An Agent-based Routing System for QoS Guarantees

Kazumasa OIDA* & Masatoshi SEKIDO**

***ATR Adaptive Communications Research Laboratories,**
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0288 Japan
oida@acr.atr.co.jp
****Toyohashi University of Technology,**
1-1 Hibarigaoka, Tempaku-cho, Toyohashi, 441-8580 Japan
sekido@avenue.tutics.tut.ac.jp

ABSTRACT

This paper describes a new agent-based routing system (ARS) for a datagram network. The ARS supports QoS routing, resource reservation, and admission control functions by using ant-like agents. The ARS works under the condition that every node in a network supports the weighted fair queueing algorithm and that network users require two service class deliveries: datagram and real-time flow. The ARS efficiently allocates network resources according to user requests. The simulation results show that more than 70% of network resources are effectively utilized and less than 10% of the resources are unnecessarily reserved when the load of user requests exceeds the network's capacity.

1 Introduction

This paper describes a new agent-based routing system for a datagram network, called ARS, which supports quality-of-service (QoS) routing, resource reservation, and admission control functions by using ant-like agents. The objective of the ARS is to achieve high resource utilization and to reduce user contention for network resources.

The ARS network supports both best-effort packets and packets with QoS guarantees; for this purpose, all transmission capacities in a network are split between the two classes according to the weighted fair queueing (WFQ) algorithm [1, 2]. The ARS supports two QoS commitments: bandwidth and hop-count. Bandwidth and hop-count are useful in that if a flow source is characterized by a leaky bucket [3, 4], the bound on the end-to-end delay and the delay jitter of the flow can be determined by the allocated bandwidth and the hop-count of the route [5, 6]. As a result, delay and jitter constraints can be mapped to bandwidth and hop-count constraints.

Recent QoS-based routing algorithms can be classified as message-passing based routing algorithms. These algorithms find a feasible path that satisfies a set of QoS constraints based on the "messages" exchanged among nodes [7]. This class of routing algorithms has a tendency to temporarily overuse bandwidth and processing resources on a network. For example, once the path computation with a Dijkstra or (distributed) Bellman-Ford algorithm [8] begins, it continues until feasible paths are obtained if they exist. In addition, link-state routing algorithms [9]-[11] periodically broadcast (or multicast) messages (link state advertisements) onto a network.

The ARS, on the other hand, is classified as a mobile agent-based routing algorithm [12]-[14]. In this system, a colony of artificial ants cooperatively discovers feasible paths. AntNet [15] is the first datagram routing algorithm of this class. The ARS supports an extended version of the AntNet, which has bandwidth and hop-count constrained routing capabilities. It is important that the ARS's utilization of bandwidth and processing resource does not sharply fluctuate, since ants are injected at regular intervals into a network and the processing time of each ant is short. Furthermore, the ARS can support the global admission control mechanism by making each ant gather resource state information on its way. This mechanism efficiently controls network-wide resource usage; the efficiency arises from the fact that the ARS always maintains the latest resource states of the shortest delay paths. This is because most ants select the shortest delay paths based on the positive feedback mechanism, so that the information about the paths is frequently updated by these ants.

The rest of the paper is organized as follows. Section 2 describes the operation environment and resource reservation mechanism of the ARS. Section 3 discusses the extended version of the AntNet. Section 4 describes the admission control mechanism. Section 5 details the efficiency of the ARS. Finally, Section 6 concludes the paper.

2 Operation Environment and Resource Reservation

The ARS works in a datagram network under two conditions. First, network users require two service class deliveries: datagram and real-time flow. A datagram is a delay-insensitive packet, while a real-time flow is a sequence of delay-sensitive packets. Second, every node in the network supports the WFQ algorithm. The WFQ algorithm evenly divides every link transmission capacity in a network into N pieces. One piece is permanently allocated to datagrams, and another piece is permanently allocated to agents, while the other pieces

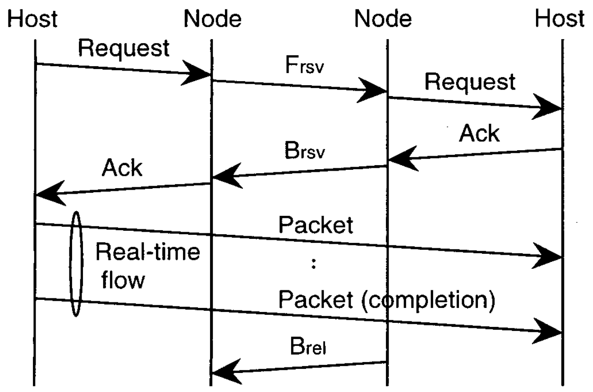


Figure 1: Delivery of a real-time flow between two hosts.

($N-2$ pieces) are allocated to each real-time flow on demand by the ARS. Hereafter, we call each of the $N-2$ pieces for real-time flows a "resource".

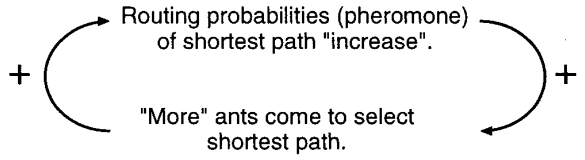
Fig. 1 illustrates a delivery of a real-time flow between two hosts. Every time a user requests a delivery of a real-time flow from a host, the user indicates the required bandwidth (m units of resources) between the two hosts. After the ARS decides to accept the request, an agent F_{rsv} reserves resources and sets up a path with a bandwidth greater than m units between the two hosts. The ARS prepares *a priori* the route information of a path that satisfies the user request. Note that an established path is uni-directional. An agent B_{rsv} is only used to notify the establishment of a successful path. Packets are routed by source routing, and the last packet piggybacks the notification of completion. Finally, an agent B_{rel} releases all resources allocated for this user request. All agents and packets in Fig. 1 use the same path. If an agent F_{rsv} arrives at a node that does not have enough resources to satisfy the user request, then it creates an agent B_{rsv} and dies. B_{rsv} releases all resources that F_{rsv} reserved.

3 QoS Routing of the ARS

The ARS uses an extended version of the AntNet to find feasible paths. A path is "feasible" when all necessary resources for setting up the path are now available. In this section, we first give an overview of the basic mechanism of the original AntNet algorithm and its characteristics, and then introduce the extended AntNet. For a more detailed description of the AntNet, see [15].

3.1 Basic AntNet Mechanism

To easily understand the AntNet, it may be helpful to learn the behavior of a real ant colony, since AntNet is an application of such behavior to datagram routing. A real ant colony is able to select the shortest path between its nest and a food source [16] because of the ants' trail-laying/trail-following behavior: individual ants emit a pheromone on the ground that attracts other ants. At first, each ant randomly selects a path between the nest and food. However, since a pheromone is quickly deposited on the shortest path, the number of ants that select the shortest path gradually increases. Finally, almost all ants select it due to the positive feedback (Fig. 2).



Routing probabilities (pheromone) of shortest path "increase".
+
"More" ants come to select shortest path.
+

Figure 2: Positive feedback mechanism forms a continuous circle, so shortest path is strongly marked with large probabilities (a great amount of pheromones).

Analogously, each node s (nest) in the AntNet network periodically generates an artificial ant (a forward ant F_{ant}) with a randomly selected destination node d (food) to observe the trip time from node s to destination node d. After F_{ant} arrives at destination node d, it creates another artificial ant (a backward ant B_{ant}) and dies. B_{ant} returns to node s to report the trip time of F_{ant}. B_{ant} takes the same path as F_{ant} in the opposite direction. Instead of a pheromone, every node s in the AntNet network maintains a set of routing probabilities $\{P^s_{d,n} | n \in N_s, d \in S - \{s\}\}$, where S and N_s denote the set of nodes in the network and the set of neighboring nodes of node s, respectively. $P^s_{d,n}$ represents a routing probability that node s sends packets and forward ants to the neighboring node n when their destination node is d; therefore,

$$\sum_{n \in N_s} P^s_{d,n} = 1, \; \forall d \in S - \{s\}.$$

Just as real ants deposit pheromones, every time a forward ant whose destination node is d goes over a link from node s to node n, the corresponding routing probability $P^s_{d,n}$ is increased when a backward ant created by the forward ant returns to node s. The amount of increase depends on the forward ant's trip time from node s to node d. A shorter trip time results in a larger probability increase. Since large routing probabilities attract forward ants (just as more pheromones attract real ants), the AntNet is able to find the shortest delay path based on the same mechanism as that of the real ant colony (Fig. 2).

3.2 AntNet Extensions for Constraint-based Routing

We now describe the extended AntNet that can find feasible paths that minimize delay under a set of QoS constraints. The ARS supports two QoS constraints:

bandwidth and hop-count. The amount of bandwidth is determined at the arrival time of each user request; in contrast, we suppose that the maximum hop-count (H_{max}) is determined *a priori* for all user requests.

The AntNet can be easily extended to support bandwidth and hop-count constrained routing capabilities. These capabilities can be obtained by simply restricting the actions of all forward ants. By making forward ants use links whose residual (unused) resources are more than a certain number of units, the extended AntNet becomes a bandwidth constrained routing algorithm. If the forward ants die when their hop-counts exceed H_{max}, then the extended AntNet becomes a hop-count constrained routing algorithm.

The AntNet with bandwidth-constrained routing capability presents a problem where under heavy user request most of the forward ants have to stay in a few lightly utilized links. In this situation, feasible paths will never be discovered and they just waste transmission bandwidth. We introduced two more rules for overcoming this problem. First, if a forward ant cannot select a next hop node since all outgoing links do not satisfy the bandwidth requirement, then the forward ant dies. Second, if a forward ant has visited the same node at least twice, then the forward ant dies. These two rules prevents many forward ants from staying on a network for a long time.

4 Global Admission Control

4.1 Objective

The objective of the ARS is to efficiently allocate network resources according to user requests. We use the term efficient to indicate that most of the reserved resources should be used to transmit real-time flows. It is important that at any time, every resource has one of three states: "used," "unused," or "reserved-but-unused," where the state "reserved-but-unused" denotes that a resource of this state is reserved but the path which will be established by using this resource has not been established yet. Under heavy user requests, the percentage of "reserved-but-unused" states becomes dominant, since attempts at setting up paths frequently fail. The ARS, however, can work well under this environment due to the efficient admission control mechanism.

4.2 Description of Admission Control

The basic idea for achieving efficient admission control is as follows. Since many forward and backward ants move around on a network to search for feasible paths, we make these ants carry information on resources of nodes that they have visited. This information is then used to decide whether feasible paths maintained in each node are currently feasible or not. Hereafter, we call a feasible path maintained in each node an "f-path".

We are now ready to describe the ARS's admission control. Every node in a network maintains one path cache. A path cache includes f-paths for all combinations of destination nodes and bandwidth requirements, and a set of states of all links that make up these f-paths. Let $P(s,d,m)$ denote an f-path from node s to node d with a bandwidth greater than m units, and let $s(n_1,n_2,m)$ denote a state that indicates whether link (n_1,n_2) (a link from node n_1 to node n_2) has at least m units of unused resources or not. Then, the path cache in node s includes

$$P(s,d,m),\ \{s(n_1,n_2,m)|(n_1,n_2) \in P(s,d,m)\},\quad \forall d \in S-\{s\},\ \forall m \in M,$$

where S and M are the set of nodes in the network and the set of bandwidth requirements supported by the ARS, respectively. The following describes the admission control mechanism.

1. When a backward agent (which corresponds to one of three kinds of agents: B_{ant}, B_{rsv}, and B_{rel}) leaves node k, it carries states of all node k's outgoing links. A state of a link carried by an agent is represented by a tuple (k,l,r), where l is a neighboring node of node k and r is the number of unused resources of link (k,l).

2. When a backward agent arrives at node f, by using each link state (k,l,r) carried by the agent, it updates link states in the path cache as follows. For all $d \in S-\{f\}$ and $m \in M$, if $(k,l) \in P(f,d,m)$, then the corresponding link state $s(k,l,m)$ is updated as

$$s(k,l,m) = \begin{cases} 1, & \text{if } r \geq m, \\ 0, & \text{otherwise.} \end{cases}$$

3. When a backward ant B_{ant}, which returns from node d to node s after discovering a feasible path with a bandwidth greater than m units, reaches its destination node s, it replaces f-path $P(s,d,m)$ in the path cache with this newly discovered feasible path; it then updates link states in the path cache as described in 2.

4. For each combination of a destination node and a bandwidth requirement, each node has one user request queue, so that one node has a total of $(|S|-1) \times |M|$ user request queues. Each arriving user request is assigned to one of these queues according to the destination node and the required bandwidth of the request. Periodically, the ARS tries to set up a path for one user request. The ARS selects a request whose corresponding f-path is "available" from all requests waiting at the head of these user request queues in a round robin manner. An f-path $P(s,d,m)$ is "available," only if $s(n_1,n_2,m)=1$ for all links $(n_1,n_2) \in P(s,d,m)$;

otherwise it is "unavailable". If the path establishment for a request succeeds, the request is removed from a queue; otherwise, it stays in a queue until the path establishment for the request succeeds.

A modification of the link states described in 2. sometimes makes a state of an f-path unavailable or makes a state of an f-path available again. We call the latter case "mosaic discovery" of a feasible path, and distinguish it from "exploratory discovery". In "exploratory discovery," a new feasible path is discovered as a result of a forward ant F_{ant}'s exploration, which is then reported by a backward ant B_{ant} as described in 3. Mosaic discovery may be effective when the number of consumable resources rapidly changes.

The merits of this admission control mechanism are summarized as follows. (1) Immediate path establishment can be achieved since f-paths are always prepared. (2) An "available" f-path is very reliable, since the contents of the path cache are constantly updated. It is worth noting that a path along which state information is gathered by a backward agent is not randomly selected. For example, a backward agent B_{rsv} (B_{rel}) carries the states of a path whose resources have just been reserved (released), while most of the backward ants (B_{ant}s) carry the states of shortest-delay paths. As a result, the states of these important paths are frequently updated, but the states of the other paths (useless information) are scarcely reported.

5 ARS Efficiency

5.1 Simulation Conditions

This section demonstrates the efficiency of the ARS by computer simulations. We first explain the simulation conditions. Fig. 3 shows the network topology. All links in the network are bi-directional and fully duplex, and have the same transmission bandwidth (1.5M bits/sec) and resources ($N-2=10$). The capacities of all output buffers are infinite, so packets and agents are never lost. The maximum hop-count (H_{max}) is set to 6. The ARS can support two bandwidth requirements: one unit and three units (i.e., $M=\{1,3\}$), so that one extended AntNet algorithm is executed for each bandwidth requirement. The processing time of each agent is 2 msec, and the generation interval of forward ants is 1 sec. The sizes for F_{ant}, B_{ant}, F_{rsv}, B_{rsv}, and B_{rel} are $(24+2h)$, $(24+2H+3R)$, $(24+H)$, $(24+H+3R)$, and $(24+H+3R)$ bytes, respectively, where h, H, and R represent the number of visited nodes, the number of nodes to be visited, and the number of link states, respectively.

User requests arrive at each node according to a Poisson process, and the arrival rates (λ) for the two bandwidth requirements are identical. The packet arrival rates for one unit and three units of bandwidth requirements are 37 packets/sec and 110 packets/sec, respectively. The total number of packets per flow is uniformly distributed in the range (1,100). The destination node for each user request is uniformly selected over the network. The datagrams arrive at 37 packets/sec. The sizes of packets and datagrams have a negative exponential distribution with a mean size of 512 kbytes. Real-time flows are shaped before injection into the network according to a leaky bucket filter, where the token bucket rate and the peak rate are equal to the bandwidth requirement of the flow, and the token bucket size and the maximum packet size are set to 2048 kbytes.

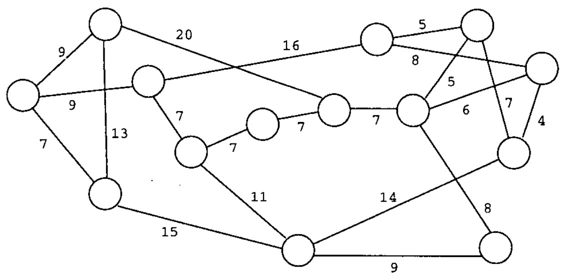


Figure 3: Network topology and link propagation delays in msec.

5.2 Simulation results

We measured the efficiency of the ARS based on the percentages of "used" states and "reserved-but-unused" states of the resources in the network. A large percentage of "used" states results in high throughput, while a small percentage of "reserved-but-unused" states decreases the path set-up times and the loads of agents in the network.

Fig. 4 shows the percentages of resource states as a function of the user request arrival rate λ. The figure demonstrates two important results. (1) The ARS achieved a high resource utilization rate. More than 70% of the resources were used to deliver real-time flows even when user request loads exceeded the network's capacity. (2) The ARS achieved a low resource contention rate. The percentage of "reserved-but-unused" states (i.e., uselessly reserved resources due to user contention) was less than 10%. No matter how efficient the ARS may be, it is impossible to completely utilize whole resources. This is because some unused resources always exist depending on the combinations of user requests. Fig. 4 also includes resources permanently allocated to the agents, since the efficiency of the ARS depends on the bandwidth allocated to them.

Here, we show in detail how the ARS controlled user requests. The user requests were divided into two groups: "unfinished" and "finished" requests. According to the results of the first path set-up trial, we further divided the group of the finished requests into three groups: "blocked," "failed," and "successful" re-

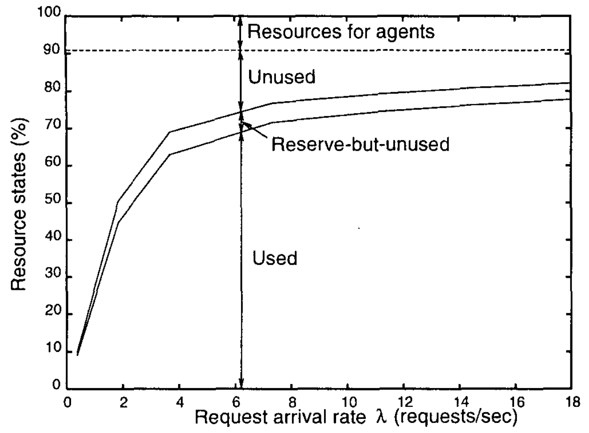


Figure 4: Percentages of resource states as a function of user request arrival rate λ

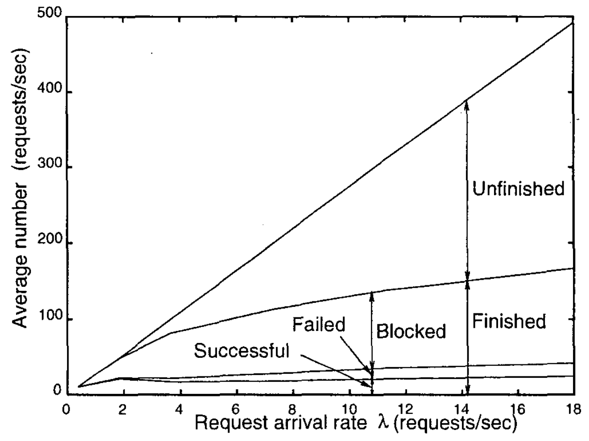


Figure 5: Average number of four user request groups as a function of user request arrival rate λ

quests, where the failed request indicates that the corresponding f-path for the request was available at that time, but the path establishment for the request failed. Fig. 5 shows the average number of these four user request groups as a function of λ. The important point is that there were very few "failed" requests compared to the sum of the other two finished requests (i.e., blocked and successful requests). This result indicates that the states of all f-paths in the path caches are almost correct.

6 Conclusions

In this paper, we described the ARS, which supports QoS routing, resource reservation, and admission control mechanism based on ant-like agents. Simulation results showed that the ARS achieves high resource utilization and low resource contention among user requests. This efficient control of user requests derives from the admission control mechanism; this mechanism maintains the current states of all feasible paths in each path cache by using agents that carry useful information for updating the cache.

Acknowledgements

We are grateful to Dr. Bokuji Komiyama, Dr. Shigeru Saito, and Mr. Jun Shiozaki for their assistance.

REFERENCES

[1] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control - The Single Node Case," in *Proc. IEEE INFOCOM'92*, Vol. 2, May 1992, pp. 915-924.

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Internetworking: Research and Experience*, Vol. 1, No. 1, pp. 3-26, 1990.

[3] R. L. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Transactions on Information Theory*, Vol. 37, pp. 114-131, 1991.

[4] R. L. Cruz, "A Calculus for Network Delay, Part II: Network Analysis in Isolation," *IEEE Transactions on Information Theory*, Vol. 37, pp. 132-141, 1991.

[5] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Network," *Proc. of the IEEE*, Vol. 83, No. 10, pp. 1374-1396, October 1995.

[6] D. Stiliadis and A. Varma, "A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithm," *Proc. of IEEE INFOCOM'97*, Kobe, Japan, April 1997.

[7] S. Chen and K. Nahrstedh, "An Overview of Quality-of-service Routing for the Next Generation High-speed Networks," *IEEE Network*, pp. 64-79, Nov./Dec. 1998.

[8] D. Bertsekas and R. G. Gallager, "Data Networks," *Printice-Hall International*, 1992.

[9] ATM Forum PNNI Subworking Group, "Private Network-Network Interface Spec. v1.0 (PNNI 1.0)," afpnni-0055.00, March 1996.

[10] Z. Zhang, C. Sanchez, B. Salkewicz, and E. S. Crawley, "Quality of Service Extensions to OSPF or Quality of Service Path First Routing (QOSPF)," *Internet Draft (draft-zhang-qos-ospf-01.txt)*, Sep. 1997.

[11] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, T. Przygienda, "QoS Routing Mechanisms and OSPF Extensions," *Internet Draft (draft-guerin-qos-routing-ospf-05.txt)*, April 1998.

[12] S. Appleby and S. Steward, "Mobile Software Agents for Control in Telecommunications Networks," *British Telecom Technology Journal 12*, pp. 104-113, 1994.

[13] R. Schoonderwoerd, O. Holland, J. Bruten and L. Rothkrantz, "Ant-based load balancing in telecommunications networks," *Adaptive Behavior* Vol. 5, pp. 169-207, 1997.

[14] E. Bonabeau, F. Henaux, S. Guerin, D. Snyers, P. Kuntz, and G. Theraulaz, "Routing in Telecommunications Networks with "Smart" Ant-Like Agents," *Agent World*, 1998.

[15] G. D. Caro and M. Dorigo, "Mobile Agents for Adaptive Routing," *Proc. of the 31st International Conference on Systems Siences* , The Big Island of Hawaii, Jan. 1998.

[16] E. Bonabeau, G. Theraulaz, J. L. Deneubourg, S. Aron and S. Camazine, "Self-organization in Social Insects," *TREE* Vol. 12, No. 5, 1997.