

NETWORK CONGESTION ARBITRATION AND SOURCE PROBLEM PREDICTION USING NEURAL NETWORKS

J. ALAN BIVENS
Computer Science

BOLESŁAW M. SZYMANSKI
Computer Science

MARK J. EMBRECHTS
Decision Sciences and Engineering Systems
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

ABSTRACT

Due to the quickly increasing need for networking, both on the Internet and intranet levels, network management is becoming more important in today's world. We propose using learning techniques to predict different network problems before they start. In this paper we focus on using a simple feedforward neural network to predict problems such as severe congestion in a network. We employ the use of neural networks to predict the source or sources responsible for the congestion, and then use simple control methods to punish the nodes convicted of the crime. This scenario sets the stage for a new wave of network managers, those that prevent networking problems instead of repairing them.

INTRODUCTION

In the growing world of networking, more emphasis is being placed upon speed, connectivity, and reliability. Network performance is vital to businesses for both inter-business operations as well as bringing a product to the consumers through electronic commerce. It has played a part in changing the household, as the interconnectivity of families and friends has changed the way we communicate and seek information.

With the growing number of network users, more emphasis is placed on the maintenance and reliability of these networks. When network problems occur, many times they bring unpleasant breaks in service to those depending on this medium. Sometimes, these breaks of service just cause annoying blackouts of communication, but other times they can cost a company thousands, or even millions, of dollars. This has emerged as a significant problem in all forms of commerce.

To address these problems, a system is needed to insure network availability and efficiency by preventing such costly network problems. The first step is creating a system with the intelligence to recognize what signals lead to network problems. If the problem can be recognized in advance, changing a couple of network parameters can possibly circumvent the problem. Many companies provide a sophisticated suite of network management applications to try to address the problem. These applications simply give the network administrator a great deal of information about the problem after it has happened, hoping this will help the administrator find the cause of the problem quickly. None of the products actually focus on learning to detecting the problem before hand, which would allow the administrator or the program itself to prevent the problem from ever occurring. We propose a system using neural networks not only to detect these network problems

before they shut down a network, but to also find the source of the problem. Once the source of the problem is determined, the problem can be fixed before it surfaces or shortly after.

RELATED WORK

Approaching networking issues with Neural Networks is not uncommon. Similar studies have been done in ATM networks with Connection Admission Control (CAC) schemes. In these works, the authors use fast converging neural networks to predict cell loss in a switch using data from a cell bouncing between two switches (Tham and Soh, 1998). This effort involved a degree of prediction, but offers no solution for determining the problem source or solution.

A group at Rensselaer Polytechnic Institute is doing related research. They first detected changes in traffic patterns through the use of a sequential Generalized Likelihood Ratio (GLR) test. That information was then correlated with a filter to provide various alarms. Their algorithm was "trained" or optimized using five of nine fault data sets, and proved general enough to detect three out of the remaining four (Thottan and Ji, 1998a; Thottan and Ji, 1998b). However, this group used a statistical method for detecting patterns, while we proposed a neural network to learn the patterns leading to network faults. We also focus on predicting problem sources and correcting the problem in a timely fashion, whereas their work was just in general detection.

Another group with similar research goals can be found at the University of Michigan. There is work being done there in congestion control and mitigation strategies. However, this group uses a queuing theory approach rather than a neural network to guide the mitigation efforts. And their mitigation efforts consist of rerouting the traffic instead of implementing a control architecture (Feng, 1999).

ARCHITECTURE

A high level view of our architecture reveals a network with a control agent existing somewhere on a node in that network. This control agent has both, the power to read from and to influence network nodes. The nodes involved would either report the necessary statistics to the control agent or the control agent would poll these nodes.

Data Network

Optimally, we would have tested the system on a local computer lab or a testing lab put together for this purpose. However, neither of these were available at the time of project development. In the lack of the needed testbed, NS, a software network simulator, was used to model the network and different scenarios of network traffic. This, of course, made the integration of a control agent easier, but the design could be implemented on a real network if needed.

In our example, the network consisted of several nodes in a configuration where all of the network nodes were attempting to send data to one node. Each node attempts to send at a random bit rate. A random amount of variance is given to each node's rate to better represent traffic in a real network and possible traffic coming in from other nodes outside of

our simulation. The link capacity between sending nodes were given arbitrary values. Some links were able to handle much more traffic than other links.

Control Agent

We create a control agent containing a neural network that is trained prior to being placed in production. In our simulation, the control agent is part of the simulation code. This enables the agent to easily monitor and influence traffic statistics from each node. Whether on a real network or on our simulator, the control agent gathers information from each managed node and makes a decision about whether network problems will occur. An illustration of the control agent organization can be found in Figure 1. With the predicted problem in our grasps, we can take steps to stop or prevent it.

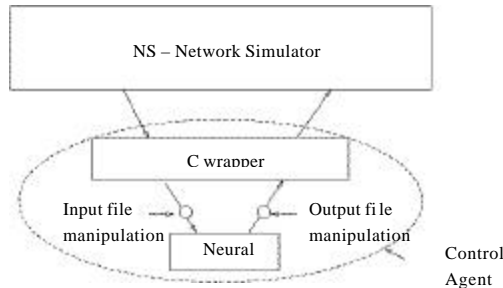


Figure 1: Control Agent Architecture

The Simulation Network

The simulated network is arranged so that six sending nodes are connected to one receiving node through several links which monitor the packets (Figure 2(a)). The sending nodes sending the data are similar to Universal Datagram Protocol (UDP) agents sending constant bit rate (CBR) traffic with a randomized parameter to add variance to the traffic. In NS, each connection is explicitly stated and each sending agent in each node must be configured to send to a particular receiving agent. To determine how fast the sender sends data, the packet size and a packet interval must be given. The sender will send a packet of the required size at the required interval. During the simulation, the control agent executes at a specified polling interval, monitoring the traffic and making decisions.

Neural Network Specifics. The neural network used by the control agent has 18 input nodes, 1 hidden layer containing 6 nodes, and 6 nodes in the output layer. The 18 input nodes correspond to the 6 traffic generating nodes in the network simulation; there are three input nodes for each node in the network simulation corresponding to the average number of packets, variance, and third momentum for each monitored node.

An additional twist placed on the neural network involved pruning the weights to the point in which the neural network reflected the connectivity of the actual network. An example is shown in Figure 2.

In the model of the neural network in Figure 2(b), the hidden layer is representative of the participating nodes in the data network. The statistical data regarding each node is provided to the hidden node representing the actual node as well as to the hidden nodes

representing the actual node's neighbors. This is continued for all first layer nodes of the neural network. As a result, the statistical information from node 1 is given to both the hidden node representing node 1, and the hidden node representing node 5 (1's neighbor.) This claim is important in realizing the relationships between adjacent nodes in a data communications network.

The neural networks were trained off-line, which involved creating a pattern file to learn from. Eighty-eight patterns were used to train the network; half were samples containing no network problems and half had congestion problems at various locations. In training the neural network, early stopping was used, allowing the training to go for about 15000 iterations. In this case, the least squares error was equal to or less than .04%.

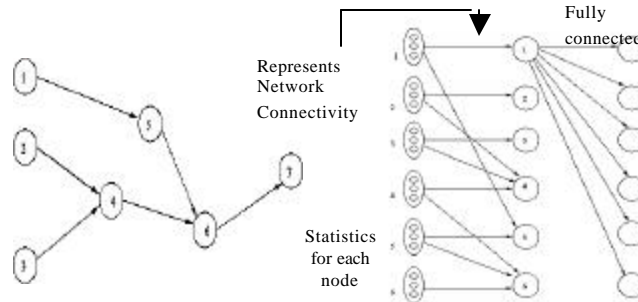


Figure 2: (a) left - example of a data communication network. (b) right - a neural network representing the connectivity of the data communications network (a).

Control from the Agent. As stated before, in NS, an interval and packet size are provided for agents sitting at the sending nodes to determine bandwidth used. The agent will send one packet at every interval. Therefore the smaller the interval, the higher the bit rate. If our neural network predicts that a particular node will be responsible for congestion, we accuse the predicted problem source of using too many resources. To punish the predicted cause node, we add to its sending interval by a small Δt , thus reducing its bit rate. For example if Node 1 was predicted as the problem source, the following equation would explain how Node 1 will be corrected ($Interval_1 = Interval_1 + \Delta t$).

This delta was chosen to be small representative to your time scale, because we don't want to take the chance of over-correcting or even worse, our prediction being wrong and applying a large correction to the wrong node. To take into account the small Δt , the interval in which our control agent executes is also relatively small. Therefore many of these small corrections can be applied and correct a problem slowly without drastically changing any one node's level of service.

RESULTS

A general breakdown of the results can be found in the exploding pie chart of Figure 3. Figure 3 shows that our current application failed in about 10% of the cases. Failing can be interpreted by either missing congestion or predicting congestion when there is no problem.

However, as explained later, all 10% dealt with cases in which the detector missed congestion. In the remaining success cases, 35% were cases in which the detector applied no correctional procedures, but none were necessary. The last 55% of the correct cases, were cases in which the detector accurately applied correctional procedures. A more detailed description of the test simulations is given below.

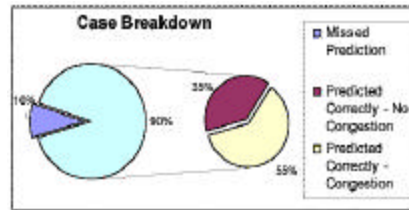


Figure 3: Breakdown of Results

We ran thirty-one simulations of networks. Eleven of those thirty-one were simulations of a network without congestion problems. In these cases we would want our control agent to realize it doesn't have to do anything. This happened in ten out of the eleven cases; however, there was one case that our agent unnecessarily applied a single small fix to a single node. The correction the control agent applied was with a single Δt , and therefore was minimal.

Twenty of the thirty-one simulations had various levels of congestion in various locations in the network. Of these twenty cases, we were able to predict the cause and fix the problems of seventeen cases. Of these seventeen, there were eleven cases in which we were able to fix the problem before the problem even surfaced in the network. These were truly remarkable results, because the congestion was completely eliminated before ever occurring. The other six cases were fixed in less than 3.5 seconds after the congestion appears.

This leaves the three cases in which our neural network missed the threat of congestion. Even this news is encouraging however, because these cases had some common characteristics. The neural network had trouble detecting when a single node in a particular part of the data network caused a problem. This can probably be easily fixed upon close examination of the training patterns and structure of the neural network.

FUTURE WORK

An extended architecture can be formed to suit the biggest communication networks by dividing these large networks into domains small enough to influence easily (most large networks already contain these divisions.) When this is in place, a separate protocol or addition to current protocols can be used to determine domain-to-domain agreements.

As illustrated in Figure 4, the learning control agent would be somewhere within each domain defined in the network. Using the forecasting and detective powers of the agent, each domain would be regulated and operating at safe levels.

To negotiate any problems between domains, the control agents will communicate with each other as to the effects of one domain on another. The control agents will take this

information into account just as it took the information from local nodes into account. This will promote the same safe state of operations between domains as it will inside of each domain.

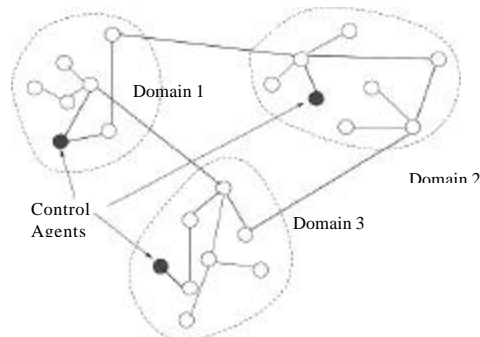


Figure 4: Extension Of The Architecture

CONCLUSIONS

In this work, we set out to show that a Neural Network is a viable method of implementing a learning mechanism for data communication networks. We have illustrated, through the use of a network simulator, that a Neural Network can achieve great accuracy in predicting the particular network problem of congestion. We realize that many more problems exist, but predicting congestion is the first step. We have also shown how a carefully constructed neural network can achieve above average results when structural information about the actual data network is used to form the neural network. This is a feature that forces the neural network to only consider the relationships of nodes we think are important.

A learning mechanism can be of great value as a network manager. The generalization power of a neural network is particularly appropriate because of the unpredicted variance of parameters the manager will have to deal with. Neural networks are certainly an appropriate mechanism for decision making in pro-active network management, and should be the target of more research.

REFERENCES

- Wu Chang Feng, "Improving Internet Congestion Control and Queue Management Algorithms," PhD Thesis, The University of Michigan, Ann Harbor, Michigan, 1999. Computer Science and Engineering.
- C.K. Tham and W. S. Soh, "Multi-service connection admission control using modular neural networks," Proceedings of the Conference on Computer Communications, pp. 1022 - 1029, San Francisco, California, March 1998. IEEE Infocom.
- Marina Thottan and Chuanyi Ji, "Adaptive thresholding for proactive network problem detection," Proceedings IEEE International Workshop on Systems Management, pp. 108 - 116, Newport, Rhode Island, April 1998.
- Marina Thottan and Chuanyi Ji, "Proactive anomaly detection using distributed intelligent agents," IEEE Network, Special Issue on Network Management, pp. 21 - 27, September 1998.