for Scenario 1. The numbers within the circles represent the task at a node and the arrows indicate which task it sends packet to, with the dotted line showing the high priority link between Task 3 and Task 1. The numbers on the arrows represent the packet size and the small italic numbers within the circles the CPU time of a node. The fraction dictates the ratio of each node in the system. In Scenario 1 all tasks send packets with a rate of one every 60 timesteps.

Scenario 2 has been created to represent a computational bottleneck in the system that a many-core designer has mitigated by including many parallel instances of Task 2, as can be seen by the parameters given in Figure 3. Task 1 generates data at a high rate and Task 2 must perform a computation on the data before it can issue its packet to Task 3. Task 3 generates small but high priority feedback packets to Task 1 at a slow but constant rate, this is to emulate dynamic quality feedback for an image compressor or other system control data that might be expected to traverse the same network as other task data by relying on the QoS scheme of the network.

Fig. 3. Application task graph for Scenario 2. This shows the unbalanced task ratio and the parallel nature of scenario 2. The star by Task 2 nodes represents the constraint that a Task 2 node must receive a packet and wait for the CPU time before a second packet out (italic number). Task 1 generates packets every 500 time steps to ensure the network becomes saturated, and Task 3 mimics small control packets by sending a packet every 100 timesteps.

4.2. Scenario 1 Results

Figure 4 shows the average packet traversal times for each task over 1000 runs of scenario 1. These results clearly show a congested network case from the high average packet traversal time, but this is required to test our routing schemes in situations where adaptive behaviours can thrive. It is clear from the plot that both the Round Robin and the FIFO Fill threshold schemes do not treat any task differently over another, the difference in extremities of the plot is due to particularly optimal/poor network topologies. The FIFO fill threshold scheme reduces the average delay because it always services busy buffers first, this is a source of long packet traversals in Round Robin as it will service other ports even if a new packet has only just arrived when it is a port’s turn to be serviced; leading to potentially large queues at some node ports.

By extending the FIFO fill scheme with detection of high priority packets, we see a significant change in behaviour. Task 3 packets are treated differently to Task 1 and 2 resulting in a severe drop in the average packet traversal time for Task 3 packets, the extremities again due to unfavourable network topologies. This faster traversal of Task 3 packets is balanced by a drop in performance for Task 1 and 2 packets, the extra network capacity has to come from somewhere and these tasks suffer although this is acceptable when providing QoS to the network.

The final scheme illustrates the power of inter-node communication. Both the spread and average are better than the High Priority case and the lower extremities suggest that bottlenecks induced by the network topology are mitigated by the adaptive routing permitted by the Hunger scheme. The high priority scheme is integrated within this scheme but does not...
suffer because of it, indeed it even reduces both the worst case and spread without sacrificing the best case performance.

4.3. Scenario 2 Results

Despite the application profile for Scenario 2 being notably different to the application profile of Scenario 1, we would ideally expect a comparable improvement despite the difference in application profiles. As can be seen in Figure 5, a similar improvement does exist between the schemes. Round Robin performs extremely varied depending on the network topologies, as characterised by the high extremities. This is also shown in the distribution of Task 3 average traversal values which shows a very large range.

Indeed a limitation of all of the schemes aside from the Hunger routing is that no Task 1 node can take advantage of the parallel nature of Task 2 and so all packets are sent to the (albeit) nearest router of a Task 2 node instead of being distributed amongst several Task 2 nodes. This results in the Hunger scheme showing not only the smallest average packet traversal time but also the smallest range and much smaller worst and best case extremities.

5. Discussion

The results of investigations presented have shown that the Social Insects provide effective inspiration for self-optimisation of Network on Chip routing in a fully decentralised fashion. Simulation of two different application scenarios yielded a very similar emergent behavioural pattern, suggesting that this approach can be applied to a wide range of applications without requiring any NoC topology pre-analysis or constraints. It is anticipated that these routing models will enable adaptive routing schemes to be used towards development of a truly flexible many-core usage model, whereby tasks can be added dynamically into the many-core system and the self-optimisation will allow the system to maintain a homeostasis through online adaptation. It will, however, be important to re-evaluate these investigations with different NoC models in either software or hardware implementations, as subtle differences between models will add or remove capabilities that the network can exploit.

In this model, for example, the sizes of the FIFO buffers of each node are not fixed, whereas in a real system these buffers will have a maximum capacity which will possibly change the network dynamics as congestion could have a knock-on effect to other routers in the network. Whilst this type of behaviour may be predicted for the simple routing behaviours presented in this paper, with more complex behaviours implemented within each node the characterisation of such behaviours would require significant analysis, which is exactly what this investigation is aiming to abolish.

6. Conclusions and Further Work

Intelligent routing was the main focus of this investigation, however the results suggest and the metaphor adopted ensures that by simply including additional sensory inputs it would be possible to enhance the behaviour of each agent. The local "sense-think-act" model for each agent has been demonstrated to be very powerful when viewed as an emergent behaviour at the system level. Therefore we shall look at how we can extend this model by taking other influences from applicable behaviours from the social insects. Indeed the polytheistic nature of the social insects is an obvious starting point for experimenting with dynamic task allocation and swapping. The outliers in the simulations have shown how important the NoC topology is to routing effectively and therefore is the next extension to the many-core self optimisation. With the right sensory inputs (for example ring oscillators, FIFO fill monitors), there is scope for extending the same intelligent task allocation in such a way to also offer fault prediction and tolerance. This would allow a many-core system to autonomously optimise core task allo-
cation whilst taking the temperature or degradation state of the device into account at runtime, as opposed to the complex analysis and simulation that is currently used to solve these issues.

However the ultimate proof of the effectiveness of an intelligent system when applied to the embedded systems field is to implement these models in a hardware system, this is the only way to fully close the reality gap. We plan to implement the 6x6 NoC into the RISA Many-Core Array. RISA [15] is a FPGA and co-processor specifically designed for bio-inspired systems research. A 36-node array of RISA integrated circuits is currently under development to emulate the behaviour of the NoC, this will also allow experimentation of custom sensory hardware at each node that would not be possible with a commercial FPGA. This platform will also allow different intelligence models (hardware friendly neural networks for example) to be experimented with and evaluated in hardware for their intelligence possibilities offset against their resource requirements.

Acknowledgements

This work was supported by funding from the Department of Electronics and an EPSRC DTA award.

References

[1] Benini, L., De Micheli, G. Networks on chips: a new soc paradigm. Computer 2002;35(1):70-78.
[2] Hemani, A., Jantsch, A., Kumar, S., Postula, A. Network on chip: An architecture for billion transistor era. Proceeding of the IEEE NorChip Conference Vol 31 2000.
[3] Lee, H.G., Chang, N., Ogras, U.Y., Marculescu, R. On-chip communication architecture exploration. ACM Transactions on Design Automation of Electronic Systems 2007;12(3):23-es. doi:10.1145/1255456.1255460.
[4] Srinivasan, K., Chatha, K., Konjevod, G. Linear programming based techniques for synthesis of network-on-chip architectures. In: IEEE International Conference on Computer Design: VLSI in Computers and Processors. 2004. ICCD 2004. Proceedings. IEEE. ISBN 0-7695-2321-9, 2004, p. 422–429. doi:10.1109/ICCD.2004.1347957.
[5] Ogras, U., Marculescu, R.. Energy- and Performance-Driven NoC Communication Architecture Synthesis Using a Decomposition Approach. In: Design, Automation and Test in Europe. IEEE; 2005, p. 352–357. doi:10.1109/DATE.2005.137.
[6] Tempesti, G. Graceful Design. International Innovation Issue 140 2014;:76–78URL: http://www.internationalinnovation.com/graceful-design/.
[7] Shin, E.S., Mooney, V.J., Riley, G.F. Round-robin arbiter design and generation. In: Proceedings of the 15th international symposium on System Synthesis - ISSS '02. New York, New York, USA: ACM Press; 2002, p. 243.
[8] Samie, M., Dragffy, G., Pipe, T. UNITRONICS: A novel bio-inspired fault tolerant cellular system. In: 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS). IEEE. ISBN 978-1-4577-0598-4; 2011, p. 58–65. doi:10.1109/AHS.2011.5969367.
[9] Dorigo, M., Gambardella, L.M. Ant colonies for the travelling salesman problem. BioSystems 1997;43(2):73-81.
[10] Stützle, T., Dorigo, M. ACO algorithms for the traveling salesman problem. Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications 1999.
[11] Chitika, L., Niven, J. Are bigger brains better? Current Biology 2009;19(21):R995–R1008.
[12] Beshers, S.N., Fewell, J.H. Models of division of labor in social insects. Annual review of entomology 2001.46(1):413–440.
[13] Sudd, J.H. An introduction to the behavior of ants. St. Martin’s Press; 1967.
[14] Cypher, R., Meyer auf der Heide, F., Scheideler, C., Vöcking, B. Universal algorithms for store-and-forward and wormhole routing. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC ’96. New York, New York, USA: ACM Press; 1996, p. 356–365.
[15] Greensted, A., Tyrell, A. RISA: A Hardware Platform for Evolutionary Design. In: 2007 IEEE Workshop on Evolvable and Adaptive Hardware (WEAH2007). IEEE; 2007, p. 1–7.