VLSI Design of AntNet for Adaptive Network Routing

Jin-Ho Ahn, Jae Seuk Oh, Sungho Kang

Department of Electrical and Electronic Engineering
Yonsei University
134 Shinchon-Dong Seodaemoon-Gu, Seoul, Korea

ABSTRACT

The AntNet is an indirect communication algorithm for adaptive routing in a network. A unit used for adaptive routing is called an agent. The AntNet consists of agents whose behaviors look like those of ants, and has many characteristics such as adaptability, survivability, and self-organization. These features are very attractive to today's network environment. The function of an agent dominates the efficiency of the AntNet, and has deep relation with the processing time of an agent in a router. Therefore, it is necessary to handle agents using a dedicated hardware for the minimization and regularity of the processing time. In this paper, the efficient architecture that can be easily adapted to a hard-wired form for an AntNet-based routing is developed, and verified by the comparisons with the original AntNet and RTL-level simulations. The results of simulation show that the proposed architecture is suitable and efficient to realize adaptive routing based on the AntNet.

1. INTRODUCTION

Internet packet data has increased rapidly in recent years. Therefore, it is not so hard to imagine serious network data congestion in the near future. Gigabit routers and the effective use of resources by differentiating services could somewhat alleviate such a problem but it cannot be solved absolutely. Intelligent routing may be another solution for the aforementioned problem. Especially, routing methods, based on bio-inspired mechanisms, have many merits such as adaptation, survival capacity, and self-organization [1]. The system that includes these features can be easily adapted to new environments, and be stable in diverse and dynamic conditions, and do all by itself without central controls. The AntNet is an adaptive and distributed routing algorithm applying ant colony organization [2]. It mimics the activities of social insects such as making swarms and communicating with each other by chemical substance called pheromone. An ant is a kind of an agent packet used to investigate network conditions with a round trip from a source to a destination node. 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. 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 is generated. While returning to the source, the backward ant updates outdated routing information of the nodes that the forward ant has visited with information collected by the forward ant. In this way, ants communicate each other about the quality of routing paths through information stored in the nodes. It is proved that the AntNet is superior to other algorithms in many respects such as routing performance, adaptability, and stability under most of traffic distributions [2,3,4]. But, its efficiency is fully dominated by quality of collected information. An ant's trip time is the most important factor to determine it. Therefore, it is necessary to minimize and regularize the processing time of an ant packet in each node to get accurate and pure trip time. Also, a rapid update of routing information is essential to select a correct routing path of normal data packets. 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.

This paper presents a hardware architecture to realize an AntNet-based routing in practical environments. The original AntNet algorithm is modified to fit a hard-wired form with minimizing the performance degradation and hardware overhead. The proposed architecture and its detail descriptions are presented in section 2 and 3 respectively, and the performance evaluation is given in section 4. The conclusion of this paper is presented in section 5.

2. PROPOSED ARCHITECTURE

The proposed architecture is mainly based on the AntNet algorithm. The block diagram of the proposed architecture is shown in Fig. 1. It consists of four major sub-blocks excluding external interface blocks. The major blocks are designed only to handle ant packets defined in Fig. 2. The detailed descriptions of an ant 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.

In contrast to the original AntNet, we can remove stack memories at all nodes using the proposed ant's structure including all the data needed for evaluating a routing path. An ant packet's length is fixed to eighty bytes. It is very useful to fix ant's size in many cases. Currently, we limit the total number of the nodes that an ant visits to ten. 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. In a manual mode, a user can directly configure the *dNode* with a predefined address table. Otherwise, the *dNode* is randomly determined according to the destination address information of normal data packets.

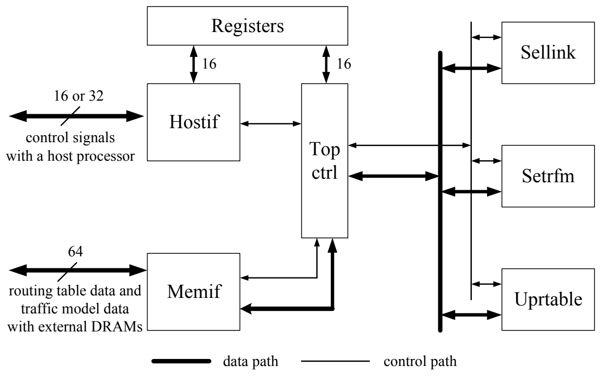

Registers
16
16
16 or 32
control signals with a host processor
Hostif
Top ctrl
Sellink
Setrfm
64
routing table data and traffic model data with external DRAMs
Memif
Uprtable
data path
control path

Figure 1. Block Diagram of the Proposed Architecture

The Total Size : 80 Bytes

Type	sNode	dNode	pNodeOdr	tNodeNum	intNode	visTime
(1)	(4)	(4)	(1/2)	(1/2)	(4*10)	(3*10)

Figure 2. Ant Packet Structure

3. FUNCTIONAL DESIGN

3.1. Routing table and Traffic model

The basic structures of a routing table and a traffic model are nearly the same as those of the original AntNet. There are two data structures at each node, one is a local traffic model, and the other is a routing table. A local traffic model is a kind of statistical model that represents a traffic distribution over networks. A routing table contains the entries, which express the probabilities to choose a next node. A next node is selected stochastically in proportion to the probabilities of a routing table. We set a probability to one byte size for efficient calculations. Therefore, each entry has a value whose range is from 0 to 255. A routing table consists of two-dimensional structures. Each row represents an adjacent node, and each column represents a destination node. If a node has ten neighborhood nodes and twenty destination nodes, the node needs to have a routing table whose size is (10, 20). The summation of all probabilities per a column must be 255. The total number of destination nodes can be limited to keep down memory usage, and to lead an efficient update of routing tables.

3.2. Sellink

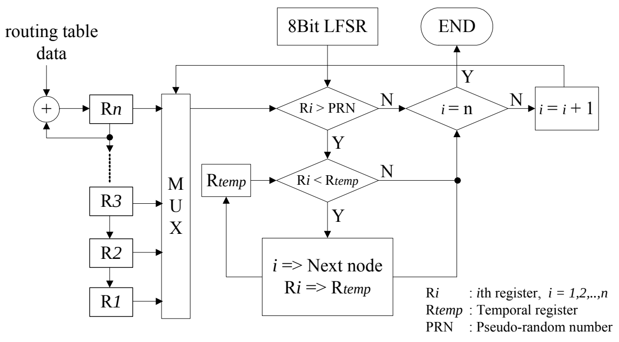


Figure 3. Selection of a Next Node

Sellink is a unit to select a next node using the probability values of a routing table at a current node. Its detailed functional procedure is depicted in Fig. 3. First, Sellink requests the probability data of a routing table to external memory, and carries the data from the memory. The data is accumulated to registers in order. If a current node has ten neighborhood nodes connected directly, it is needed to have ten byte registers. The accumulated 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.

3.3. Setrfm

It is essential to estimate the relative goodness of each link for adaptive routing in dynamic network environments. A reinforcement value allows us to speculate a quality of each link, and set the amount of varying a probability of a routing table. Though several factors are induced to get a reinforcement value [2,5], we only use the trip time of ants. The equation for a reinforcement value, *r*, is shown in (1) and (2).

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

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

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 *bCost* is a local traffic model data stated previously in section 3.1. In equation (1), a difference between *curCost* and *bCost* is normalized to a predefined reinforcement level. Currently, we define it a byte size level. Therefore, the difference, *r'* in (1), has a value from 0 to 255. But, *r'* just indicates the absolute difference of a trip time regardless of relative quality of paths. To solve the 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 somewhat heuristic, we parameterized it to change easily. A calculation flow of Setrfm block is shown in Fig. 4. If a *bCost* is not accessed during the given time threshold, it is initialized. This helps to raise the reliability of a *bCost*. The limitation of a *curCost* by size threshold also provides the reduction of calculation time and hardware size.

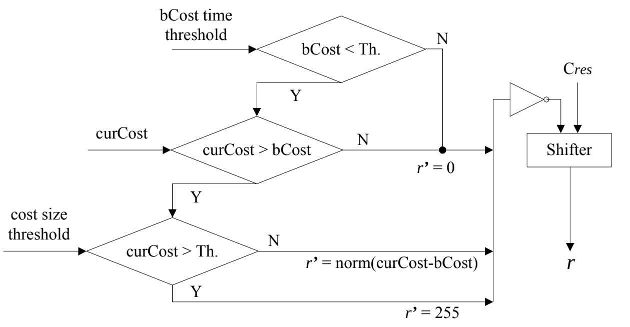


Figure 4. Calculation Flow of a Reinforcement Value

3.4. 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.

$$P_{fd} \leftarrow P_{fd} + (r * (255 - P_{fd}) / 256)$$
$$P_{nd} \leftarrow P_{nd} - (r * P_{nd} / 256) \quad (3)$$
$$n, f \in N, n \neq f$$

In equation (3), *r* 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, decrease by the certain amount that is dependent on *r*. A procedure to accomplish equation (3) is shown in Fig. 5.

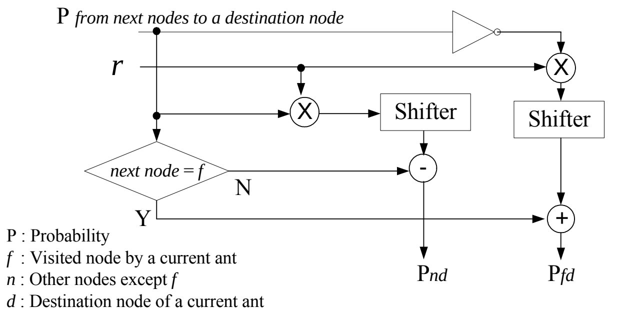

P from next nodes to a destination node
r
X
X
Shifter
Shifter
next node = f
N
-
Y
+
P : Probability
f : Visited node by a current ant
n : Other nodes except f
d : Destination node of a current ant
Pnd
Pfd

Figure 5. Updating Procedure of a Routing Table

3.5. Topctrl

Topctrl block controls local functional blocks and other glue logics through the results of parsing ant packets. A control flow consists of FSM including five stages.

- Start: If the ant function is allowed to activate, a forward ant is generated, and transmitted to a next node.
- Forwarding: The forwarding operation happens if a current node is not a stating node. *pNodeOdr* and *tNodeNum* of a current forward ant increase by one. Also, current node address and time, *intNode* and *visTime*, are inserted into the ant. Then the new ant is sent to a next node selected at a Sellink block.
- Destination: If a current node is a destination node, a forward ant changes into a backward ant as soon as *pNodeOdr* and *tNodeNum* increase. The backward ant starts to return to a source node with the routing information.
- Backwarding: A backward ant traces along the nodes, which a forward ant has visited, until it arrives at a source node. On the way of returning to the source node, a routing table and local traffic model at the visited nodes are changed to some degree according to the routing information.
- Source: If a backward ant reaches a source node, the ant vanishes after updating a routing table and traffic model of the node.

Before the decision of aforementioned states, there are two checkpoints to select proper states. One is a circle or not. A circle means a loop of routing path, and causes an ant to have wrong routing information. Therefore, circle must be checked by the comparison of current node address with visited node address. If a circle occurs, Topctrl removes a current ant, rather than sending it back to a previous node. The reason for that is certainly for hardware efficiency. The other checkpoint is the total number of visited nodes (*tNodeNum*). In section 2, we stated that the size of *tNodeNum* is limited to ten. If *tNodeNum* reaches the predefined maximum number, a current node becomes a destination. Therefore, a current forward ant changes into a backward ant. We think that the limitation of the number of visited nodes will not induce serious degradation of performance because many ants are generated periodically at all nodes enough to compensate the loss due to it. The control flow is depicted in Fig. 6 to clarify the above explanations.

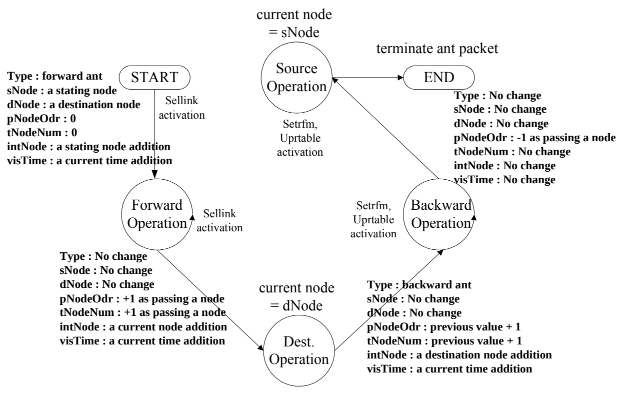
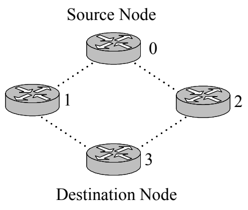


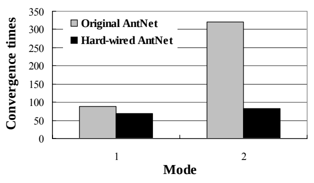
Figure 6. FSM of Topctrl

4. PERFORMANCE EVALUATION

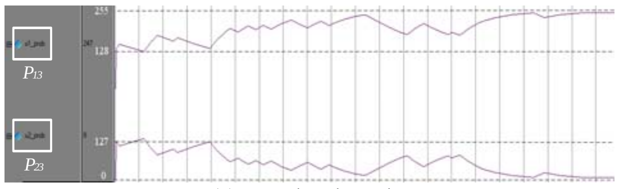
First, we compare the proposed routing algorithm with the original AntNet on a topology presented in Fig. 7 (a). The C-level simulation is used to compare results for satisfying identical conditions. The comparison results are shown in Fig. 7 (b). The first mode in Fig. 7 (b) is a case where the routing time between nodes is equally distributed except for one path, and the second is to have the same routing time for all paths. According to the simulation results, we can confirm that the routing scheme of the proposed structure is the same or superior to the original under static environments. Convergence time means the number of an ant's routing cycles used to choose a routing path that has a predefined dominant probability over others. And then, RTL-level simulation results in the case that a routing time from node 0 to node 1 is shorter than that of others are given in Fig. 7 (c). In real environments, time information at all nodes is synchronized by a network time protocol such as SNTP. However, we imposed it directly using a testbench file. P_{13} (the upper graph in Fig. 7 (c)) means the probability of routing path via node 1, and P_{23} (the lower one) is the probability of routing path via node 2. P_{13} increases gradually as ants repeat their action. But, P_{23} decreases inversely. If P_{13} and P_{23} reach a steady state, the state is preserved unless routing time changes. Most of simulations with diverse routing times give similar results except for the time of convergence to a steady state. The gate counts of core blocks excluding external interface logics are about 160K under 100MHz operating clock and TSMC 0.25μm CMOS technology. The used memory is (m*n+3*n)bytes, where m is the number of neighborhoods nodes, and n is that of destination nodes.



(a) Test node topology



(b) Comparison results



(c) RTL-level results

Figure 7. Simulation Results

5. CONCLUSION

In this paper, we presented a hardware architecture based on the AntNet. It can provide the high reliability of routing information by reducing and regularizing the processing time of ants. Simulation results show that the architecture selects a proper path not only faster, but also more uniform than the original AntNet under various environments. As there are some heuristic points in the AntNet until now, we have developed the architecture that can be parameterized and modified easily for future works.

6. REFERENCES

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

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

[3] K.M. Sim and W.H. Sun,"Multiple Ant-Colony Optimization for Network Routing", *Proc. of the First International Symposium on Cyber Worlds*, pp. 277-281, Nov. 2002.

[4] Y. Yang, A. N. Zincir-Heywood, M. I. Heywood, and S. Srinivas, "Agent-Based Routing Algorithms on a LAN", *Proc. of the IEEE Canadian Conference on Electrical & Computer Eng.*, vol. 3, pp. 1442-1447, May 2002.

[5] M. Dorigo, M. Zlochin, N. Meuleau, and M. Birattari, "Updating ACO Pheromones using Stochastic Gradient Ascent and Cross-Entropy Methods", *Proc. of the EvoWorkshops2002*, LNCS 2279, pp. 21-30, Springer, 2002.