Modeling and Simulation of Internet Network Fault Diagnostics and Optimization using Neural Network

Mrs. Puspita Dash

Assistant Professor, Dept of computer sciences
GKCET, Jeypore, Orissa

Abstract:

The fault diagnostics system constructs a neural network model of the performance of each subsystem in a normal operating mode and each of a plurality of different possible failure modes. This work explores the implementation of a neural network (NN) for analysis and decision-making purposes within the realm of IP network management(IPNM) .The majority of NN models examined in this thesis use RTA and Loss, but Time is also included as an input for some of the models.

We shall start this work by addressing RQ1 and RQ2 and built a NN model that will a benchmark on which to base the rest of our work. This initial model will then be examined to determine if outliers existed and whether their removal will improve the benchmark model. In order to determine where the network may have been experiencing trouble when classifying the sample event data, a translated model configuration is used. Unlike the previous NN model which had a single output, this model had three outputs, one for each state the network is tasked with determining.

I. Introduction: This work explores the implementation of a *neural network* (NN) for analysis and decision-making purposes within the realm of IP *network management* (IPNM). The current landscape of *network management software* (NMS), explore the possible benefits of a NN augmented solution, examine the efficacy of a NN in several scenarios where it must classify network element performance data, and compare these results against more familiar linear regression methods.

NM is represented by McCabe [1] as a layered pyramid (Figure 1) ordered in a “top-down approach, with the most abstract component (business management) at the top of the hierarchy, and the most specified, concrete component (network-element management) at the bottom”. This research falls within the element layer of this hierarchy, using performance data from network-element layer, with implications for the network—and by extension—the service and business layers.

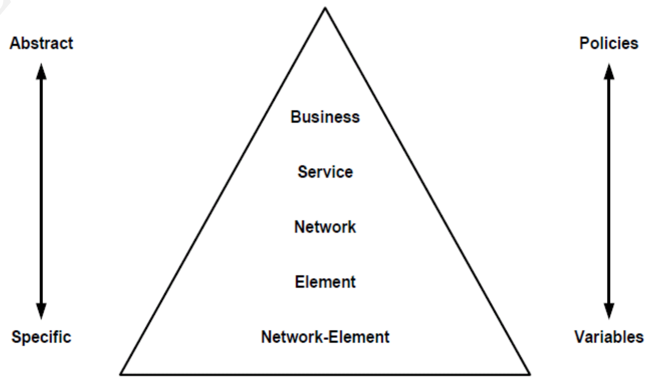

Figure 1: Network Management Hierarchy [1]

1) The performance metrics considered will be latency, calculated as Round Trip Averages (RTA) and represented in milliseconds; and Loss, represented as a percentage of total packets lost.

2) The IP network elements surveyed will not be all of the same type.

3) The dataset used for this thesis was obtained from a commercial IP. It is composed of more than five hundred thousand

events generated by over eighty network-elements.

4) The Neural Network (NN) approach will be implemented for data generation and analysis.

II. REVIEW OF PREVIOUS WORKS:

Growth in computing power over the years is nothing less than extraordinary; today we can fit more computing power in the palm of our hand than was conceivable not even twenty years ago.

With computer networks ever-expanding, the complexity and importance of monitoring these systems become increasingly important. More often than not a network operations center (NOC) monitors thousands of sources (some more thoroughly than others) and relies on human analysis and intervention to recognize and intervene when a failure occurs. As stated in [10], many network operators are left to wonder how they can even begin to monitor the operations of their network in a fashion that both detects and corrects faulty operations and allows them to plan for the future.

III. PROBLEM STATEMENT:

Today's NOCs have must monitor hundreds, thousands, or millions of network elements (servers, switches, routers, etc.) multiplied by the interfaces and services running on each one-and growing every day. As Cisco's top NM expert said, "If you think about all the state that is maintained and manipulated on any one of those nodes…then it is easy to become depressed about the prospect of having to manage all of this information" [11]. Gathering information from all of these sources can be done with relative ease with tools like Simple Network Management Protocol (SNMP) [12], [13], [14], Ping [15], [16], Trace Route [17], and others. Understanding what this information means for the health of a network and tracking down problems is where things become difficult.

NM as part of a hierarchy consisting of business, service, network, element, and network-element management. McCabe explicitly defines NM itself as, "The management of all network devices across the entire network" [18]. Terplan described it as "deploying and coordinating resources in order to plan, operate, administer, analyse [sic], evaluate, design and expand communication networks to meet service level objectives at all times, at a reasonable cost, and with optimum capacity" [19]. Still others have described it as a concept encompassing fault, accounting, configuration, performance, and security management (FACPS) [20].No matter how you define network management, the real challenge is in creating a tool with enough features "to make it useful for the service provider but not overwhelm them in cost and management complexity" [21]. Some NMS (including [22], [23], [24], and [25]) are able to perform automated discovery of elements attached to the network using port scanning-that is, the process of iteratively checking a network-element for open ports and then (based on which ports are open) determining what services are available.

IV.Network Management and Neural Networks

Introduction to NM: NM as a set of functions to control, plan, allocate, deploy, coordinate and monitor resources.

NM as part of a hierarchy consisting of business, service, network, element, and network-element management.
The six subsystems of NM are accounting, Configuration, Fault, Performance, Planning and security.
Capabilities:
Most NMS available today focus on performance and fault management and can have their capabilities broken down into three categories: monitoring, correlation and analysis.

Neural Networks:

NMS has traditionally relied on rote logic and expert system/expert problem solver style intelligence. This research discusses the idea of implementing a NN for analysis and decision making purposes within the realm of NM.

The direction of developing systems that mimic the brain's pattern recognition functions and this is where we enter the domain of NNs. The brain is capable of complex thinking because of its massively parallel nature. This is the true benefit of a using a NN for problem solving; it has the ability to generalize based on learned information. After the NN has been trained, we can present it with inputs it has never learned a response to. It will then, based on what is has learned previously, formulate an output that is consistent with its knowledge. That is, if given enough training information, the NN can analyze the unfamiliar data it is presented with and, based on the previous identification of similar cases, rationalize a solution. This is a form of inductive reasoning, and the more data made available and time spent on training, the more valid the conclusions become.

The Neural Network:

NN models are algorithms that are designed for learning and optimization and are patterned after research into the way the human brain works. Unlike other knowledge-based systems that are pre-populated, NN models require experience to "learn", that is, "[They] use external data to automatically set their parameters" [27].

Their design is such that "instead of storing knowledge explicitly in the form of frames, semantic nets or rules, the knowledge is stored implicitly in the strengths [weights] connecting nodes" [28]. Figure 2.3 shows the adaptive system design process by which a NN uses a feedback loop to calculate error and "learn," i.e., the NN improves as its error is minimized.

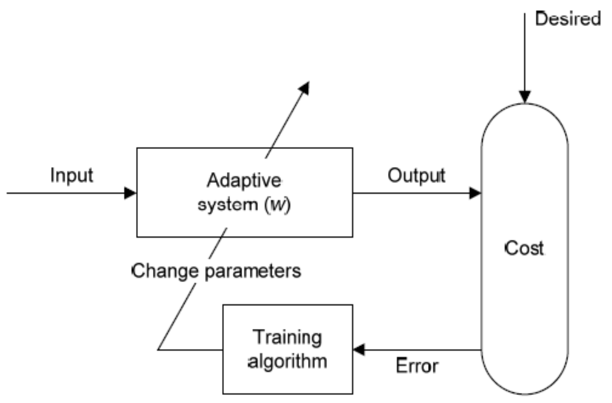

Figure- 2.3: Adaptive system design [42]

Within this thesis, the focus will remain on the simplest implementation of a NN, the *multilayer perceptron* (MLP). The reasoning for this is to give our research a good baseline from which we may adjust, compare, contrast, and improve future models.

Artificial neural networks:

Artificial NNs are attempts to model their biological counterparts found in the brain. While the human brain is quite complex (consisting of 1011 neurons and about 1015 synapses [29]), advancements in neuroscience have allowed us to grasp the workings of the brain on more foundational levels. Neurons found in nature are the basis of all memory, logic, reflexes, etc. The operation of these neurons is complex to be sure; however, the basic concept is this: a neuron collects information from previous neurons.

If the sum of the collected information exceeds the threshold defined in the neuron, the neuron will pass its own signal on to other neurons. In Figure 2.4, we see a diagram of the basic components of two natural neurons. The three most important components of these neurons are the *dendrites*, *axon*, and *synapse*. The dendrites are chemical fiber receptors that extend from the cell body (soma) and serve as the inputs for the neurons. A neuron will have many dendrites, with many branches, each connecting to the axon of other neurons. The axon is a chemical emitter that functions as the output of a neuron.

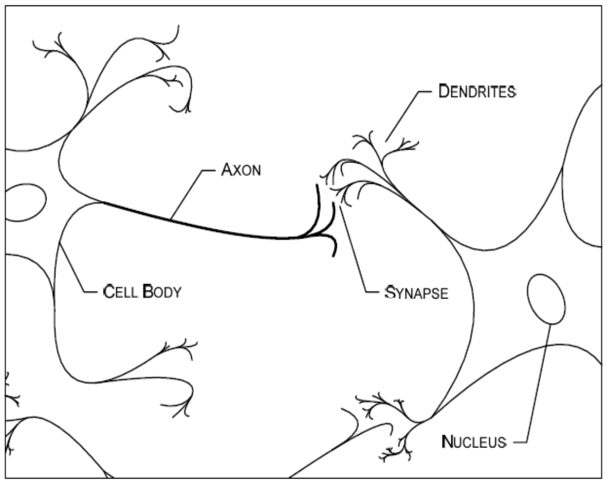

Figure -2.4: Natural Neuron

Each neuron only has a single axon and this axon has minimal branches. The connection between the dendrites and axon is called a synapse. However, this isn't a connection as we would normally think of one; it is more like the spark gap of a spark plug found in an engine. Except in the case of a synapse, chemicals are emitted instead of an electrical spark jumping the gap between axon and dendrite.

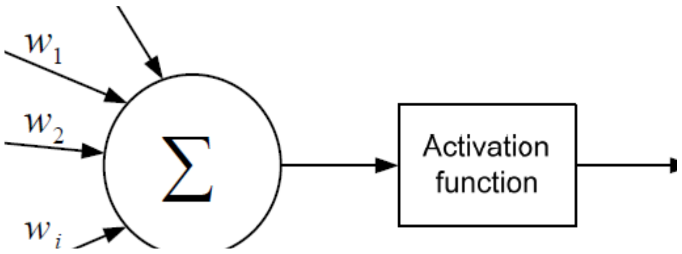

Figure- 2.5: An Artificial Neuron [80]

Consider the logical diagram of an artificial neuron in Figure 2.5; every artificial neuron consists of $x_1 \rightarrow x_i$ inputs, each adjusted by their associated input weight (w_i). This weight could be considered a measure of how relevant the input is to the neuron. A larger weight increases the influence an input has; conversely, a smaller weight (or even a negative weight) decreases the influence of input stimuli. These weighted inputs are then summed and, if it exists, bias (b) is applied. This value is then either passed to the output directly or run through an *activation function* that either "squashes" the result and/or determines whether or not to pass the value on to the next neuron.

Activation functions:

Before a neuron passes the result of its calculation of the weights and bias applied from the input(s) to the output(s), the result must satisfy a certain threshold. These thresholds are called activation functions and are an important part of the process of determining what information is or is not important enough to pass onto the next layer of neurons. There are many activation functions in use—Figure 2.6 shows some of the most common: linear (a), ramp (b), step (c), and sigmoid—including logistic (d) and hyperbolic tangent (e) (characterized by their recognizable "S" shaped curve) [30]. A NN can include any number or combination of these functions. In the case of this work, we concerned with models built using the hyperbolic tangent (e) function.

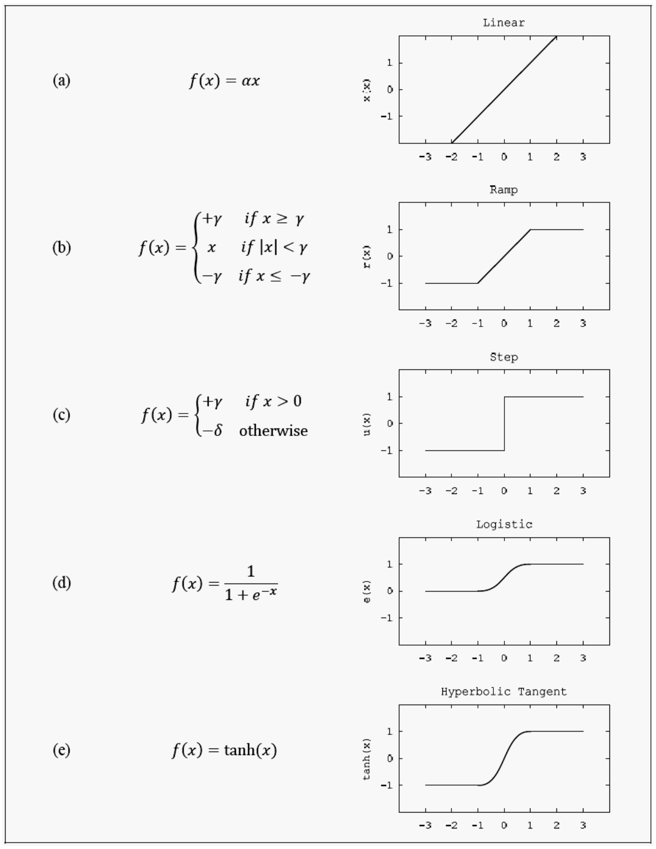

Figure 2.6: Activation functions [81]

Learning:

It is by adjusting the weights of the inputs (w_i) for each neuron that the output is adjusted and the network is trained. Every network must go through this process; how we set these values depends on the training method utilized.

In this research, *backpropagation*, a supervised *error-correction* learning process, is utilized and is therefore the focus of our continued discussion.

Backpropagation:

The basic idea of backpropagation involves taking a NN system and feeding the output error back into the system in such a way as to let the system adjust its own internal weights and bias in an effort to minimize the resultant error (thus, error-correction). This is done by first calculating the error for the network output by comparing it to the desired training output, and then computing the change necessary for each weight connecting the output layer to the hidden layer and the hidden layer to the input layer of the network (as shown in Figure 2.8). This process repeats until the output *mean-squared error* (MSE) reaches a level that is acceptable by the creators of the model.

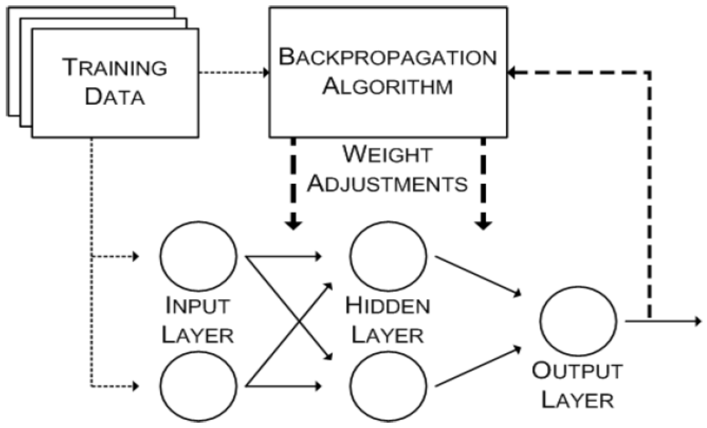

Figure 2.8: Backpropagation in a NN

For backpropagation to work, a classified training data set must exist. As shown in Figure 2.8, this training data is fed into the NN wherein each of the neurons, starting with random weights and bias, processes the data and passes the result on to the next layer. This process repeats until the output layer is reached. At this point the MSE is calculated, the weights between the layers are adjusted, and the process starts again.

V. METHODOLOGY, RESULTS AND DISCUSSION.

Method:

This work will use NN modeling software called NeuroSolutions from MATLAB tool kit, Inc. [26] to preprocess the dataset and create the NN model. The dataset used for this work is obtained from a commercial IP and, while this dataset contains over five-hundred thousand events from over eighty network-elements, not all of these events are relevant to the scope of this work. As such, the dataset was filtered so that only data from seven network-elements of interest was present. Due to limitations with the version of the NN modeling software used for this thesis, the data was further trimmed to three-thousand events. From this dataset, samples of three-hundred events were selected due to further limitations of the NN software. Statistical analysis was performed on the data using Microsoft Excel and SPSS.

Design:

As can be seen in Figure 3.1, three hundred events were randomly selected from the dataset of three thousand events (1). Except for the benchmark, the data was then subjected to analysis and pruning according to z-scores during preprocessing (2), after which the data was separated into three groups (3):

• 60% for training

• 15% for cross-validation

• 25% for testing

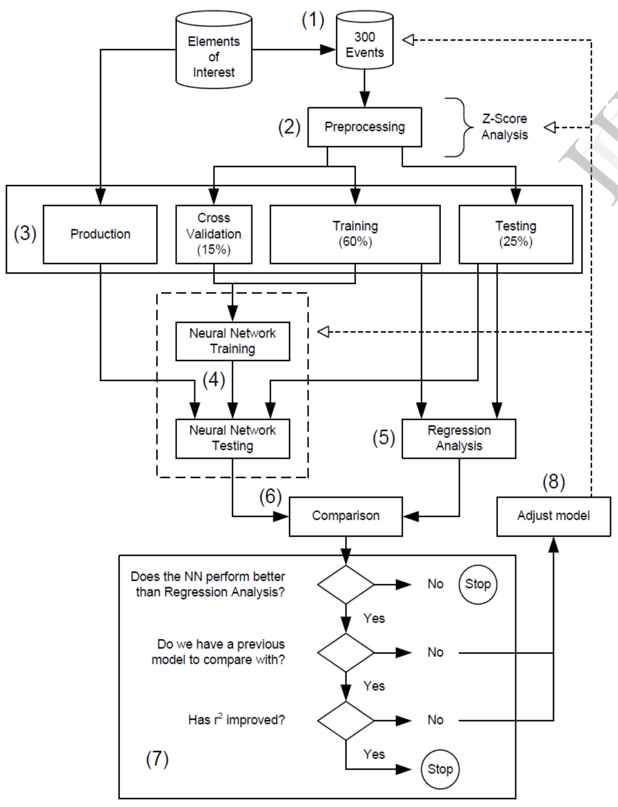

Figure 3.1: Research Methodology

As is shown in Figure 3.1, a separate sample of thirty events is selected as production data. This sample is randomly selected from the dataset of three thousand events in order to address RQ2. The training, cross-validation, and testing data are then run through the NN modeling software (4). To give an initial benchmark to compare the NN against, regression analysis was also applied (5) to the training and testing data. After this, the outputs of the NN model and the regression analysis are compared (6) and subjected to further analysis (7). The result of this analysis might require the model to be adjusted (8), in which case the cycle starts over with adjustments (e.g. identification and elimination of outliers at the preprocessing stage (2)).

Status Representation:

The status representation for the original data was split into three categories; 1, or "OK" status; 2, or "WARN" for conditions that are not detrimental to the network but should be of concern; and 3, or "CRIT" for critical status—when the network element is experiencing severe performance degradation.

Status		1	2	3	RTA	Loss
OK	1	1	0	0	< 300	< 30%
WARN	2	0	1	0	> 300	> 30%
CRIT	3	0	0	1	> 500	> 60%

Table 3.1: Status Categories

To better allow the NN to process these symbolic representations of network status, a decision matrix was established based on a unary encoding scheme provided by the NN modeling software (also shown in Table 3.1 and Figure 3.2 as a matrix).

Model:

This research utilizes a simple MLP NN with a single hidden layer between the input and output layers. This initial model uses supervised learning and backpropagation to learn the proper weights of each connection.

The network shown in Figure 3.2 is composed of two inputs: *Round Trip Average* (RTA) and

Packet Loss (Loss). The hidden layer is made up of at least four perceptrons and the output layer consists of three nodes that return either a 0 or 1 depending on the status.

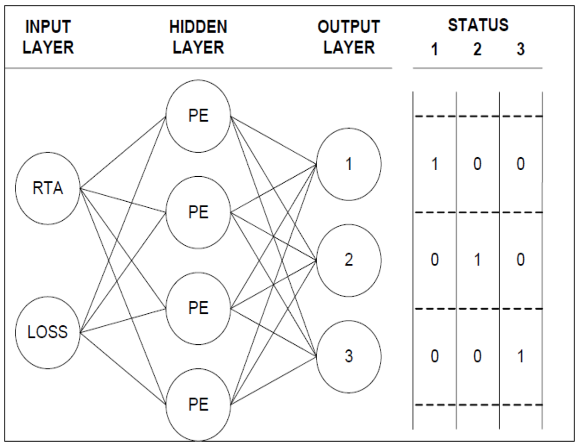

Figure 3.2: MLP Model Diagram

VI. EXPECTED OUTCOME:

Result:

The creation and testing of a feed forward MLP network model (like the network represented in) that uses backward error propagation (backpropagation) for training. The NN modeling software used for this thesis was Neuro-solutions [26]. This thesis had two stages each with two questions. The first stage dealt with the creation and analysis of a benchmark NN model (RQ1), including testing, to determine whether the model was extensible to network-elements not included in the dataset (RQ2). The second stage modified the training data to determine whether modeling on only fault data (RQ3) or temporal data (RQ4) improved performance.

VII. CONCLUSION:

This work will explore the implementation of a NN based approach for analysis and decision-making purposes within the realm of NM. The research itself was focused on developing a NN model that could analyze and categorize sample data from a production IP. The majority of models examined in this thesis use RTA and Loss, but Time is also included as an input for some of the models.

We shall start this work by addressing RQ1 and RQ2 and built a NN model that will serve as a benchmark on which to base the rest of our work. This initial model will then be examined more fully in an effort to determine if outliers existed and whether their removal will improve the benchmark model. While ultimately the elimination of outlier data assisted in smoothing the learning curve for the NN model, it did not produce a model with an improved; that is, an improved ability to classify the event data. Analysis of the sample data for the model showed a high correlation between the inputs when extremes were represented. (When Loss reaches one hundred percent, RTA will return an infinite result; this is due to traffic no longer being passed to the network element.) Elimination of these outliers was not necessary to improve the model, but doing so did improve the linear regression benchmark models. Calculating z-scores forth data and eliminating outliers based on this was also examined. However, this too did not yield an improved for the model.

In order to determine where the network may have been experiencing trouble when classifying the sample event data, a translated model configuration is used. Unlike the previous NN model which had a single output, this model had three outputs, one for each state the network is tasked with determining.

VIII. FUTURE SCOPE:

A NN learns by storing its data as weights between its neurons. Using rule extraction, this information can be used to lift the veil on the black box stigma that NN implementations have received. Additionally, on a smaller scale these rules could be used without the NN as advanced filters for events within the network. While this is not a full-fledged implementation of a NN, it could make handling and making decisions on large amounts of network event data easier to handle. Other issues to be addressed include scaling the NN model to include more than RTA, Loss, and Time information as its inputs. This would include data easily gleaned from SNMP queries like interface statistics for incoming and outgoing packets including: counter information, errors, discards. Additionally, testing should be done to examine the applicability and inclusion of network-host data in order to use a NN model to monitor hosts and their services.

IX. REFERENCES:

[1] J. Finlay and J. Jones, “Neural networks for system fault management,” in Techniques and applications of neural networks. New York, N.Y., USA: Ellis Horwood, 1993, pp. 287-299.

[2] Miniwatts Marketing Group. (2008, May) World Internet Usage Statistics News and World Population Stats. [Online].

[3] Internet Systems Consortium, Inc. (2008, Jul.) Internet Systems Consortium, Inc. [Online]. [4] K. Barron, “Logistics in Brown,” Forbes, vol. 165, no. 1, pp. 78-83, Jan. 2000.

[5] G. E. Moore, “Cramming more components onto integrated circuits,” Electronics, vol. 38, no. 8, pp. 114-117, Apr. 1965.

[6] M. Dubash, “Moore’s Law is dead, says Gordon Moore,” Techworld, Apr. 2005.

[7] J. Teresko, “Defying Moore’s Law: IBM researches for tomorrow’s nanocircuits,” Industry Week, Dec. 2006.

[8] Microsoft Research Ltd, Towards 2020 Science, S. J. Emmot, et al., Eds. Cambridge, England: Microsoft Research Ltd, 2006.

[9] S. Cheshire and D. H. Steinberg, Zero Configuration Network: The definitive guide, M. Loukides, Ed. Sebastopol, CA, USA: O'Reilly Media, 2006.

[10] J. Walrand, Communication Networks: A first course, 2nd ed., New York, N.Y., USA: McGraw-Hill, 1998.

[11] L. L. Peterson and B. S. Davie, Computer Networks: A systems approach, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2003.

[12] D. Harrington, R. Presuhn, and B. Wijnen. (2002, Dec.) An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. http://www.ietf.org/rfc/rfc3411.txt.

[13] R. Frye, D. Levi, S. Routhier, and B. Wijnen. (2003, Aug.) Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework. http://www.ietf.org/rfc/rfc3584.txt.

[14] J. Case, R. Mundy, D. Partain, and B. Stewart, (2002, Dec.) Introduction and Applicability Statements for Internet Standard Management Framework. http://www.ietf.org/rfc/rfc3410.txt.

[15] Internet Systems Consortium, Inc. (2008, Jul.) Internet Systems Consortium, Inc..[Online].
Ping (8) - Linux man page. [Online].

[16] M. Muuss. (2007, Sep.) The Story of the PING Program. [Online].

[17] Traceroute (8) - Linux man page. [Online].

[18] J. D. McCabe, Network Analysis, Architecture, and Design, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2007.

[19] K. Terplan, “Integrated Network Management,” in Network management and control. New York, N.Y., USA: Plenum Press, 1990, pp. 31-57.

[20] B. N. Jain and A. K. Agrawala, Open Systems Interconnection: Its Architecture and Protocols. New York, N.Y., USA: Elsevier, 1990.

[21] C. Harler, Web-based network management: beyond the browser. New York, N.Y., USA: John Wiley, 1999.

[22] Hewlett-Packard Development Company, L.P. (2008, Jul.) HP Operations Manager Software. [Online].

[23]OpenNMS, (2008, Jul.), DiscoveryOpenNMS. [Online].

[24] Paessler AG. (2008, Jul.) Enhanced Feature Overview: Installation and Configuration. [Online].

[25] SolarWinds, Inc. (2008, Jul.) SolarWinds: Automate Creating Network Maps & Diagrams with LANsurveyor. [Online].

[26] NeuroDimension, Inc. (2007) NeuroSolutions. [Online].

[27] J. C. Principe, N. R. Euliano, and L. W. C, *Neural and adaptive systems: fundamentals through simulation*. New York, NY: Wiley, 2000.

[28] A. Hart, *Knowledge acquisition from expert systems*, 2nd ed.. New York: McGraw-Hill, 1992

[29] M. Young, "The Brain," in *Towards 2020 Science*, S. J. Emmott, et al., Eds. Cambridge, England: Microsoft Research Ltd, 2006, pp. 54-55.

[30] P. K. Simpson, *Artificial neural systems: foundations, paradigms, applications, and implementations*. Elmsford, NY, USA: Pergamon Press, 1990.