AN EFFICIENT HARDWARE IMPLEMENTATION OF ADAPTIVE PACKET ROUTING BASED ON ANT ALGORITHM

Jin-Ho Ahn and Sungho Kang

Department of Electrical and Electronic Engineering
Yonsei University, 134 Shinchon-Dong Seodaemoon-Gu, Seoul, Korea
sominaby@soc.yonsei.ac.kr, shkang@yonsei.ac.kr

ABSTRACT

In this paper, we present a new hardware architecture for adaptive packet routing based on ant algorithm, specially AntNet, to efficiently and effectively handle ant-like mobile agents to find the best route toward destination in a network. Simulation results show that the proposed architecture is suitable and efficient to realize AntNet-based routing.

Keywords – ant, mobile agents, adaptive routing

1. INTRODUCTION

Stability, scalability and distributed process capacity are necessities for the future networks, but until now they are not much considered as compared to others related to the amount of packet throughput. So, artificial model inspired by social insect behavior attracts the public attention. The characteristic of social insect behavior can be summarized to collective behavior of individuals. They make swarms, and perform task or make decisions such as nest building, food foraging, and so on as if they have intelligence. Ant Colony Optimization (ACO) is an artificial model based on social insect behavior, particularly ants, to solve optimization problems [1]. Communication network is also a major application of ACO. The routing method, based on ACO mechanisms, has many merits such as adaptability, survivability, and self-organization. The network system that includes these features can be easily adapted to new environments, and be stable in unpredictable and dynamic conditions, and do all by itself without central controls. Di Caro and Dorigo proposed the AntNet applied to routing in packet switched data networks [2]. The AntNet uses agents to investigate appropriate network conditions. There are two types of ants: forward ants and backward ants. A forward ant collects network information on the way of going to a destination node with the same priority as normal data packets. When the forward ant reaches the destination node, it becomes a backward ant. The backward ant returns to the source node that the forward ant was generated. While returning to the source, the backward ant updates the outdated routing information of the nodes that the forward ant has visited with information collected by the forward ant. The AntNet is superior to other routing algorithms in many respects under various traffic distributions [2]-[3], and improved versions of AntNet-based algorithm are proposed continuously [3]-[4]. But, its efficiency is fully dominated by the quality of collected information. An ant's trip time, same as routing time, is the most important factor to determine it. Therefore, it is necessary to minimize and to regularize the processing time of an ant packet in each node to get accurate and pure trip time of ants. Also, a rapid update of routing information is essential to select a correct routing path of normal data packets to maximize the throughput. According to consider these requisites, the block to process ants should have a hard-wired form like forwarding or classification of packet in a router as possible.

In this paper, we present a new hardware architecture to realize an AntNet-based routing in practical system on a chip application. The original AntNet algorithm is modified to fit a hard-wired form with minimizing the performance degradation and hardware overhead. The modified AntNet algorithm for hardware implementation is compared with the original algorithm on the various traffic patterns, and its hardware implementation results are also shown. We first introduce the proposed hardware architecture and its detail descriptions in section 2, and the performance evaluation results are given in section 3. Finally, the conclusion of this paper is presented in section 4.

2. PROPOSED ARCHITECTURE

1. Ant Packet Format

An ant is a mobile-agent to collect network information such as link conditions and the amount of traffic.

Currently, we define an ant packet by 160byte size. But, it can be extended according to supported applications like QoS service. The organization of an ant is shown in Fig. 1, and its detailed descriptions are as follows:

- *Type* indicates whether the packet is a forward ant or a backward ant.
- *sNode* denotes the address of the start node that an ant is produced.
- *dNode* denotes the address of the end node to which an ant goes.
- *pNodeOdr* indicates the order of the nodes that an ant visits. The number starts from zero.
- *tNodeNum* indicates the total number of the nodes that an ant visits.
- *intNode* denotes the address of the nodes that an ant visits.
- *visTime* indicates the time that an ant arrives at the node.

To fix an ant's length, we limit the total number of the nodes that an ant can visit to 12. The ant's information except a *dNode* is set automatically on the way of routing, but the *dNode* is initiated and fixed at a starting node. The *dNode* can be determined by a manual mode or random mode. The time information at all nodes is synchronized by a network time protocol such as SNTP (Simple Network Time Protocol). We currently use the protocol based on SNTP ver. 4 [5], which configures the time information by 64bit size and 200ps resolution.

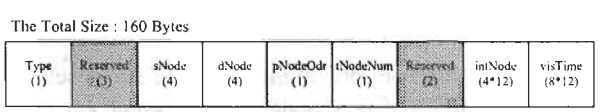


Figure 1. Ant Packet Structure

2. Overall Architecture

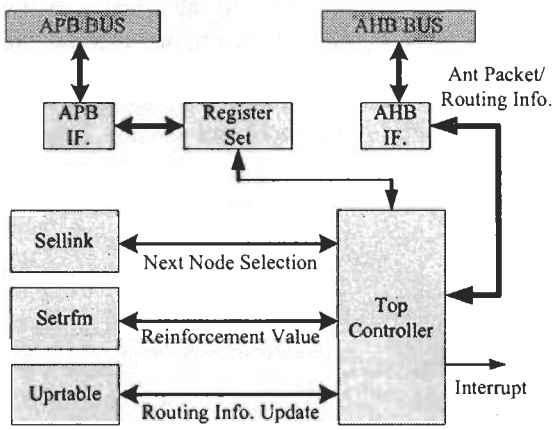


Figure 3. Top Block Diagram of SE

An example of SoC application including proposed architecture is shown in Fig. 2. In Fig. 2, we call the proposed architecture "Stigmergy Engine (SE)" which means a dedicated hardware for the modeling of ecology. A top block diagram of SE is shown in Fig. 3. The structure is designed only to handle ant packets defined in Fig. 1 and optimized to minimize and regularize the processing time of an ant. It consists of 4 major sub-blocks excluding external interface blocks. We develop the SE to fit system on a chip peripheral based on ARM920T microprocessor and AMBA bus specification ver. 2.0. The SE is an AMBA master module, and connects to AHB bus. Therefore, all address and data have 32bit width. A user can control all sub-blocks with register configuration using a controller including APB interface such as UART.

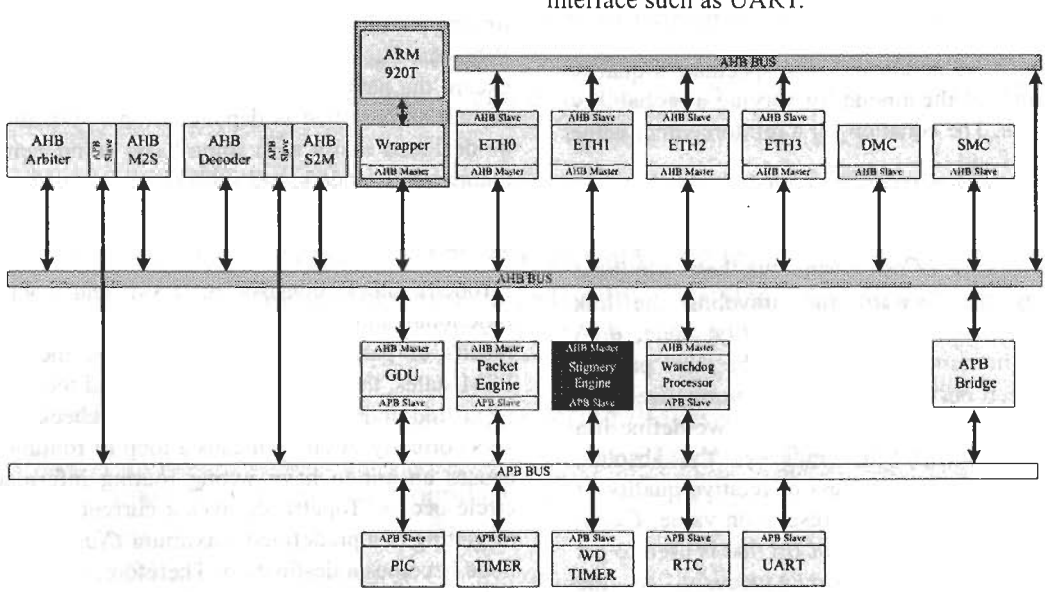


Figure 2. An Example of SoC Application including Proposed Architecture

A. Routing Table and Local Traffic Model

The basic structures of a routing table and a traffic model are the same as those of the original AntNet. We set a probability to a byte size for efficient calculations. We do not use a mean or variance of trip time until now; therefore local traffic model only includes the best trip time information of ants.

B. Sellink

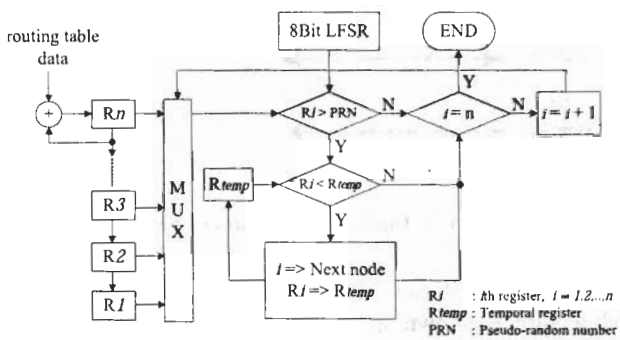


Figure 4. Selection of a Next Node

Sellink is a unit to select a next node using the probability values of a routing table at a current node. Its detailed procedure is depicted in Fig. 4. The input probabilities of routing table are accumulated to registers in order. The registers are compared with a pseudo-random value produced by a LFSR (Linear Feedback Shift Register) one by one. After all registers are compared, a next node is determined. Finally, the selected node is translated to a real IP address through a predefined address table.

C. Setrfm

A reinforcement value allows us to speculate a quality of each link, and set the amount of varying a probability of a routing table. The equation for a reinforcement value, r, is shown in (1) and (2).

$$r' = norm(curCost - bCost) \quad (1)$$

$$r = (255 - r') / C_{res} \quad (2)$$

In equation (1), a *bCost* means the best trip time experienced by the forward ants traveling the link between a next node, n, and a destination node, d. A *curCost* means the current trip time on the same path. A difference between *curCost* and *bCost* is normalized to a predefined reinforcement level. Currently, we define it a byte size level. But, r' just indicates the absolute difference of a trip time regardless of relative quality of trip time. To solve this problem, resolution value, C_{res} in (2), is induced to weigh r'. If the *bCost* that is used to get r' increases, C_{res} decreases simultaneously. As the amount of varying C_{res} in proportion to *bCost* is variable according to practical network conditions, we parameterized it to change easily. A calculation flow of Setrfm block is shown in Fig. 5. If *bCost* is not changed during the given time threshold, it is initialized. This helps to raise the reliability of *bCost*. The limitation of *curCost* by size threshold also provides the reduction of calculation time and hardware size.

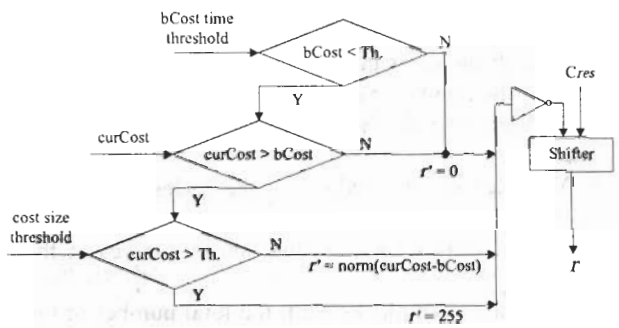


Figure 5. Calculation Flow of a Reinforcement Value

D. Uprtable

A routing table is refreshed in this block according to an ant's routing information and a reinforcement value. The probability entries that have the same destination node are updated using a following rule.

$$P_{fd} \leftarrow P_{fd} + (r * (255 - P_{fd}) / 256)$$

$$P_{nd} \leftarrow P_{nd} - (r * P_{nd} / 256) \quad (3)$$

$$n, f \in N, n \neq f$$

where $r \in (0,1)$ is a reinforcement value and N is a set of neighborhood nodes. f is a node that a current ant has visited, and n means other neighborhood nodes except f. d is a destination node. P_{fd} indicates a probability of the routing path from f to d, and it increases in proportion to r. Whereas, P_{nd}, the probabilities of routing paths to reach d except the path via f in a current node, decreases by the certain amount that is dependent on r. Equation (3) can be designed easily with some basic components such as comparators, adders, and shifters.

E. Topctrl

Topctrl block consists of FSM and controls local functional blocks and other glue logics through the results of parsing ant packets. Before the decision of FSM states, the occurrence of circle and the total number of visited nodes (*tNodeNum*) must be checked to handle ants correctly. A circle means a loop of routing path, and causes an ant to have wrong routing information. If a circle occurs, Topctrl removes a current ant. In the case of getting to a predefined maximum *tNodeNum*, a current node becomes a destination. Therefore, a current forward

ant changes into a backward ant.

3. PERFORMANCE EVALUATION

The modified algorithm for hardware implementation is evaluated through the comparison with the original AntNet on topology presented in Fig. 6 (a). The metric of comparison is the variation of probabilities within a routing table in each node. We mainly monitor the course of routing probabilities in a source node. In Fig. 6 (a), a source is node 0, and a destination is node 12. We assume that network traffic is equally distributed except when the update number is located between 500 and 1000. The update number means the number of changing routing probabilities. If the update number arrives 500, the network traffic of the path via 0-3-5-6-7-9-12 diminishes until the update number reaches 1000. As shown in Fig. 6. (b) and (c), both algorithms can select the proper paths according to the traffic variations.

The RTL design of the modified AntNet is coded using verilog HDL, and verified by Seamless HW/SW co-verification environments of Mentor Graphics. The RTL simulation result on topology presented in Fig. 6 (d) is shown in Fig. 6. (e). If the traffic of the path via 0-1-3 is less than the opposite path, P_{13} increases gradually as ants repeat their action. But, P_{23} decreases inversely. The gate counts of the proposed architecture excluding register sets are about 60K with 100MHz operating clock and TSMC 0.25μm CMOS technology. The used external memory is $(m*n+4*n)$ bytes, where m is the number of neighborhood nodes, and n is that of destination nodes.

4. CONCLUSION

In this paper, we propose a hardware architecture inspired by social insect behavior, especially AntNet, for adaptive routing. We develop the architecture to apply to network devices based on SoC environments such as a network processor. The modified AntNet for hardware implementation shows nearly the same performance as the original algorithm on various network environments. The results of simulation and synthesis of RTL design also show that the architecture is efficient and optimal to realize an AntNet-based routing function.

REFERENCES

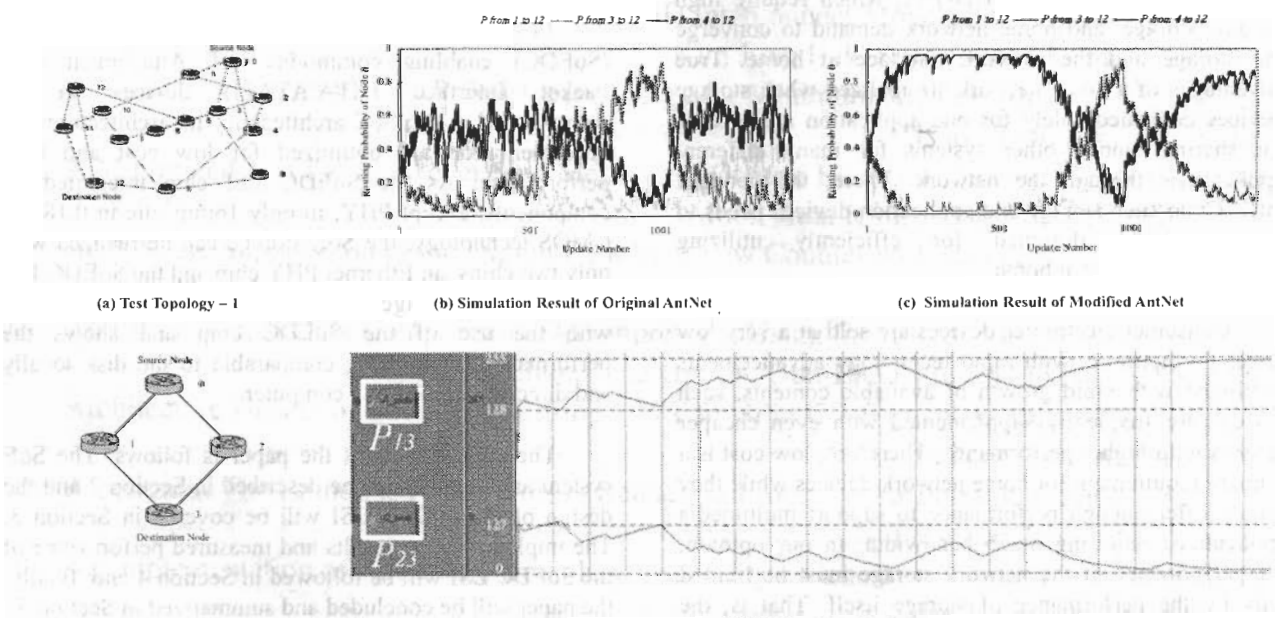
[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for Optimization from Social Insect Behavior," *Nature*, Vol. 406, July 2000, pp. 39-42.

[2] G. Di Caro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks," *Journal of Artificial Intelligence Research 9*, Dec. 1998, pp. 317-365.

[3] B. Baran and R. Sosa, "A New Approach for AntNet Routing," *Proc. of Ninth International Conference on Computer Communications and Networks*, Oct. 2000, pp. 303-308.

[4] S. Liang, A. N. Zincir-Heywood, and M. I. Heywood, "The Effect of Routing under Local Information using a Social Insect Metaphor," *Proc. of the Congress on Evolutionary Computation*, Vol. 2, May 2002, pp. 1438-1443.

[5] RFC-2030 : *SNTPv4 for IPv4 and IPv6 and OSI*, October 1996.



(a) Test Topology – 1 (b) Simulation Result of Original AntNet (c) Simulation Result of Modified AntNet

(d) Test Topology - 2 (e) RTL Simulation Result

Figure 6. Simulation Results