Implementation of a Software IP Router using Ant Net Algorithm

A. Jusrut, A. Khedan, and V. Ramnarain-Seetohul
Computer Science and Engineering Department, University Of Mauritius, Reduit, Mauritius
Computer Science and Engineering Department, University Of Mauritius, Reduit, Mauritius

Abstract - Nowadays, routing standards are governed by the Internet, and the state-of-the-art is link-state routing. These types of algorithms adapt well to network topology changes, but respond poorly in high-traffic environment, where the network load changes dynamically. Very often the shortcomings of these algorithms are not apparent, because their poor performance is compensated for by powerful routers and high-bandwidth links. In this paper, Ant Net, a novel routing approach based on research done mostly in reinforcement learning and swarm intelligence was explored. A Software router using Ant Net algorithm to route IP packets between Ethernet networks was implemented. CNET, a network simulator was used to test the routing algorithm. The simulations done were used create network environment so as to test all aspects of the routing algorithm. Besides these simulations, the software router was tested on a test-router, constructed using a Pentium-4 PC, equipped with 3 PCI Ethernet Network Interface Cards, and running a Linux distribution. Using this test-router, the forwarding mechanisms of the algorithm were tested on real Ethernet hardware.

Keywords: Ant Net, Algorithms, CNET, Ethernet, topology, router

1 Introduction

Today, computer networks are the core of modern communication. For a computer to send data to another computer found in its own LAN, communication is straightforward (e.g., broadcasting on Ethernets). But when it comes to inter-connecting different networks together, a whole new layer of protocols needs to be introduced. A router is a device dedicated to the task of routing data from one network to another and every router maintains a routing table. A hardware router has several ports to connect to different network and also has processing power and memory. Routers are generally expensive devices and their maintenance and reparation costs in case of failure are also very high. Moreover, modern routing algorithms implemented by hardware routers have some shortcomings that make them unsuitable in certain situations. Hence a software router with same capabilities as a hardware router can be a way to reduce cost and efficiently route packets. The paper consists of designing and implementing a software router that will help network administrators in experimenting with routing, without actually requiring a specialized hardware router. The router will run as a software application on a computer. It will provide the routing functionality of a real (physical) router efficiently, so that it can be used as a viable alternative by network designers. The paper focuses mainly on routers for connectionless networks; more specifically, the focus is on IPv4 routers. Since the objective is to find a useful, efficient and inexpensive solution to routing, only the routing and forwarding functionalities of a router are considered, while specialized hardware routers may provide a variety of other functions, aside from routing, such as NAT, firewall, DHCP among others. Static routing is not implemented in the software router, as the aim is to design a software router to work in dynamic environment.

2 Swarm Intelligence in networking

Swarm intelligence is the collective constructive behaviour that is observed when many individuals work independently, and coordinates by interacting amongst each other or with their direct environment [1].

2.1 Ant Net Algorithm

Ant Net [2,3] is a new type of routing algorithm, inspired from work done in the fields of ant colony optimization, where the natural characteristics of biological swarms (e.g., ants) are studied to solve optimization problems. Ant Net has been implemented and tested on simulation networks and real networks. The results and conclusions from Di Caro and Dorigo’s[4] original paper show that the algorithm is superior to all its competitors that were also subjected to the same tests. The competing algorithms included OSPF. Ant Net makes use of mobile agents, known as agents or ants. The agents are data units, generated, received and processed by the routers. The purpose of the agents is to explore the network and to exchange collected information. The marked difference here between Ant Net and other protocols, like OSPF, is that the agents communicate indirectly [4]. In OSPF, each router explicitly sends protocol packets to other routers, but in Ant Net, the agents just travel around the network, modifying routing tables...
of routers accordingly. This type of communication that happens through the environment and not directly between agents is called stigmergy; a concept inspired from social insects [5]. Ant Net uses two types of mobile agents: forward ants and backward ants [4]. Forward ants are launched at intervals by each router. The forward ant moves step-by-step towards the destination, and at each intermediate node it collects information as to how much time it took for the trip. On reaching the destination, the forward ant becomes a backward ant, with all the collected information [6]. The backward ant visits the same nodes as the forward ant, but in reverse, and updates routing data structures at each node. Each router keeps two data structures [4]: Fig. 1 shows the data structures required by Ant Net.

The Ant Net algorithm is robust, scalable with respect to network growth, highly traffic adaptive and provides inherent load balancing [7]. However, if topology is large, routing protocol traffic increases with ant stack size [8].

2.2 Modified Ant Net algorithm

A simplified version of the Ant Net algorithm will be implemented. The changes made to the algorithm, and the rationale behind will be discussed. The main change that will be done is that a constant reinforcement factor will be used to update routing table probabilities. Normally, in the original algorithm, the factor is calculated from the trip times of ants [4]. There is a practical difficulty in calculating packet trip times, since the clocks of all routers must be synchronized. Using a constant factor eliminates the need to use trip times and thus there is also no need for the statistical models: only the routing table is required. The probability values are then affected only by the arrival rates of ants [2]. Newly generated forward ants are forwarded on all router-router links. The changes mentioned will desirably have the following effects: Much lower protocol traffic, since ant stacks no longer contain trip times and considerable reduction of router processing loads. Calculation of the reinforcement factor repeatedly, and working with trip times, in the original algorithm, required lots of floating-point operations.

3 Proposed System

The software router will be used in packet-switched networks and will run on a computer that will need some additional hardware configuration. The machine to be used for routing will need to have more than one network interface card, else it will have the capacity to act only as a standalone host. The software will support only Ethernet NICs. The number of NICs installed will effectively determine the number of links that the router will have, and directly affect the number of networks that can be connected to it. Hosts on connected networks will not require any special configurations; they are configured normally for IP communication. A router may connect to another router, to a single host, or to a LAN. LANs may connect to the router via a switch, as in switched Ethernet. Fig. 2 illustrates the possible connections.

The sets of all the routers will form the backbone. Some routers in the backbone may not connect to any network, but only provide routing. These may be used on sites where data traffic between networks is very high, and will effectively tend to improve performance as they provide additional routes for packets to travel. The software router will implement the modified Ant Net algorithm as its adaptive routing algorithm. A router distinguishes between two types of packets: data packets and routing protocol packets. Any other packet not forming part of the routing protocol will be treated as plain data, and forwarded using information from the routing table. The routing packets will be packets generated, decoded and processed by the routers to construct the routing table. The router will work directly at layer 3 of the TCP/IP stack; routing protocol packets will have the IPv4 header as the last encapsulation. The router will not use any transport mechanism to ensure routing packets get delivered. This is also applied to data packets. As Ant Net protocol is designed for best-effort packet-switched networks, the routers will not be responsible for any data loss. For reliability, hosts will need to implement TCP. If the routers have to be configured for dynamic routing, then some configurations will be needed initially. Each router should know all reachable networks and its neighbors. This information will be available in configuration files.
4 System Architecture

4.1 Design

The software router will be implemented and tested on a network simulator first, and then if the algorithm works, the code will be tested on a Linux machine. Due to peculiarities with how network simulators work, there will be architectural differences in the implementations of the two versions. The core router architecture should be independent of platform specific details. The core architecture comprises of the routing and forwarding mechanisms, and platform refers to the environment in which the router will be implemented and tested; that is, on the network simulator or on a real physical setup. This prevents tying the core design with the intricacies of the simulation tool. It will also be easier to change the software later to use the libpcap API [9]. The performance of router should also be considered. A router is a complex real-time processing system, with hard deadlines. If it is poorly design, this will result in an excessive loss of packets.

4.2 Architectural Design

Most networking software is written with a layered design. The interfaces of the router need to be managed at a very low level; network traffic should be handled as soon as they enter an interface. This will allow link statistics to be maintained, and most importantly, allow the flexibility to queue up packets based on their incoming/outgoing interface. Thus, a proper architecture operating at the link layer needs to be designed. The way the Ant Net algorithm is defined requires a lot of book-keeping at the link level [5]. Hence it is important to decide when to process inbound and outbound packets. One possibility with inbound packets is to start processing them immediately as soon as they have been stored by the interface hardware (i.e., Ethernet card). Similarly, outbound packets are sent to the hardware for transmission as soon as they have been processed. However, this scheme has inherent problems:

- Inefficient processing/scheduling. The operating system will have a limited buffer for incoming packets. Processing packets one by one as soon as they come will mean that the OS buffer will tend to fill up, resulting in loss of packets.
- It is difficult to build up data-structures (queues) that are required for statistical data collection.

The solution that was chosen is to defer the processing of incoming packets. Packets are stored for later processing or transmission. As soon as a packet comes, it is buffered in a queue. Each interface on the router has a corresponding queue to store incoming packets. Packets are then dequeued from the queues and processed. Similarly, when packets have been processed, they are queued in the outgoing queue for the appropriate link. Later on, the queues will be emptied and the packets will be transmitted on the network card. Fig. 3 depicts this architecture.

At the network layer, the modified Ant Net algorithm is used. This layer determines how to process packets received from the link layer, and after processing, packets are sent again to the link layer, to be queued. A suitable format for an ant packet will be designed. This layer is best described by means of the data-structures and algorithms. The link layer and network layer have to exchange messages. One possibility is to copy the protocol data units from layer-to-layer. But, there will be too much processor time wasted in copying data around. Instead, a different design is used, but it blurs some extent the separating line between both layers. When a frame comes in, the link layer allocates memory needed to store it in the queue. When the network layer requests a packet, instead of copying the packet in the frame, the memory reference of the frame itself is passed. All processing will happen on this memory, and it is this memory itself that will be queued on an outgoing link queue. Hence, only memory addresses, and not whole memory buffers will be passed back and forth. Fig. 4 illustrates this concept.

A radically different approach will be used for the link queues: the length of a queue will have no limit. Dedicated hardware routers have limited memory; hence they use fixed-length queues and perform load-sheding if there is no more room. The software router will run on a Linux PC, and Linux provides a virtual address space of 4 GB, irrespective of physical RAM installed. When a new frame comes in, the Operating System will be requested for memory to store the frame. If the request is unsuccessful, only then the frame is discarded. Queues will be serviced in a round-robin manner. The design of the router is inherently multi-threaded. The multi-threading API of Linux will be used. All the threads will have simultaneous access to shared data segments, like the
routing table. Mutexes will be used to provide mutual exclusion between threads, for variables that can change values during program execution. Locks will be applied whenever shared memory needs to be read or written.

The frame queues will be constructed using linked lists. Frame queues are lists of ether_frame structures, logically maintained as FIFO data-structures (queues). Each link on the router will have its own incoming and outgoing queue. Moreover, there is a priority queue for holding backward ant packets, since these have to be processed very fast. All the queues are implemented in the same way; a queue is accessed using a global variable, which is an array of pointers to the queue heads. The queue heads are ether_frames.

4.3 Simulation Plan

The simulations will be carried out on topologies that consist only of routers and LANs. A LAN will have a network id and a subnet mask. A LAN will be represented by a single host workstation with an IP address. The LANs in the simulation are hereafter referred to as hosts. Different simulation topologies consisting of different numbers of routers and hosts will be created, and the routing algorithm will be tested on them. Fig.5 illustrates a simple topology.

Each node in the simulation, router or host, will have a log file to which it will dump information about its operations. These log files will provide the basis for the simulation analysis. Modification made to the routing tables can be verified by analysing these files.

5 Implementation and testing

The software router was first simulated using the CNET simulator. Fig. 6 shows a simulation window, with the network shown graphically. The simulation consists of five hosts each on a different network and six routers to route data on the network.

Fig 6: CNET Simulation shown graphically

5.1 Testing

Using simulator CNet, performance testing was done. The variables that were tested were packet delay, packet size packet loss, data rate.

Fig 7: graph for delays for packets in 3 specific flows

Fig 8:graph for packet delay variation with packet size.

Fig 9: Graph for packet delay variation with time, given one particular flow is made to generate data at a fast rate.
According to results obtained it can be deduced that on average the packet delay does not increase considerably as number of packets increases. As packet size increases, there is a slight increase in the packet delay. If a given flow is made to generate data at a faster rate, it can be observed that the packet delay increases considerably at first but then stabilises as number of packets increases. If a particular routing node has failed then according to its data rate, variation of the packet loss was measured and results concluded that as number of packets lost increases, the time between generations of new packets decreases. Finally the variation pheromone values of a router were tested. This test helps to analyze the changes in routing table during load-balancing. Result shows that the best path is through link 4 in Fig. 11, and is used initially. But soon, due to a fast data rate towards a host, this best path suffers some delays. Hence, an alternate path, through link 3 is then chosen. However after some time, the performance of the alternate path starts degrading due to congestion and gradually, the router starts switching packets on link 4 again.

To fully test its operations, the software router has to be tested in real life situation but the main limitation was the unavailability of a real network to perform the testing. Hence, to counterbalance this limitation, the software router has been tested on a PC with three LAN cards to verify if it is operating correctly. It was observed that the software router was indeed forwarding data to the hosts. However to create a topology for testing, several PCs with more NICs were required.

6 Conclusion

The software router understands only IP at the internetwork layer of TCP/IP. There is no support for additional protocols like ICMP, IGMP, IP multicasting, IPsec. Support for these protocols can be added by extending the router code base to distinguish special IP packets from normal data packets. Furthermore, the software router also understands only IPv4. To support IPv6, a redesign of the routing table and other major data structures will be needed. At the link layer, the router simply broadcasts packets on destination LANs, and the network layer on hosts’ filters them based on IP address. This can be avoided by implementing ARP and maintaining an ARP cache on the router. Also, the router can be made to detect neighbors and networks by implementing hello and flooding mechanisms similar to OSPF. The flooding will only aid in discovering new networks being added, and the hello packets will query neighbors. However, these mechanisms will be done less frequently than in OSPF, since here they do not form integral part of the routing algorithm.

7 References