高性能 RMON によるネットワークの適応的異常検出の開発

Integrated Learning and Adaptive Detection of Network Anomalies Using High Performance RMON Probes

Hassan Hajji Shadan Saniepour Esfahani
Department of Information and Computer Sciences
Saitama University
Shimo-Okobu 255 Saitama-Shi
Email: hajji@aise.ics.saitama-u.ac.jp
hhassan@jcom.home.ne.jp

ABSTRACT.
In this report, we explain the details of our implemented RMON software, for adaptive detection and identification of network faults, as part of the project supported by IPA. We first give the background of this project, and then overview the internals of the software. Further algorithmic and evaluation results are given in the references [1] and [2].

1. BACKGROUND

Networks and distributed processing systems have become an important substrate of modern information technology. The rapid growth of these systems throughout the workplace has given rise to a discontinuity in expertise of human operators to manage them. There is a need for automating the management functions, to reduce network operations and management cost.
A large amount of work has gone into developing mechanisms and protocols for collecting traffic statistics. Indeed, most of the work in the Internet Engineering Task Force (IETF) concerns with the aspects of standardizing and defining extensive network and system Management Information Base (MIB). Comparatively, the *critical task* of how to use these statistics *to actually carry out the management functions is still missing*.
In this project, we developed an RMON-based network monitoring tool for automatic and adaptive detection of network anomalies. The rest of this report gives the

architectural overview of this software. Further algorithmic details and evaluation results can be found in references [1] and [2].

2. GOAL OF THE PROJECT

The proposed software for network monitoring aims at replacing the current common sense approach of network managers, with principled mathematical models. The ultimate goal is to abstract network management functions to a level high enough, such that even normal users can carry out these functions, without the need of a detailed understanding of the low-level semantics of network objects.

The developed software is an RMON probe running on Linux operation system, with automatic anomaly detection functionality. The principal innovations of this project are:

- **Use of sound statistical methods for traffic modeling**: We implements a novel approach to network operation baselining. Our model is particularly tuned to detect changes in network states, while being aware of network traffic fluctuations.
- **Automatic fast generation of alarms**: formulating the problem of alarm generation as a statistical sequential analysis problem, we are able to both generate alarm with the shortest delay to detection, and set the adaptive thresholds automatically. Users are not required to have advanced knowledge of network low-level semantics.
- **Automatic capture of offending packets**: our software supplies the most important information elements related to anomaly understanding. That is a reproduction of the network activity exactly at the time of network anomaly occurrence, by filtering and logging packets related to the anomaly. This way network anomaly is both well understood, and their recurrence can be avoided.

In contrast with Network Management Station (NMS) based network polling of raw statistics, our software *pushes* only value-added information to the RMON clients. By limiting the overhead on RMON clients, this organizational model scales to large networks. The next section gives a more detailed overview of the functional specification of our proposed software.

3. SOFTWARE ARCHITECTURAL OVERVIEW

This section describes the software for network anomaly detection. Figure 1 gives the full picture. As shown in this figure, the implemented software for automated network monitoring consists of three major components:

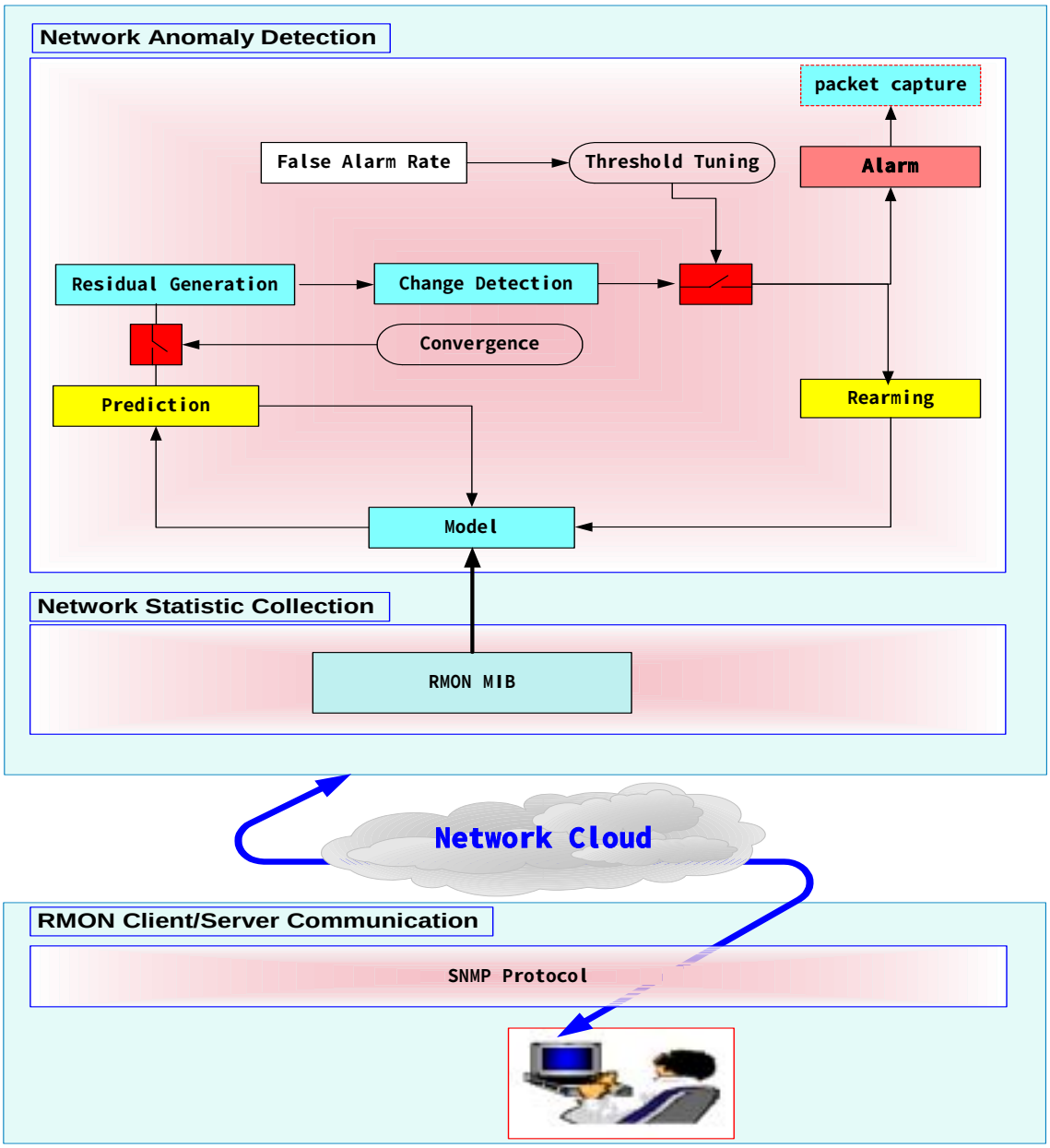


Figure 1 Architectural overview of proposed software

1. **Network Anomaly Detection Module**: monitors network traffic, learns network normal operations, and detects anomalies in real time, with the shortest delay to detection.
2. **Network Statistics Collection Module**: interfaces the network for statistics collections about all aspects of network traffic, from MAC layer up to protocols and applications activity.
3. **RMON Client/Server Communication Module**: implements communication primitives at the RMON servers (RMON probes) such that network monitoring and traffic collection can be controlled in a machine and place dependent fashion.

In the sequel, the three modules functional specification, and architecture are overviewed.

（1） Statistics Collection Module

Statistics collection is performed using Remote Monitoring (RMON) Management Information Base (MIB). Our implemented RMON probe provides detailed statistics about network traffic, ranging from MAC-layer to application-layer activity. We believe that our software will have a major impact on how networks are managed. Note that, currently, the cost of adding RMON capabilities to a large network can exceed $100,000, and yet the benefits are hard to quantify [3]. Figure 2 shows the usual way RMON probes are deployed.

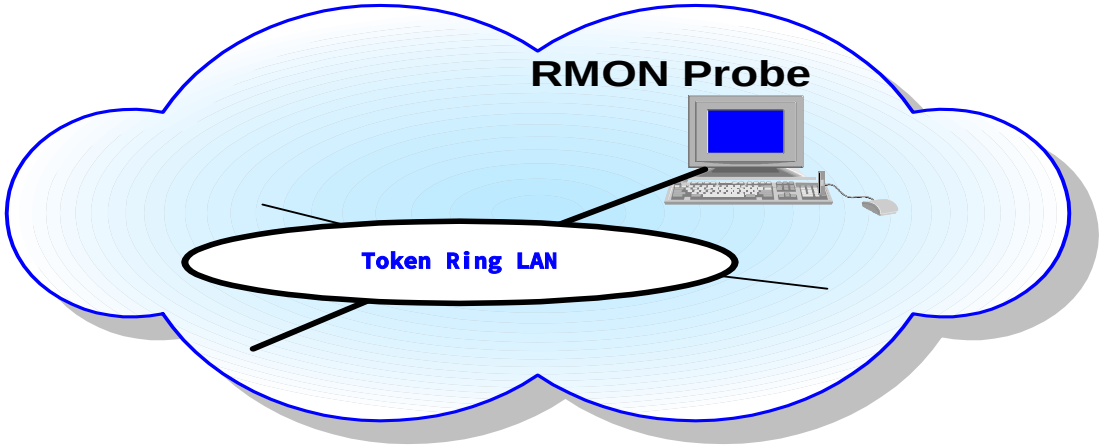


Figure 2 RMON probe monitoring a Token Ring Local Area Network (LAN)

In our implementation, we have designed a link abstraction layer that isolates the Operating System (OS) specific details. At present, our RMON runs on Linux, but porting it to other operating system should be straightforward.

（2） Anomaly Detection Module

Our software detects anomalies by studying the difference between the predicted value and the observed value of the monitored variables. We call this difference *residuals*. Anomaly detection is shown to reduce to the study of the mean of the model residuals. Figure 3 gives an overview the anomaly detection module.

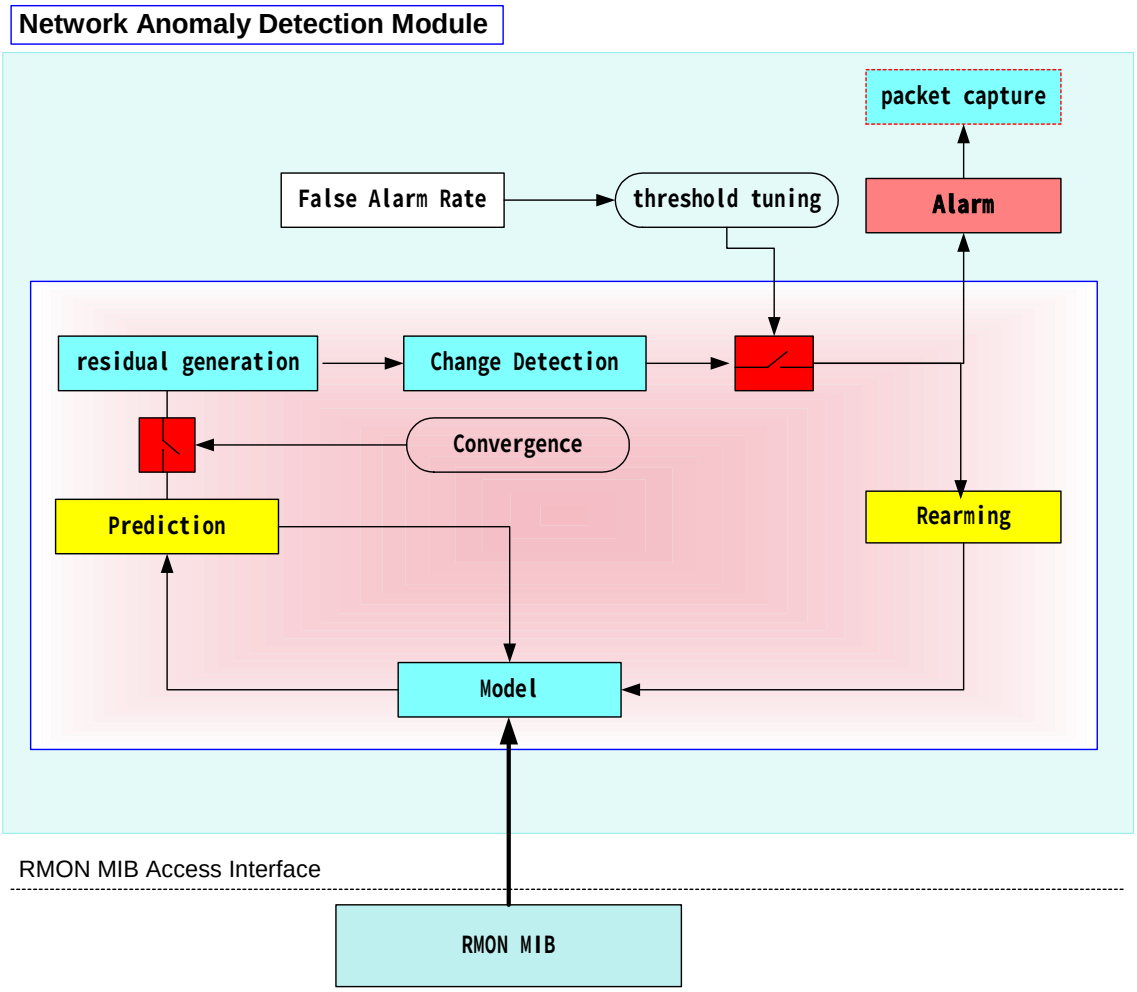


Figure 3 Pictorial illustration of anomaly detection module

As shown in Figure 3, network operation statistics are passed to the model by the RMON probe, through the MIB access interface. The model first iterates until convergence, to learn the network normal behavior. Then, the detection of network anomalies is shown to reduce to detecting changes in the mean value of the residuals. As shown in the figure, the change detection sub-module processes these residuals sequentially, and raises an alarm as soon as the mean value of the residuals departs significantly from 0. To be able to gain an in-depth understanding of the nature of the anomaly, the packet capture sub-module dumps to the permanent storage packets corresponding to the alarm, and generates appropriate notification events.

The remaining of this section describes the internals of the model and how residuals and generated. Then, we show how anomalous activity is detected.

Normal Behavior Model and Residuals Generation

Our approach to network model parameterization is to view the traffic as switching between

different regimes, where each regime is a Gaussian distribution. This is a form of what referred to in the literature as finite mixture model. Each regime corresponds to a given mode of operations. Our approach accounts explicitly for network traffic fluctuations in the modeling process.

Estimation of network traffic parameters is performed sequentially [1]. Compared to batch algorithms, and due to the sequential nature of our learning scheme, it both consumes less memory and time and is able to *adaptively track* changes in network traffic. Figure 4 illustrates pictorially the learning process.

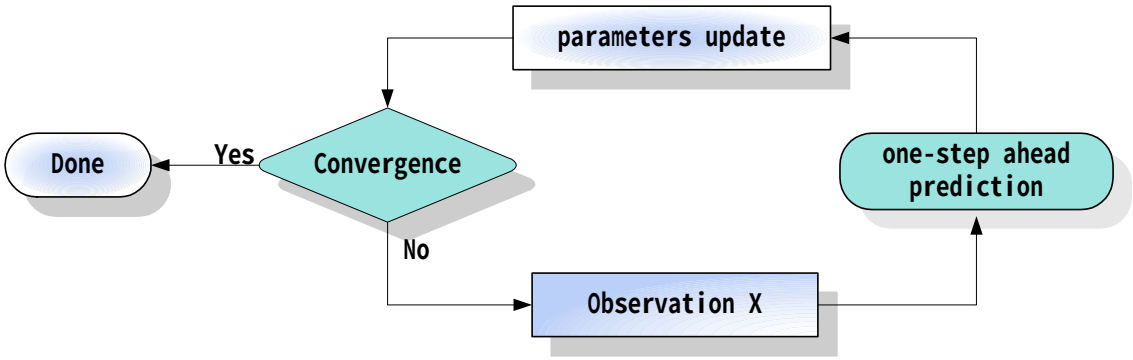


Figure 4 Pictorial illustration of online learning process

Figure 5 shows the convergence process of the online estimation algorithm. As it can be seen clearly, the difference between successive estimates converges to 0, as the sample size grows. This property forms the basis of how we characterize normal operations of the network. That is, under normal operations, the difference between successive estimates is transformed to a random variable that *has mean 0 and variance 1*, by an appropriate scaling matrix [1][2].

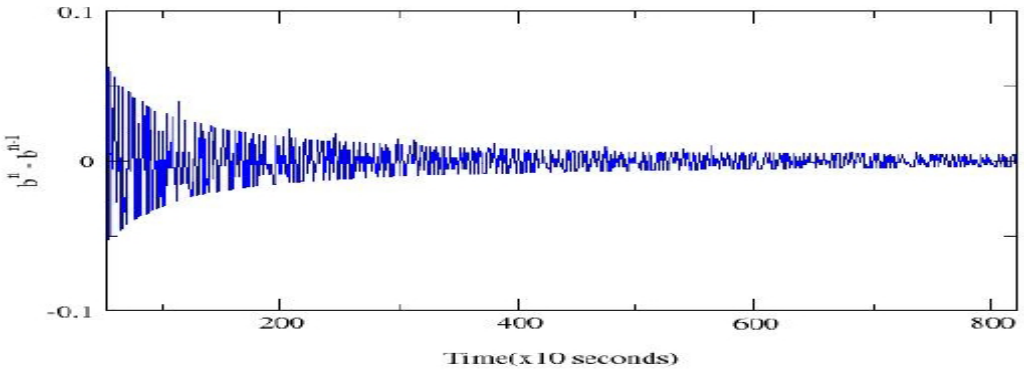


Figure 5 difference between successive estimates of the model parameters converges to 0, as number of iteration grows

Figure 6 shows an example of the resulting residuals. It can be clearly seen that the mean of the residuals is very close to 0 and its variance is 1.

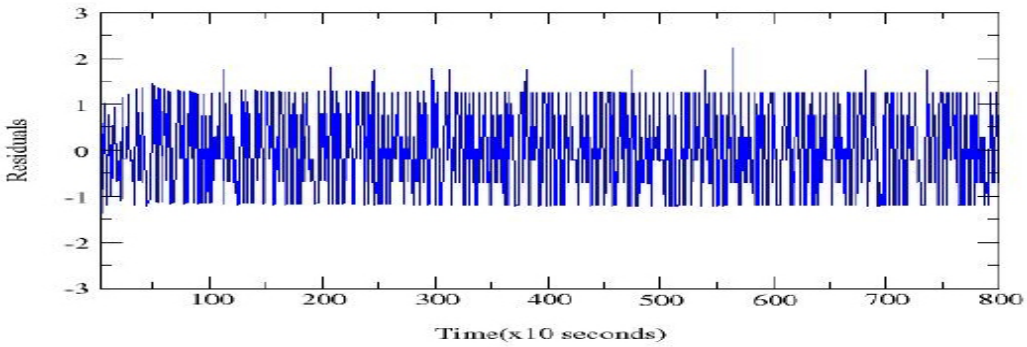


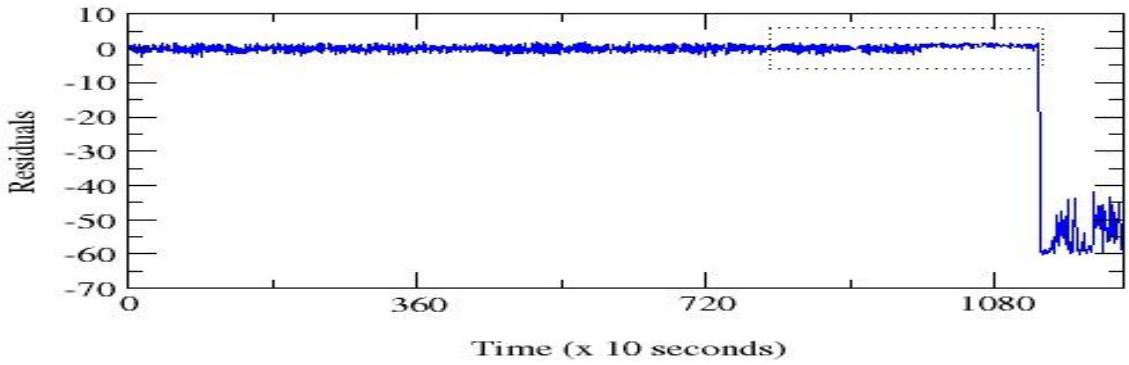
Figure 6 residuals under normal operation have mean 0 and variance 1

In summary, network operation is modeled such that under normal conditions, the mean of the residuals is 0 and their variance is 1. The following section shows how the residuals behave under abnormal conditions, and how detection is performed.

Anomalous Activity Detection

Anomaly detection is determining the discrepancy between the normal behavior and the predicted behavior. We showed earlier how the normal operation of the network manifests itself in terms of the mean and variance of the residuals. Figure 7 shows the behavior of the residuals under a real abnormal condition.

Figure 7 reaction of the residual to network anomalies: sudden jump in the mean



Zooming in the behavior of the residuals just before the problem occurs shows an interesting fact, as illustrated in Figure 8. We notice that the sudden jump is preceded by a slight change in the mean of the residuals. If the detection mechanism is designed to be sensitive to slight changes in the mean of the residuals, we could have detected this anomaly at least 19 minutes before it became serious.

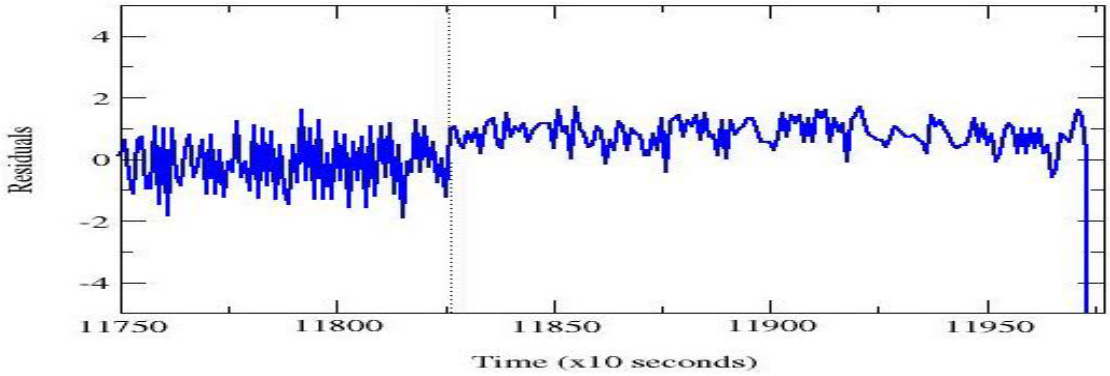


Figure 8 slight changes in the mean preceded the sudden disruption

Our approach to anomaly detection proceeds by studying the residuals *sequentially.* That is, the goal is to detect as fast as possible changes in the statistical mean of the residuals. The decision functions processes residuals online, and stops the first time the *mean departs significantly from 0*. Here the word significant is mathematically formulated as a threshold that controls the first time the alarm is generated [1][2]. The user can specify a desired false alarm rate, and the threshold tuning sub-module, as shown in Figure 3, takes care of translating the supplied value to its corresponding threshold.

Once the anomaly occurs, packets capture sub-module filters the capture buffer, according the variable being monitored, and dumps its contents to permanent storage for later analysis. Appropriate events and notifications are sent to the RMON client, owning the monitoring in course. This way, network operators are given valuable and timely information to demystify the problem, and eventually avoid its recurrence. This functionality is unique to our project, and is not available in any commercially available RMON products.

（3） RMON Client/Server Communication Module

The communication module implements communication primitives between the RMON servers and clients. It allows users to access the servers independently on the client machine, as shown in Figure 9. Through this interface, statistics collection and monitoring control is realized.

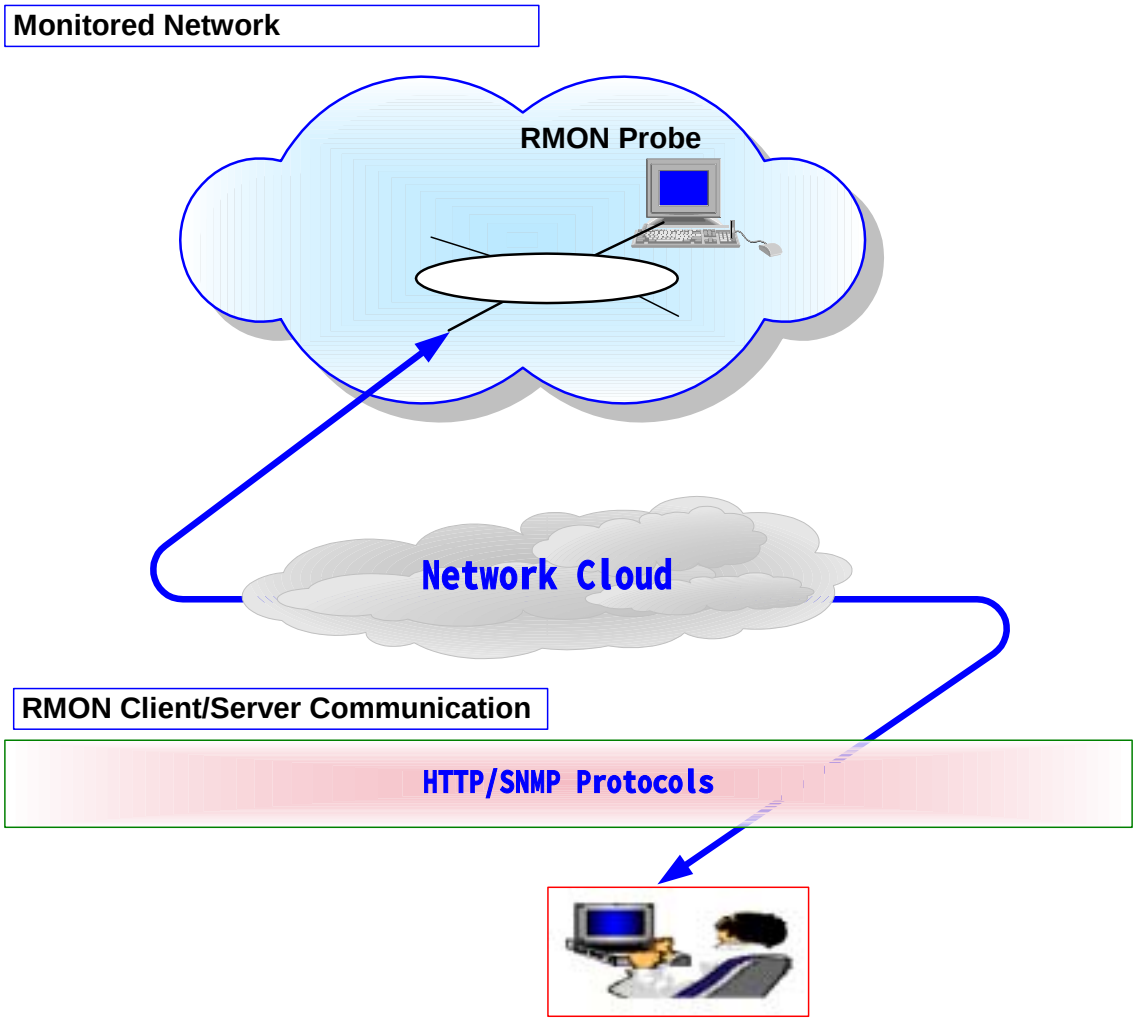


Figure 9 Communication capabilities allows the client to control RMON probes, independently on the client machine

Since RMON is part the whole Simple Network Management Protocol (SNMP) management architecture, SNMP communication access and control capabilities are supported. This allows conventional NMS clients to take advantage of new capabilities of our RMON servers. We provide both client and server communication stacks. We plan to support HTTP protocol in the future. The reason is that, instead of forcing the network operators to sit in front of specific machines, running specific NMS software, support to HTTP protocol allows the user to access and control the RMON probes from anywhere.

4. AN ILLUSTRATIVE EXAMPLE

In this section, a real example is given to illustrate the capability of our software. Figure 10 shows

how test statistic jumps as a reaction to abnormal broadcast packets. The threshold (in red) is crossed, and packets are dumped to permanent storage.

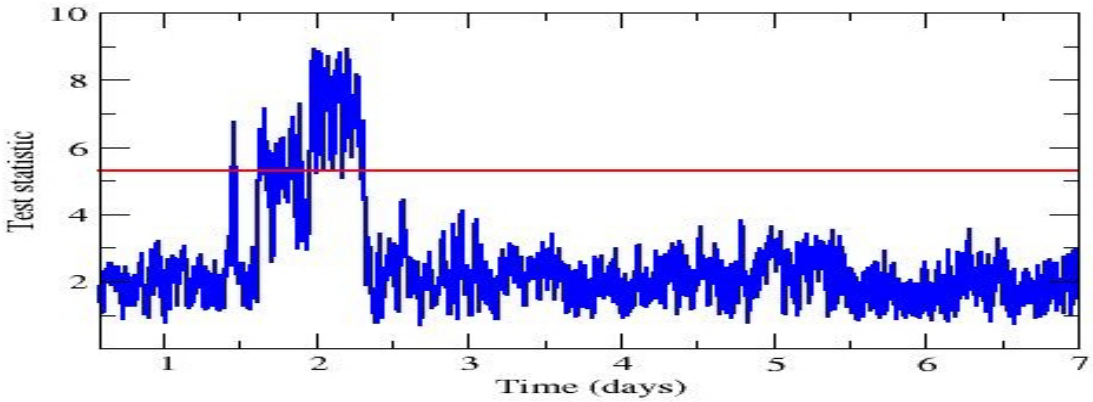


Figure 10 Detection of Abnormal Broadcast Packets

To identify the real cause of the problem, packets are analyzed using our implemented protocol analyzer, and the results are shown in Figure 11.

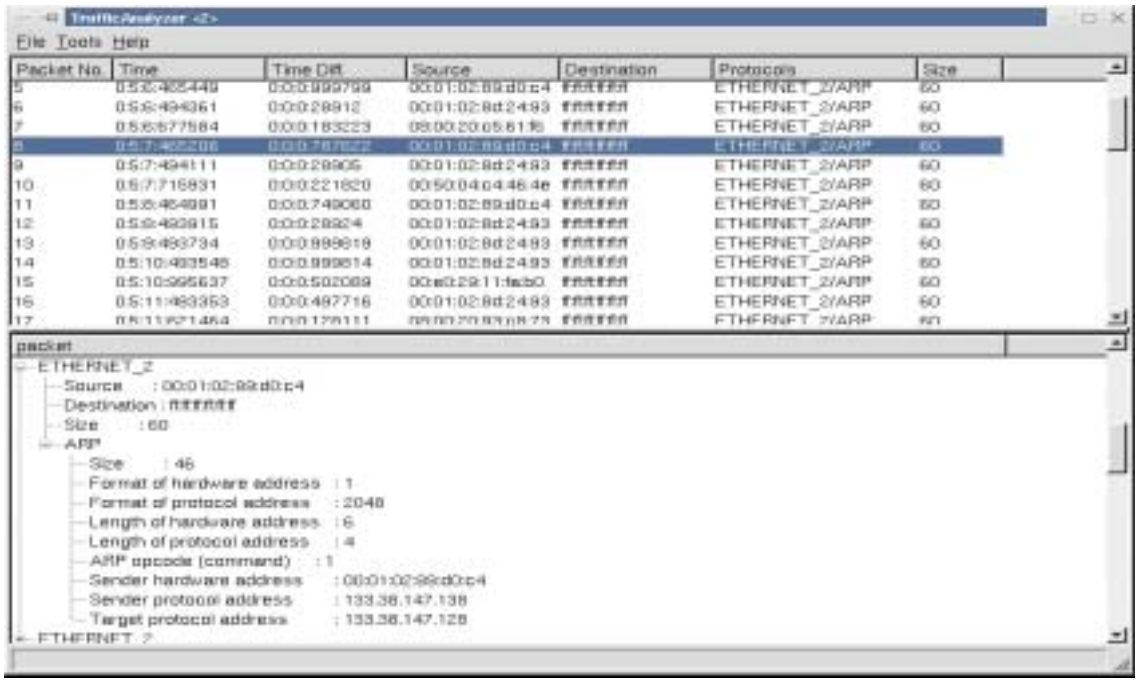


Figure 11 Anomaly Caused By Failure of the NFS Server

As it can be clearly seen here, most packets are ARP packets, destined to resolve the NFS server, but the NFS server is not responding. It has actually found that the NFS server was down, and after rebooting it, the network traffic gained its normal state again.

The example is meant to give only a feeling of the capability of our RMON software. Detailed evaluation results are given in references [1], [2].

5. REFERENCES

1. Hassan Hajji, B. H. Far, J. Chenge. Detection of Network Faults and Performance. Internet Conference 2001. (論文賞を受賞)
2. Hassan Hajji, B. H. Far. “Continuous Network Monitoring for Fast Detection of Performance Problems”. In Proceeding of 2001 International Symposium on Performance Evaluation of Computer and Telecommunication Systems, (学生プレゼンテーション賞を受賞)
3. Paul Korzeniowski, RMON: Capable but costly, in PC Week, 1998