SMART PERMANENT VIRTUAL CIRCUITS IN ASYNCHRONOUS TRANSFER MODE NETWORKS

Tibor Gyires
Applied Computer Science Department
Illinois State University, Normal, Illinois 61790-5150
tbgyires@ilstu.edu

ABSTRACT

Asynchronous Transfer Mode (ATM) is a high-speed network technology that transmits various types of information across networks such as voice, video, image, data, etc. In an ATM network the basic data units, called cells, are routed through switched or permanent virtual circuits (virtual channels). A Smart Permanent Virtual Circuit is a connection that looks like a Permanent Virtual Circuit at the local and remote endpoints with a Switched Virtual Circuit in the middle. If a link carrying a Smart Permanent Virtual Circuit goes down and there is an alternate route, then the network automatically reroutes the Smart Permanent Virtual Circuit around the link. As a result of the rerouting, the network may not be able to deliver the guaranteed quality of services as it was negotiated. It may have to change the quality of service parameters negotiated for other connections. The objective of this paper is to apply Distributed Artificial Intelligence (DAI) methodologies, "intelligent" agents", in ATM networks management. The paper presents a search algorithm that helps the agents learn from previous interactions and experience. Agents can evaluate alternate paths in order to maintain as many connections as possible with the quality of service guaranteed originally.

1. INTRODUCTION

Asynchronous Transfer Mode (ATM) is a new high-speed network technology that transmits various types of information across networks such as voice, video, image, data, etc. In an ATM network the basic data units, called cells, are routed from a source to a destination through switched or permanent virtual circuits (or virtual channels) that can be bundled into virtual paths. A Switched Virtual Circuit is used to transport information between two locations and lasts only for the duration of the transfer. Permanent Virtual Circuits are used for dedicated long-term information transport between locations. Establishment of virtual circuits and paths are based on a negotiation between the calling party and the ATM network. The parameters being negotiated include average and peak bandwidth, burst length, and quality of service parameters for transmitting various types of information. As a result of the connection establishment procedure, a "contract" is secured between the ATM network and the ATM end station. The ATM network promises to deliver the guaranteed quality of service and the ATM end station promises not to send more traffic than it requested in the connection establishment procedure.

The cell routing is controlled by the ATM switches along the path between two ATM endpoints. A Smart Permanent Virtual Circuit as it has been defined in [1] is a connection that looks like a Permanent Virtual Circuit at the local and remote endpoints with a Switched Virtual Circuit in the middle. Smart Permanent Virtual Circuits are more robust than Permanent Virtual Circuits. If a link carrying a Permanent Virtual Circuit goes down, then the Permanent Virtual Circuit goes down. If a link carrying a Smart Permanent Virtual Circuit goes down and there is an alternate route, then the end switch of the Smart Permanent Virtual Circuit automatically reroutes the Smart Permanent Virtual Circuit around the link. As a result of the rerouting, the network may not be able to deliver the guaranteed quality of service as it was negotiated in the contract. It may have to change the quality of service parameters negotiated for other connections. There is no widely accepted standardized procedure for creating and maintaining Smart Permanent Virtual Circuits. It is still a research topic that is similar to the issues discussed under one of the ATM network management functions called Fast Resource Management.

Routing and rerouting circuits in high-speed telecommunication networks are important areas of research. A current research direction solves this rerouting problem by using Distributed Artificial Intelligence approaches to select between alternative rerouting plans [2]. The objective of this paper is to apply Distributed Artificial Intelligence (DAI) methodologies, "intelligent" agents", in ATM networks management. "Intelligent agents" cooperate in order to ensure network-wide control. Each agent has local decision-making capabilities and some knowledge about system parameters of part of the network. None of the agents can control the entire network alone. We present

This research has been supported by the Center of Information Systems Technology (InfoTech) at Illinois State University through their Research Incentive Program, Normal, Illinois 61790 - 5151.

0-7803-4053-1/97/$10.00

a search algorithm that help the agents learn from previous interactions and experience. Agents can evaluate alternate paths in order to maintain as many connections as possible with the quality of service guaranteed originally. This paper presents an improved and modified version of the rerouting algorithm in [3]. The main difference is that while in [3] the rerouting algorithm is started from both ends of the disrupted circuit following a bidirectional approach, in this paper the rerouting algorithm is initiated by the a single switch that originally established the circuit. The presented method reduces the exponential complexity of the rerouting algorithm to linear-space complexity. The routing algorithm of this paper applies a modification of the Simple Recursive Best-Search algorithm [4]. The advantage of our method is discussed later.

The rest of this paper is organized as follows. Section 2 is a discussion of the motivation for investigating Fast Resource Management issues. Section 3 contains an overview of our search algorithm. The complexity of the rerouting algorithm is discussed in section 4, followed by conclusions in Section 5.

2. MOTIVATION FOR INVESTIGATING FAST RESOURCE MANAGEMENT

Fast resource management enables the immediate allocation of necessary capacity in an ATM network, such as bit rate or buffer space to individual burst-type connection for a duration of a cell burst. For instance, the user would like to temporarily exceed the contracted circuit capacity to send a large amount of data. If the ATM network determines that the necessary resources exist along the route of the virtual circuit for the data burst, then it reserves the resources and establishes the connection. The situation is similar to finding an alternate path in Smart Permanent Virtual Circuits when a circuit is disrupted.

For years, queuing models of network traffic assumed a predictable Poisson distribution. A relatively new result published in [5] proves that network traffic is more bursty and cannot be predicted its variability as assumed previously. According to the authors network traffic has similar statistical properties in a certain amount of time called self-similarity. The importance of the results is also summarized by W. Stallings in his comment in [6]. Stallings pointed out the consequences of this finding. One of the consequences is that combining of various flow of data, as it happens in ATM virtual paths, does not result in the smoothing of traffic. Combining bursty data streams will produce also bursty combined data flow. The result implies that the methods and models used in traditional network design require modifications. Self-similar traffic makes traditional assumptions incorrect. For instance, it has been assumed that linear increase in buffer sizes results in exponential decrease in packet loss. In self-similar traffic such a decrease of packet loss is far less than it has been predicted by traditional models. It has been proved that in case of self-similar traffic the addition of few new circuits can cause a large loss of data packets. Also, in a system where a high priority circuit can use the channel for a burst of traffic with no traffic policing, other lower priority circuits are kept from using the network for a long time. As a consequence of the property of self-similar traffic the various network design parameters can be better estimated based on the traffic patterns rather than by applying traditional queuing models. Self-similarity occurs in ATM networks. The bursty nature of ATM traffic and related issues have been identified by the ITU-T and the ATM Forum under one of the traffic management functions entitled "Fast Resource Management".

3. OVERVIEW OF THE SEARCH ALGORITHM

We assume a wide area network of connected ATM switches. The circuit rerouting process is started by the switch that established the circuit and observed its interruption. The goal of the switch is to identify the switches and links to restore the disrupted circuit. We use the network topology of Figure 1 to illustrate the restoral algorithm. The sites X, A, B, C, Y, E, F, G, H, I, J, K denote the switches. S and D are the two end-users of a connection with the circuit c:{X, K, G, F, Y}. We assume a link failure between sites K and G. The rerouting process is initiated by sites X that established the circuit before the failure.

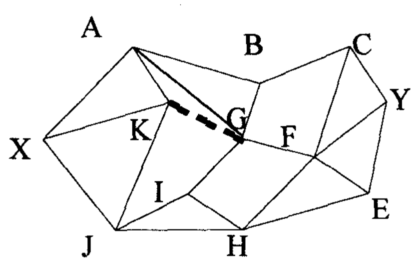
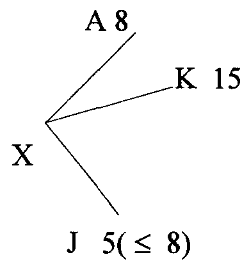


Figure 1: Circuit failure between X and Y

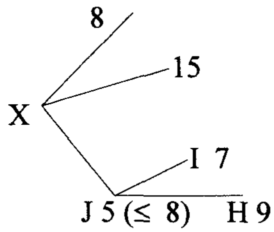
X sends a request to restore circuit c to adjacent switches A, K, and J along with an upper bound on the rerouting cost and the circuit characteristics, such as service categories (constant bit rate, variable bit rate, available bit rate, unspecified bit rate), traffic parameters (peak cell rate, sustained cell rate, cell delay variation tolerance, burst tolerance, minimum cell rate), quality of service parameters (cell loss ration, cell delay variation, maximum cell transfer delay, mean cell transfer delay). A switch is qualified if it has been able to restore similar circuits in previous cases. The selection mechanism is based on a performance parameter P which is a measure of a switch's inefficiency (as calculated by its predecessor switch along the rerouting circuit) based on its previous reroutings. P may be different for different circuits and is dependent on the specific circuit to be rerouted, because the resource and capacity requirements of a circuit determine the switch's capability to restore it. Small (close to zero) values of P indicate high quality and efficiency, and high values indicate that the switch is unreliable and unstable in rerouting the circuit. A switch's P value is recalculated after each circuit rerouting by that adjacent switch which sent a rerouting request to it. After each rerouting it is determined how realistic a switch's cost estimate has been with respect to the recalculated cost. By testing a statistical hypothesis it can be determined if it has been unrealistic. If the corresponding hypothesis H_0 is rejected no rerouting request messages will be sent to this switch in the future for rerouting circuits. If a switch's cost estimate has been realistic, i.e., the corresponding hypothesis H_0 is accepted, the P value is recalculated. The closer the estimated cost is to the actual cost, the smaller is the P value. Subsequent values of P for each switch are calculated using the statistical sampling method given in [3]. The motivation for using P is to enable the search process to learn from past performances and use this knowledge to select the most efficient rerouting switches.

The switches compute the estimated cost of each of the activities associated with the rerouting process, such as the allocation or reallocation of network resources, rerouting existing but lower priority connections, and reestablishing interrupted connections. These cost estimates are sent back to X, which will select the route with the smallest cost estimate, e.g., via J. The process continues recursively. At each recursive step a site uses three arguments: a subsequent site, an upper bound on the restoral cost, and the circuit number. Each step expands the path by those children through which the estimated restoral costs do not exceed the upper bound. (The first step assumes an upper bound of infinity at X.) Each step returns the estimated cost along the path to a child, replacing parent values with the minimum of the estimated costs via the last children expanded, going back along the path, until a better cost estimate is reached. Then, the procedure continues along that path. Generally, the upper bound on a child is equal to the minimum of the upper bound on its parent and the current value of its best sibling. Initially, a switch is assigned the estimated cost via itself. After a recursive step this cost value will be equal to the minimum estimated cost path to the last child along the expanded subtree. We call it the switch's stored value after the original Simple Recursive Best-First Search (SRBFS) algorithm in [4]. The following chart shows the first steps of the algorithm. After receiving the cost estimates from J, K, and A, X decides to continue the search via J initializing a recursive procedure on J with the upper bound 8. In the chart below the figures denote the cost estimates through A, K, and J, and the upper bound (in parentheses):



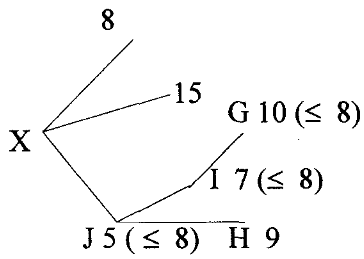
a.

J continues the search by sending a request to adjacent sites K, I, and H to restore circuit c. In the chart below, only I and H will respond with their cost estimates, hence J can estimate the cost of the paths via these sites:

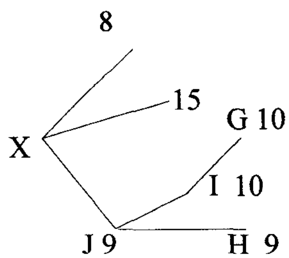


b.

Site J decides that the search continues via I with an upper bound 8, that in turn, will send a request to G and H. Because H has already received a request from J, only G responds with an estimated cost exceeding the upper bound 8. The process returns back up the tree replacing the parent's value. The following two charts show these steps:

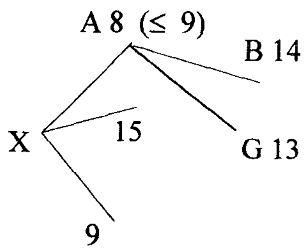


c.

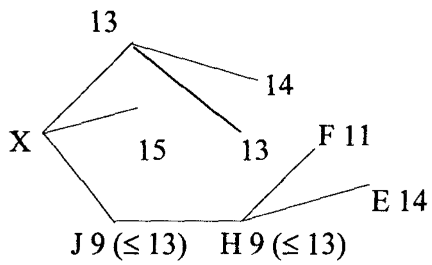


d.

The search switches to J's next best sibling A which sends requests to B, G, and K. The following charts show the progress of the search and depict only the newly explored paths for simplicity:

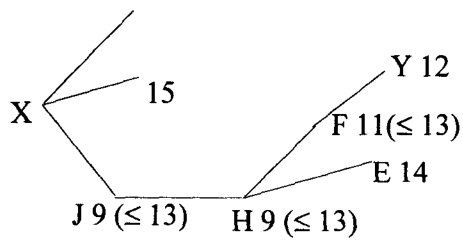


e.

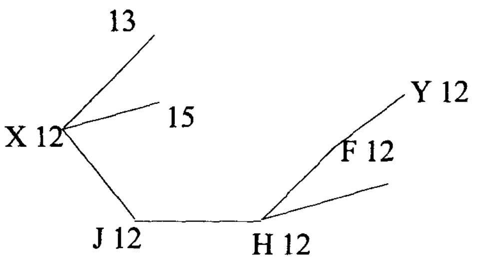


f.

The search continues from H that will send a request to F and E. The charts below display the final steps of the algorithm:



g.



h.

Once the destination Y is reached the process stops. The rerouted circuit is along the return path of the recursive steps. Selection of the rerouted circuit by the participating switches is based on cost estimates and information about the switches' previous performance. Since at each step the estimated least-cost path is selected, the resulting restoral path is the expected least-cost path or close to it. It can be proved that the above algorithm always finds the least-cost rerouting circuit if some conditions are met. Due to space limitation the proof of this statement and the formal description of the algorithm are omitted but available from the author upon request.

4. The Complexity of the Algorithm

First we give an overview of the similar search algorithms. The Iterative-Deepening-A* (IDA*) in [7], which is a modification of A* [13], performs a sequence of depth-first searches, pruning branches when their cost exceeds a threshold for the current iteration. The initial threshold is determined by the cost estimate at the root and increases for each iteration of the algorithm. Each subsequent threshold for each iteration is the minimum cost of all values that exceeded the previous threshold. IDA* expands the same number of nodes asymptotically, as A*. It is shown in [8] that IDA* is asymptotically optimal in terms of time for tree searches. The important property of A*, that it always finds the lowest-cost solution path if the heuristic is admissible, also holds for IDA*. Although it is easier to implement than A* (as there are no open and close lists to be managed), it uses a global threshold, that is difficult to maintain in a distributed system.

Our algorithm is a modification of the Simple Recursive Best-First Search (SRBFS) which is an extensions of the IDA*. Therefore, the complexity of our algorithm can be derived from the complexity of this algorithm. While iterative-deepening uses a global threshold, SRBFS uses a local cost threshold for each iteration with two parameters: the next child along the path, and an upper bound on cost. It explores the branch below the child as long as it contains expanded children, whose costs do not exceed the upper bound. Each iteration returns the minimum cost of the newly expanded children.

Although the space complexity of our algorithm is O(db) (similarly to SRBFS), where b is the branching factor and d is the maximum search depth, the worst-case time complexity is $O(b^{2d})$ depending on the cost function. With a monotonic cost function, it finds an optimal solution while expanding fewer children than iterative-deepening. The method in SRBFS and in our algorithm reduces the space complexity of best-first search from exponential to linear (assuming a constant branching factor). The reason is that the recursive procedure only maintains the path to the best frontier children of the explored subtree, and all siblings along the path. While in IDA* (and A*) each new iteration regenerates the entire previous tree, our algorithm only explores the branches of siblings on one of the last paths of the most recent iteration. The algorithm increases the time complexity by only a constant factor [4]. It can also been shown [4] that the property of A* also holds for SRBFS, therefore the same property holds for our method as well, i.e., if the cost estimates at each switch are close to the lower bounds of the actual cost, then the restoral algorithm will find the least-cost rerouting circuit.

5. CONCLUSIONS

In this paper we have presented a circuit rerouting algorithm that can be used in establishing Smart Permanent Virtual Circuits in ATM networks. Our contribution is a search method which can learn over the time. The search is sensitive to the switches' previous performance. It will choose the "best" switches to reroute a circuit. "Best" switches are the ones that are involved in locating the least cost rerouting path. The method tends to find the least-cost path in the effort to reroute a failed circuit. Traditional search algorithms start over again without remembering previous "successful, least cost" rerouted path, therefore they visit switches again, even though it has been statistically proven that they cannot restore the circuit in an efficient way.

6. REFERENCES

[1] ForeRunner ATM Switch Configuration Manual, FORE Systems, Inc., MANU0066, March, 1996

[2] Conry, S., Kuwabara, K., Lesser, V., "Multistage Negotiation for Distributed Constraint Satisfaction," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 6, November/December 1991, pp.1462-1477.

[3] Gyires, T., Muthuswamy, B., "A Bidirectional Search Approach for Restoring Circuits in Communication Networks", the Winter 1996 issue of The Journal of Computer Information Systems.

[4] Korf, R., E., "Linear-Space Best-First Search: Summary of Results," Proceedings of the Tenth National Conference on Artificial Intelligence, San Jose, California, July 12-16, 1992, pp. 533-538.

[5] Leland, W., Taqqu, M., "On the Self-Similar Nature of the Ethernet Traffic," IEEE Transactions on Networking, Vol 2, no. 1, February 1994, pp.1-15.

[6] Stallings, William: "Viewpoint: Self-similarity upsets data traffic assumptions," IEEE Spectrum, January 1997, p. 28.

[7] Korf, R., E., "Iterative-Deepening-A*: An Optimal Admissible Tree Search," Proceedings of the Ninth International Joint Conference on Artificial Intelligence, UCLA, August 18-23, 1985, pp. 1034-1036.

[8] Dechter, R. and Pearl, J., "The optimality of A* revisited," Proceedings of the National Conference on Artificial Intelligence, Washington, DC, August, 1983, pp. 95-99.

[9] Durfee, E., "Coordination of Distributed Problem Solvers," Kluwer Academic Publishers, 1988.

[10] Korf, R., E., "Depth-First Iterative-Deepening: An Optimal Admissible Tree Search," Artificial Intelligence, Vol. 27, 19985, pp. 97-109.

[11] Bond, A., and Gasser, L., "Readings in Distributed Artificial Intelligence," Morgan Kaufmann Pub. Inc., 1988.

[12] Gyires, T., "A Heuristic Algorithm for Distributed Control in Manufacturing Systems," Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem Solving Technologies, Vol. 1, No. 2, 1991, pp.145–155.

[13] Hart, P.E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions on Systems Science and Cybernetics, 4(2), 1968, pp. 100-107.

[14] Sycara, K., Roth, S., Sadeh, N., and Fox, M., "Distributed Constrained Heuristic Search," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 6, November/December 1991, pp. 1446-1461.