
Proceedings Paper:
Rowlings, Matthew orcid.org/0000-0003-3800-2055, Tyrrell, Andy orcid.org/0000-0002-8533-2404 and Trefzer, Martin Albrecht orcid.org/0000-0002-6196-6832 (2020) Embedded Social Insect-Inspired Intelligence Networks for System-level Runtime Management. In: Design, Automation and Test in Europe Conference:DATE2020. , Grenoble, France.

Embedded Social Insect-Inspired Intelligence Networks for System-level Runtime Management

Matthew R. P. Rowlings, Andy M. Tyrrell and Martin A. Trefzer
Department of Electronic Engineering
University of York, York, UK
matthew.rowlings,andy.tyrrell,martin.trefzer@york.ac.uk

***Abstract*—Large-scale distributed computing architectures such as, e.g. systems on chip or many-core devices, offer advantages over monolithic or centralised single-core systems in terms of speed, power/thermal performance and fault tolerance. However, these are not implicit properties of such systems and runtime management at software or hardware level is required to unlock these features. Biological systems naturally present such properties and are also adaptive and scalable. To consider how these can be similarly achieved in hardware may be beneficial. We present *Social Insect* behaviours as a suitable model for enabling autonomous runtime management (RTM) in many-core architectures. The emergent properties sought to establish are self-organisation of task mapping and system-level fault tolerance. For example, large social insect colonies accomplish a wide range of tasks to build and maintain the colony. Many thousands of individuals, each possessing relatively little intelligence, contribute without any centralised control. Hence, it would seem that social insects have evolved a scalable approach to task allocation, load balancing and robustness that can be applied to large many-core computing systems. Based on this, a self-optimising and adaptive, yet fundamentally scalable, design approach for many-core systems based on the emergent behaviours of social-insect colonies are developed. Experiments capture decision-making processes of each colony member to exhibit such high-level behaviours and embed these decision engines within the routers of the many-core system.**

***Index Terms*—bio-inspired hardware, fault tolerance, social insects, adaptive system, many-core, runtime management**

I. Introduction

State-of-the-art electronic design allows the integration of complex electronic systems comprising thousands of high-level functions on a single chip. This has become possible and feasible because of the combination of semiconductor technology providing atomic-scale devices, allowing very large scale of integration (VLSI) of billions of transistors, and electronic design automation (EDA) tools that can handle their useful application and integration by following a strictly hierarchical design methodology breaking down a system into blocks, sub-blocks or cells. This results in many layers of abstraction within a system that makes it implementable and verifiable, hence, explainable which is usually desired. However, while many layers of abstraction maximise the likelihood of a system to function correctly (because it can be verified and debugged) this can prevent a design from making full use of the capabilities of a process technology.

This work was supported by an EPSRC DTA and the EPSRC Platform grant 'Bio-inspired Architectures and Systems' EP/K040820/1.

Moreover this places electronic systems, in the way they are currently designed, at opposite ends of the scale from emergence as—by design—they can be understood from a purely reductionist point of view. The fundamental component of an electronic system, the transistor, is known and the design hierarchy is constructed bottom-up. Starting at the top level, this hierarchy can be traversed in the opposite direction and each block can be understood and explained by looking at the components it is made of. The whole methodology has been developed to avoid unforeseen behaviours and therefore there appears to be no room for emergence.

However, the ever-increasing transistor density and design complexity makes modern systems brittle. As we start to meet fundamental device scaling limits when touching the atomic scale, design challenges arise including the thermal/power constrained Dark Silicon and other deep sub-micron silicon fabrication issues such as intrinsic variability and electrical wear out (ageing). This gives VLSI designers a large number of pessimistic design constraints that must be met to avoid faults and guarantee a certain lifetime of a device. Despite that, the yield (percentage of chips on a silicon wafer that operate according to specification) continues to decline.

This gives rise to the idea of biologically-inspired hardware, which is indeed capable of emergent behaviours or features. Of course the challenge here is to adopt and implement these concepts while achieving a "next-generation" kind of electronic system which is considered at least as useful and trustworthy as its "classical" counterpart—plus additional features. Considering this, the question may be asked whether it is acceptable or useful to speak of emergence at all in the context of bio-inspired hardware, given that the bio-inspired parts also need to be designed using a VLSI methodology and must be comprehensible.

The concept of "emergence" is usually taken to relate to something like an unexplained or unexplainable appearance of an entity or property which is not further reducible to known interactions of other components (non-reductionist, holistic) [1], [2]. For example, when observing ants by looking at the behaviour of the entire colony rather than the individuals, the colony can indeed be regarded as a singular entity. Based on this it can now be suggested that drawing inspiration from structure and behaviour of biological systems can bring new, useful behaviours to electronic systems which are explainable and verifiable at some lower level, but which

can indeed be regarded as "emergent" properties, e.g. in the context of the entire system.

In this case, the emergent property sought to establish is system-level fault tolerance, the inspiration from biology are social insects (ant colonies), and the hardware system is a many-core computing architecture where application tasks and data need to be allocated, transferred and organised. The model of processing elements communicating amongst each other via a network on chip (NoC) provides a conceptual link with many scalable biological models.

This paper introduces a self-optimising and adaptive, yet fundamentally scalable, design approach for many-core systems based on the emergent behaviours of social-insect colonies. Experiments aim to capture the relevant decision processes made by each member of the colony to exhibit such high-level behaviours and embed these decision engines within the routers of the many-core system. Results with the bespoke 128-core Centurion platform suggest that there is potential for the social insect model as a distributed, embedded intelligence within a many-core system and with the relevant knobs and monitors, such as packet routing events, timing violation detection, router behaviour, clock frequency and temperature, to close the loop for emergent autonomous adaptation and fault tolerance [3], [4].

II. Social Insect Intelligence

There are many examples of large complex systems in Nature that exhibit both the the scalability and collective co-ordination that are required for large many-core systems. From the molecular interactions of gene regulatory networks driving development of multi-cellular structures [5], [6], to the chemical signalling between bacteria in a *Protoza* society [7] and complex systematics of colonial organisms in a *Hydrozoa* consisting of many sub-organisms [8]; often these natural systems exhibit highly scalable development and maintenance abilities for solving a particular set of actions for survival in a number of challenging environments.

In ant colonies, individual workers have limited memory and decision making capabilities [9] yet when working together at a huge scale exhibit many non-centralised features that are desirable for large distributed computing systems including self-organisation, self-optimisation and fault tolerance. This section introduces intelligence abilities of individuals, what capabilities emerge at the colony level as a result, and how these behaviours translate into desirable hardware system capabilities.

A. Biological Model of Ant Behaviour

The decentralised yet highly-scalable and adaptive task allocation of social insect colonies is the key dynamic that needs to be captured for autonomous task management in large scale many-core systems. Considering the limited cognitive capabilities of an individual, task allocation capability necessarily emerges from colony dynamics rather than some central coordinator or highly-informed decisions by specific individuals. In this work, we are focussing on ant colonies,

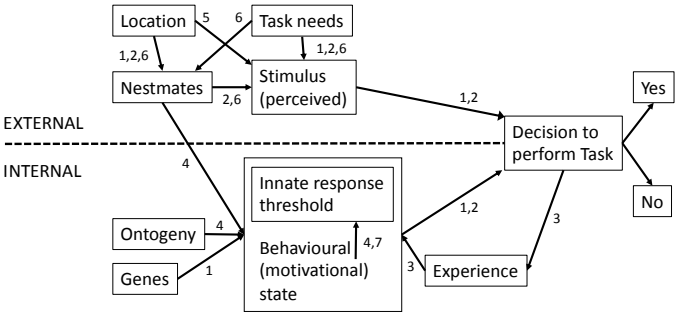

Fig. 1. Illustration of factors influencing an individual's choice to undertake a particular task. Numbers on the arrows indicate effects that are included in each type of model: 1 response thresholds; 2 information transfer; 3 self-reinforcement; 4 social inhibition; 5 foraging for work; 6 network task allocation. Figure adapted from [11]

because their tasks are well defined and their communication methods are relatively simple compared to other social insects, e.g. honey bees use visual cues [10], requiring neural pathways for image processing that would not translate well and impose large overhead in a hardware RTM unit.

Tasks that individual ants can undertake are either primarily internal to the nest or require leaving the nest for longer periods of time. Internal tasks include brood rearing, nest maintenance and expansion, removing dead and ill ants, food processing, distribution, storage, tending to fungus farms (some species) and queen care. External tasks include foraging for food, sharing locations of food sources, patrolling the nest and aphid farming (some species). This diverse set of tasks requires different sets of skills and sensory inputs for many of the tasks and so switching tasks will require cognitive decisions by an individual and the ability to decide what sensory inputs should be prioritised. The number of individual ants performing each task is also important and needs to be constantly within certain bounds to ensure the colony operates effectively as well as efficiently. For example, if the number of brood-rearing individuals increases, more foragers are needed to ensure new ants will survive.

Six classes of ant behaviour models are generally used in the literature [11], with each one differing in what information source is used by individuals to determine which task they should be undertaking: (1) *response threshold* responding to task-specific stimuli matching capability and capacity, (2) *integrated information transfer* adding information exchange between individuals to response threshold, (3) *self-reinforcement* to balance specialists vs. generalists through experience feedback, (4) *social inhibition* large numbers of experienced specialists inhibit more take up, (5) *foraging for work* using a production line analogy with spatial separation of tasks, and (6) *network task allocation* modelling at a higher abstraction level using differential equations. These schemes are represented in Figure 1 with illustration of which factors are present in each model.

B. Hardware System Implications

For a successful translation of a bio-inspired model into hardware, it is imperative that the underlying organisation

method used by biology maps appropriately to the targeted hardware architecture. For instance, any chemical or molecular based systems will require numerical computation for the model and so will be expensive to implement in hardware or may be lacking the desired biological properties if, e.g. the numerical precision is too low. In the context of modelling social insect decision making processes with a neuromorphic hardware approach, a key consideration is how to balance neural complexity with hardware overhead.

Looking at the biological examples from the previous section, it is found that large colony sizes can reduce the intellectual requirements of individuals, and that intelligence (neural) pathways in social insects are highly optimised for specific task performance using minimal space. This high level of optimisation does not necessarily mean that the behavioural abilities of an individual are limited in the context of the large colony. This is a useful feature for distributed large-scale hardware systems, because it means that small intelligence units embedded on a core-by-core basis will benefit from high node-counts found in many-core fabrics. It also means that it should be possible to exhibit more complex (emergent) behaviours with a relatively small number of embedded neural pathways, provided that it can be relied on minimal learning to create and maintain these circuits. It is envisioned that most of the pathways of the embedded intelligence will be analogous to the many innate behaviours of social insects with a few pathways providing the cognitive aspects which enable, disable or modify the innate behaviours.

The task models being related to an individual's location and local stimuli has a direct translation to many-core systems as each node will have a different set of task, thermal and NoC traffic stimuli depending on its location and current task. Hence, embedded intelligence should be located at each node to capture the locality aspects of the sensing and acting (decision making). Sensor readings from the many-core system should include stimuli that affect the ability of an individual node to complete a task (fulfilling the *response threshold* model), stimuli that signify work that needs to be done (fulfilling the *foraging for work* model) and stimuli that communicate what neighbouring nodes are working on (fulfilling the *network task allocation* model). A physical difference between social insect colonies and hardware systems is that individuals are mobile and carry information with them, whereas processing cores are stationary and send information packets through a network. Whilst the dynamics of both systems may not be identical, they are equivalent in terms of the information network formed.

Both the *response threshold* and *information transfer* models rely directly on a stimulus-threshold decision making intelligence, i.e. when a stimulus exceeds a threshold then a decision is made. However the experimental implementations of *foraging for work* [12] and *network task allocation* [13] both also use stimulus-threshold structures to make decisions. This motivates the implementation of a stimulus-threshold-based intelligence architecture for the embedded hardware model.

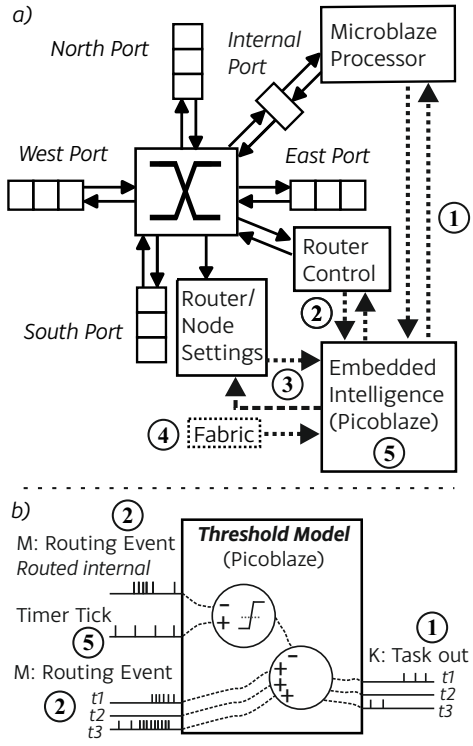

Fig. 2. *a)* The Centurion 5-channel NoC router. The main ports consist of the cardinal directions and an internal port connected to the processing element. A sixth-port, the *Router Configuration Access Port (RCAP)*, allows the router to be configured remotely. The embedded intelligence is implemented within a Picoblaze connected to varies sense/actuator interfaces within the system; the *monitors* and *knobs*. The numbers represent areas where monitors and knobs can be found: 1 Microblaze Node interface, 2 router control, 3 router settings, 4 FPGA fabric (ring oscillators), 5 internal to the Picoblaze *b)* Typical arrangement of sense-react thresholders that are used for implementing the intelligence models. A series of impulse based inputs are read into the Picoblaze, when they fire a counter is either increased (excitatory) or decreased (inhibitory). When a counter exceeds its respective threshold then the output knob is set (either impulse or vector).

III. Many-core Experimentation Platform

The Centurion many-core system has been developed as an experimentation platform to study scalability and performance of bio-inspired RTM algorithms in hardware. Specifically, FPGA-based Centurion platform has been designed with scalability, flexibility and usability in mind to enable experiments exceeding the number of cores that are routinely possible in simulation and inherently provides a real-world embedded hardware scenario. The hardware version used in this work is Centurion-V6 which consists of 128 processing elements connected in a 8×16 grid and is implemented on a Xilinx Virtex-6 LX760 FPGA. Each node in the many-core consists of a Xilinx MicroBlaze Micro Controller System (MCS) [14], a custom 5-port NoC router, and a configurable artificial intelligence module (AIM) implementing the bio-inspired models. This arrangement is shown in Figure 2.

A larger processor, the *Experiment Controller*, is connected to the NoC via the North ports of four of the (otherwise unconnected) routers in the top row. This larger AXI-based Microblaze manages the LVDS-based data communication link between the NoC and the PC and is used to manage

experiments and data. This allows experiment parameters to be sent from the PC and experiment runtime data to be sent from the NoC to the PC. The experiment controller can inject and receive packets from the NoC through its four NoC interfaces connected to the North channels of four routers on the top row. The experiment controller can also access the nodes separately to the NoC via a dedicated debug interface. This allows experiment data to be downloaded and parameters to be set at runtime (e.g. for fault injection) without interfering with the NoC traffic of active experiments.

A. Router Design

The Centurion router, shown in Figure 2, is a five port NoC router that includes a *Router Configuration Access Port (RCAP)* to allow router and embedded intelligence settings to be changed remotely via the NoC. It also includes several performance monitoring signals and settings for modifying the router's behaviour, which is utilised by the embedded social-insect intelligence model as explained in Sections II and III-C. The router supports two packet routing modes and is based on wormhole routing to reduce device resources spent on packet buffers. A basic deadlock recovery mechanism is included within the router to enable experiments to survive deadlock conditions, however this is not as comprehensive as other NoC deadlock avoidance schemes and so is not guaranteed to alleviate all deadlock conditions or detect and release deadlocked packets within any guaranteed timespan. Up to five concurrent connections can be set-up between the six router ports and independence between their input and output interfaces allows full-duplex communication across the five channels.

B. Knobs and Monitors

The embedded intelligence module has access to many of the internal signals of the router and processor, called "monitors" in our system. These include signals such as:

- the task IDs of packets routed through the router,
- watchdog signals from the node,
- the current node frequency,
- local temperature sensing,
- signals from intelligence modules of neighbouring nodes.

The intelligence module can also affect several aspects of the router and processor, referred to as "knobs", for example:

- the task the processor node should be running,
- clock Enable for the processor node,
- reset of the processor node,
- node-level frequency scaling (10MHz - 300MHz).

C. Embedded Artificial Intelligence Module

The Artificial Intelligence Module (AIM) implementation used in this work is based on the Xilinx Picoblaze micro-controller [15] for maximum flexibility and simplicity with a small hardware footprint when translating and developing the bio-inspired social insect-intelligence model in hardware as illustrated in Figure 2. The program code is uploaded by the Experiment Controller to allow rapid prototyping of embedded intelligence models. The AIM has access to the internal signals of the router and processor via the monitors, and can also affect several settings and behaviours of the router and processor through the knobs. To facilitate the implementation of the response threshold models, the Picoblaze software platform provides functions for: interfacing to convert between impulse sequences (spike trains) and binary number representation, logical comparators that generate impulses when vector inputs match, and threshold circuits that act as final decision makers. The intelligence models can then be implemented by tying these functions together to produce a response-threshold decision pathway from the monitors through to the knobs.

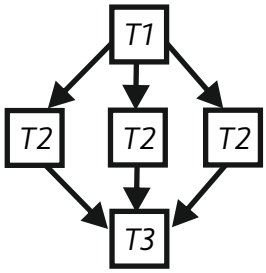

Fig. 3. The fork-join task graph. This has out-tree and an in-tree phase and requires Task 1 nodes to start the shape of the emerged topology. The ratio experimented with is 1:3:1

IV. Emergent Runtime Management Behaviour

For this paper two embedded intelligence schemes based on the *Network Interaction* and *Foraging for Work* models discussed earlier have been implemented on the AIM. We expect that these models should be capable of: *a)* adapting the distribution of tasks around the network from a random task-mapping to a more efficient one and *b)* coping with faults injected into the system by adapting the node task topology such that complete, efficient mapping of the task graphs are restored.

A. Task Allocation

For the experiments a fork-join task graph, shown in Figure 3, was assigned to Centurion with the aim of having as many instances of this task graph as possible (i.e. maximising total many-core throughput of task 3 nodes) [16]. The embedded intelligence was implemented in the following way:

1) Network Interaction: A *single monitor (task of packet routed)* is sufficient for the *Network Interaction* (NI) model and a dedicated thresholder for each task number in the system. Each time a packet is routed an impulse generated by the monitor increases the count for its destination task. Once a task count exceeds its threshold, the node is switched to performing that task and the task counters are reset.

2) Foraging for Work: Foraging for Work (FFW) has a temporal aspect to the model and requires three monitors: *task of packet routed*, *packet routed to internal node*, and *time since sent*. A threshold circuit is used to detecting when a packet deadline comes too close or has lapsed and setting up an appropriate timeout counter. Once this timer expires, the local node switches to the task of the next packet in the routing queue in order to sink and process it locally. Every time a packet is routed internally (i.e. accepted for processing by the node), that impulse is used to reset the task switch

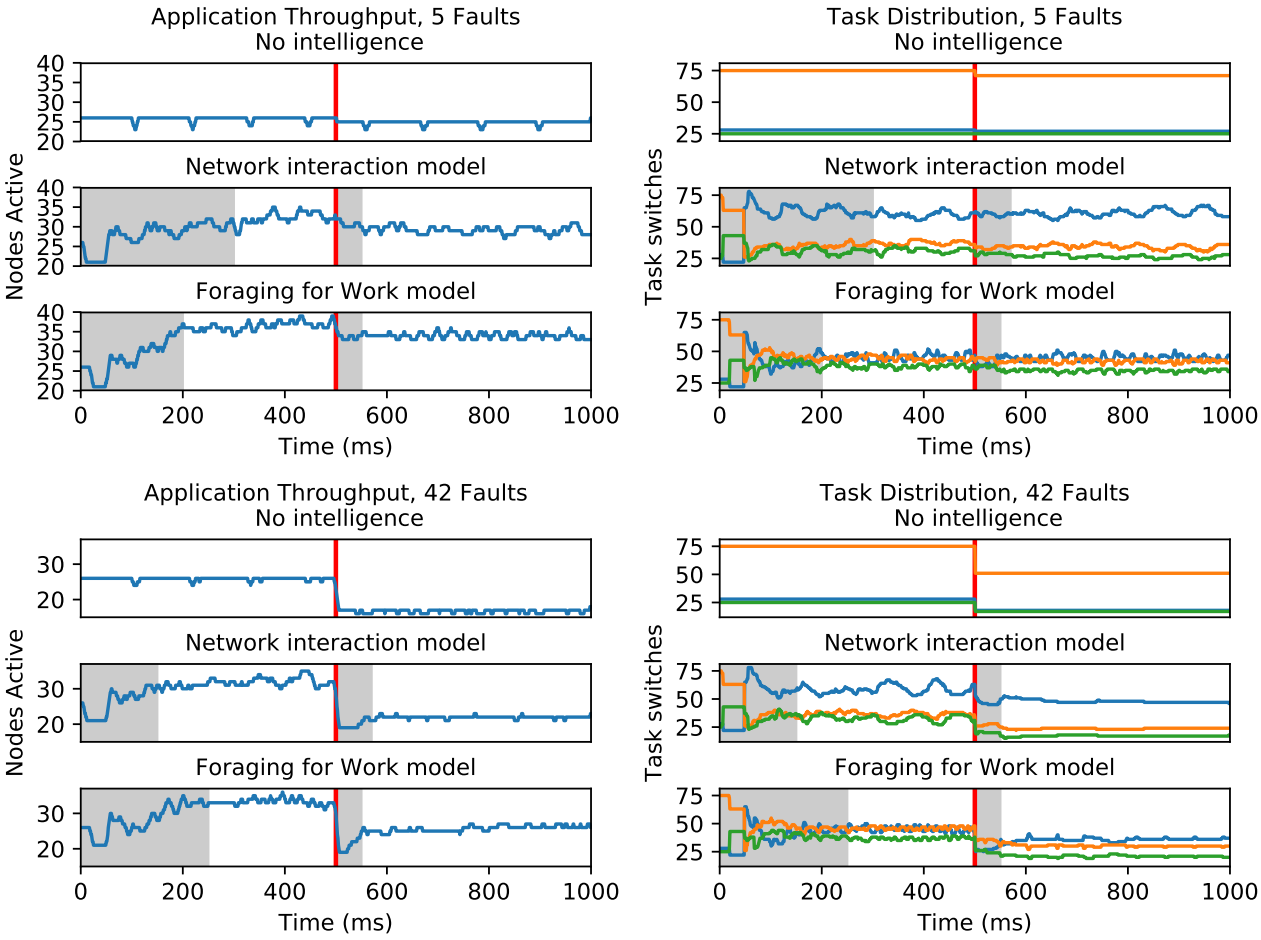

Fig. 4. Results of fault injection experiments for five faults and 42 faults (1/3 of Centurion). In both experiments the systems were started and then left to self-optimise (the shaded area shows the settling period as the task topology adapts). After 500ms the faults are injected and the system resettles into a new task topology. This recovers some of the performance compared to the pre-fault state by reorganising the task topology to reflect the task graph (Figure 3).

timeout. Therefore, as long as a node's current task is well suited to the overall routing and processing requirements, the stream of packets routed internally suppress task switching. For these experiments the task switch timeout is set to 20ms and task 1 (the source task) produces 1 packet every 4ms. This ensures that even the least busy nodes receive several packets to route internally within the 20ms window.

3) Experimental Performance: The first 500ms of the graphs in Figure 4 show the adaptivity of both bio-inspired approaches, *Network Interaction* and *Foraging for Work* in comparison with an implementation using a heuristic fixed routing approach (minimised Manhattan distance). As can be seen from Figure 4, both approaches exhibit a settling phase as the network adapts to the initially random task topology. FFW then enters a steady state (settled) phase that is similar to the performance of the heuristic approach. The NI model also settles into a steady state, but does not achieve the same performance as FFW. Quantitative results for 100 independent runs of each model are summarised in Table I.

B. Fault Tolerance

Both bio-inspired task allocation models support a degree of inherent fault tolerance against node faults. This is an emergent property of this kind of dynamical decision-making system and requires no additional processes or knobs and monitors. In this work our fault model considers multiple node failures, which will change the balance of packets being routed around the network and so requires the task allocation to automatically adapt. Table II shows quantitative results for different numbers of faults injected. The recovery time after fault injection and the performance achieved after recovery are compared with the pre-fault case. Typical examples are shown in Figure 4 where at 500ms a proportion of the nodes develop faults and fail. In the first experiment 5 faults are injected, representing a series of local application faults, whilst in the second experiment 42 faults (1/3 of of the 128 Centurion nodes) fail, e.g. representing a failure of a global clock buffer, other critical global circuitry, or a thermal issue.

The recovery phase can be seen in Figure 4, starting immediately after faults occur as the intelligence adapts to

TABLE I

PERFORMANCE REACHED—RELATIVE TO HIGHLIGHTED CASE—AFTER SETTLING TIME WITHOUT FAULT INJECTION. SHOWN ARE MEDIAN (Q2) AND 25TH/75TH PERCENTILES (Q1/Q3) FOR 100 INDEPENDENT, RANDOMLY INITIALISED RUNS OF EACH EXPERIMENT.

	Settling Time			**Relative Performance**		
	Q1	**Q2**	**Q3**	**Q1**	**Q2**	**Q3**
No Intelligence	6	6	7	96%	**100%**	103%
Network Interaction	12	56	58	93%	102%	108%
Foraging For Work	10	86	170	105%	114%	124%

the new task landscape and starts to route around the failed nodes. Once the recovery period has settled, the system settles into a steady state where it has recovered to an overall lower performance than before due to the loss of a number of nodes. However, within the limits of reduced resources performance has recovered and a task structure required for data to effectively reach task 3 nodes has been restored. Again, FFW outperforms the network-interaction model in terms of performance recovery.

V. DISCUSSION AND CONCLUSION

The results from the dynamic task allocation experiments have shown that both bio-inspired runtime management models exhibit emergent adaptive properties that are useful in large fault cases, hence, indicate a degree of inherent scalability. Whilst such global failure cases may be mitigated through circuit hardening or additional redundancy, this fault case is also relevant for high-processing power devices that require the parallel throughput of a many-core system but are deployed in remote application scenarios with requirements of autonomous operation and long lifetime. As faults develop in the field over the lifetime of a device the emergent task allocation can adapt the task topology to achieve a graceful degradation of system performance that may allow a device to operate for longer in its deployed environment.

Several further improvements to the embedded intelligence are conceivable that would go beyond the models presented here. For example, adaptive and multi-cast routing would allow greater throughput as it exploits the inherent parallelism of a task graph. Whilst there is not a direct mapping of a social insect model for this kind of adaptive routing, the derived bio-inspired stimulus-response threshold model could be used to allow the embedded intelligence to make decisions on the destination output port of incoming packets. Many of the models shown in Figure 1 feature mechanisms for adaptive thresholds, which are not yet considered in this paper.

Indeed a next step is to embed these threshold pathways in a form that is specifically hardware efficient, i.e. with a small footprint and ultra-low power consumption. The impulse/count nature of these pathways maps well to digital hardware fabrics such as FPGAs—but also to dedicated analogue implementations with potentially ultra-low hardware overhead in terms of area or power—and can be used to implement the bio-inspired social insect intelligence models presented in this paper in low-level hardware. This will provide a pathway for creating a design methodology for a generic social insect-inspired RTM subsystem.

TABLE II

PERFORMANCE REACHED—RELATIVE TO HIGHLIGHTED CASE—AFTER RECOVERY TIME FOLLOWING FAULT INJECTION AT 500MS. SHOWN ARE MEDIAN (Q2) AND 25TH/75TH PERCENTILES (Q1/Q3) FOR 100 INDEPENDENT, RANDOMLY INITIALISED RUNS OF EACH EXPERIMENT.

		Recovery Time			**Relative Performance**		
	Faults	**Q1**	**Q2**	**Q3**	**Q1**	**Q2**	**Q3**
No Intelligence	**0**	–	–	–	96%	**100%**	103%
	2	3	3	19	95%	98%	102%
	4	3	3	17	94%	96%	100%
	8	3	3	5	88%	93%	98%
	16	3	3	3	79%	84%	89%
	32	3	3	3	63%	69%	75%
Network Interaction	**0**	–	–	–	98%	108%	117%
	2	3	30	160	94%	104%	113%
	4	3	30	153	92%	102%	109%
	8	3	19	141	85%	97%	105%
	16	3	3	76	76%	85%	92%
	32	3	3	3	52%	64%	74%
Foraging For Work	**0**	–	–	–	117%	129%	141%
	2	3	29	136	115%	125%	140%
	4	3	36	177	112%	124%	136%
	8	3	53	175	109%	118%	129%
	16	3	81	276	100%	107%	122%
	32	3	3	272	81%	89%	101%

REFERENCES

[1] C. Emmeche, S. Køppe, and F. Stjernfelt, "Explaining emergence: Towards an ontology of levels," *Journal for General Philosophy of Science*, vol. 28, no. 1, pp. 83–117, Jan 1997.

[2] S. R. Brown, "Emergence in the central nervous system," *Cognitive neurodynamics*, vol. 7, no. 3, pp. 173–195, 2012.

[3] M. Rowlings, A. M. Tyrrell, and M. A. Trefzer, "Social-insect-inspired networking for autonomous fault tolerance," in *IEEE Symp. Series on Comp. Intel. (SSCI)*, December 2015.

[4] ——, "Social-insect-inspired adaptive task allocation for many-core systems," in *IEEE World Cong. on Comp. Intel. (WCCI)*, July 2016.

[5] E. H. Davidson, "Emerging properties of animal gene regulatory networks." *Nature*, vol. 468, no. 7326, pp. 911–20, dec 2010.

[6] T. Kuyucu, M. A. Trefzer, J. F. Miller, and A. M. Tyrrell, "An investigation of the importance of mechanisms and parameters in a multicellular developmental systems," *IEEE Trans. on Evo. Comp.*, vol. 15, no. 3, pp. 313–345, Jun. 2011.

[7] M. K. Gould and H. P. de Koning, "Cyclic-nucleotide signalling in protozoa," *FEMS Microbiol. Rev.*, vol. 35, no. 3, pp. 515–541, may 2011.

[8] J. B. Jackson, "A functional biology of clonal animals," *Trends Ecol. Evol.*, vol. 5, no. 12, pp. 425–426, dec 1990.

[9] D. Gordon, *Ant Encounters: Interaction Networks and Colony Behavior*. Princeton University Press, 2010.

[10] N. R. Franks, S. C. Pratt, E. B. Mallon, N. F. Britton, and D. J. T. Sumpter, "Information flow, opinion polling and collective intelligence in house-hunting social insects." *Philos. Trans. R. Soc. Lond. B. Biol. Sci.*, vol. 357, no. 1427, pp. 1567–83, nov 2002.

[11] S. Beshers and J. Fewell, "Models of division of labor in social insects," *Annu. Rev. Entomol.*, 2001.

[12] C. Tofts, "Algorithms for task allocation in ants. (A study of temporal polyethism: Theory)," *Bull. Math. Biol.*, vol. 55, no. 5, pp. 891–918, sep 1993.

[13] D. Gordon, B. Goodwin, and L. Trainor, "A parallel distributed model of the behaviour of ant colonies," *J. Theor. Biol.*, 1992.

[14] Xilinx Inc, "MicroBlaze Micro Controller System v1.4 (PG048)," 2013.

[15] ——, "PicoBlaze 8-bit Embedded Microcontroller User Guide (UG129)," 2013.

[16] Y. K. Kwok and I. Ahmad, "Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, 1996.