ANFIT: Two-layered Framework for Network Troubleshooting

Nazleeni S. Haron, Zuraidah Sulaiman and Mohd H. Hasan

***Abstract*—This paper proposes a framework for troubleshooting network faults pertaining to Internet applications. The system is called ANFIT which stands for Automated Network Fault Inference Tool. We have designed the system in two layered architecture in order to efficiently troubleshoot the faults. The first layer is dedicated to detect where the network has gone wrong and the second layer is to identify the cause of fault. However, at present we narrow down our focus on Web Service application. We have analyzed real failure scenarios and ANFIT has been tested against them. This paper however, presents only the parts on system framework and overall architecture of ANFIT.**

***Index Terms*—automated network fault identification, network diagnosis, network troubleshooting.**

I. Introduction

Diagnosing why a network service does not work is a difficult and time consuming task, even for someone who knows what they are doing. "Is the net down?" is a common query. The cause can range from an unplugged network cable on the client machine, to the server being down, with many other possibilities in between, such as DNS misconfiguration, routing failure, or link failure. It is also an arduous task due to the fact that network is a complex system with many inter-dependent properties that affect its behavior. Furthermore, the increasing number and complexity of technologies in today's network also contributes in making the troubleshooting process more intricate.

Network troubleshooting is an ideal candidate for automation because the underlying network elements themselves are digital devices inherently capable of computation and communication. Automated diagnostic tool is likely to be as accurate as human diagnosis but much faster because it is capable of rapid identification, analysis of conditions and diagnosis in real time. Besides, human experts are not systems of rules; they are library of experiences which make them sometimes unable to articulate reasoning process sufficiently and precisely. Moreover, the inferences given may vary from one expert to another though given the same problem scenario. Furthermore, human experts are expensive, can be affected by fatigue, emotional states, forgetfulness which makes automated tool a preferred choice.

Even though there exists network troubleshooting tools in the market, but due to their proprietary nature, the technical details were not disclosed. Additionally, despite the numerous kinds of tools, lack of general tool for accurate fault diagnosis has been identified as one of the top problems for network troubleshooting [10].

Internet has been known for its layered model in order to mitigate the complexity of networking. However this excellent approach actually has a significant drawback in terms of error reporting. This is because applications must operate independently of the network environment and lower layers of the network do not generally report meaningful errors to upper layer applications. It is always the case that lower layer network problem can cause upper layer application but without giving any information why the errors are occurring. To worsen things, the nature of applications is that they do not possess any sophisticated methods for identifying network-related errors. As a consequence, none of any corrective measures can be taken as the network does not identify any specific problems for the application. For the normal network user, with limited knowledge, this will only result on them being exasperated and frustrated for not knowing what has happened to their applications.

Therefore, these issues have motivated us in coming up with a simple type of automated and extensible network fault diagnosing tool that is user-friendly to different type of users; be it the normal network users or the advanced users such as the network experts or professional support teams respectively.

II. Related Work

There have been numerous studies on network diagnostic approaches. Brodie, *et al*. describe an architecture using Bayesian networks and how to use probes in order to identify network faults [1]. Lee, *et al*. present an optimal strategy for network diagnosing by checking the candidate nodes first instead of checking the most likely faulty nodes [4]. The idea is somewhat similar to our two-layered approach because diagnosis is performed only to identified faulty component. Thaler and Ravishankar are proposing architecture for diagnosing faults using a network of experts [12]. Lee, *et al*. are also proposing an architecture comprises of a network of intelligent agents to collect data and diagnose faults [6]. Lee suggests CAPRI, an architecture that uses Probabilistic Reasoning Model (PRM) to support autonomous diagnosis of IP reachability [5]. Chen and Bindel describe a novel approach in diagnosing network using unbiased diagnosis

Manuscript received Novermber 23, 2007.

N. S. Haron, is with the Computer and Information Sciences (CIS) Department of Universiti Teknologi Petronas, Tronoh, 31750 Perak, MALAYSIA. (phone: 605-368 7486; fax: 605-365 6180; e-mail: nazleeni@petronas.com.my).

Z. Sulaiman., was with Universiti Teknologi Petronas, MALAYSIA. She is now with Faculty of Management, Universiti Teknologi Malaysia, Skudai, Johor, MALAYSIA. (e-mail: zuraidahs@utm.my).

M. H. Hasan is with the CIS Department, Universiti Teknologi Petronas, Tronoh, 31750 Perak, MALAYSIA. (mhilmi_hasan@ petronas.com.my).

[2]. Li and Bara prove that distributed fault diagnosis can be done using belief networks [8].

Our solution is also inspired by few research works that emphasize on user's limited knowledge in network troubleshooting [3, 9]. Mahajan, *et al.* develop an Internet diagnostic tool called tulip focusing only on performance faults [9]. Emodis [3], another diagnostic tool developed by Hu and Steenkiste is focusing on computing route-sensitive path metrics such as available bandwidth and packet loss rate. Other tools are also developed for network diagnosis such as Shrink [4] and Scriptroute [11]. However, users need to have good understanding of these tools before they can use them [3]. Therefore they are not user-friendly enough to be used by normal network user.

III. Our Work

To develop an accurate automated network troubleshooting tool for a web service application, we require a thorough understanding on how normal network works, classes of network components that might fail and classes of tools available for testing these components. Accordingly, in order to come out with this solution, we embarked on a course of researching and studying on the normal network behaviors on how different things are working together in order to obtain a web page. We then gathered and understood how the existing technical implementations, algorithms and protocols that are associated to the network actually operate. Subsequently, we then determine a set of possible failure cases that may occur in the network followed by figuring out the set of diagnostic tests that can be executed in order to reveal the failures. Some of the related and appropriate protocols, tools and technologies are then adopted and customized to our project needs. All these initial preparations then encourage us in a great deal of coming out with the inference table, which is the heart of the project. After all are set, we then automate those set of different diagnostic tests into code where we then sum up everything with the displaying out of different error messages to the different level of users.

A. ANFIT Conceptual Model

The relationship among all the main components in ANFIT is depicted in Fig. 1.

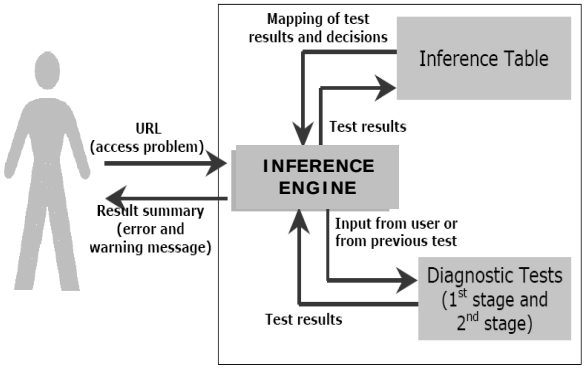

Fig. 1. ANFIT conceptual model

As depicted in Fig. 1, users will need to enter the URL of the web page that they are having problem with in order to start off with ANFIT. This pertinent information will then be passed to *Framework* which is the heart of ANFIT as here lies the *inference engine*. The mechanism that involved is that all the relevant information to be processed in this inference engine is based on the *inference table* that we outlined during the designing stage. This table comprises of a set of diagnostic tests, problem layers and the mappings of diagnostic results and decisions. Besides being the central repository of all the results obtained from executing the diagnostic tests, *Framework* also functions as the inference engine that decides whether or not to launch the further detailed diagnostic test for the particular problematic layers. The results of these thorough diagnostic tests will then again be sent to *Framework* as to allow the final mapping of error codes to the corresponding error messages. Eventually, these error messages or warning messages (where necessary) generated by *Framework* will be displayed to the users which will inform them the cause of failure to obtain a web page that has been detected by ANFIT.

IV. ANFIT Diagnosis Architecture

In this section, we will describe the components that make up our fault detection architecture as illustrated in Fig. 2. The architecture can be divided into three main components, which are Framework, Two-layered Diagnostic Tests and Error Messages.

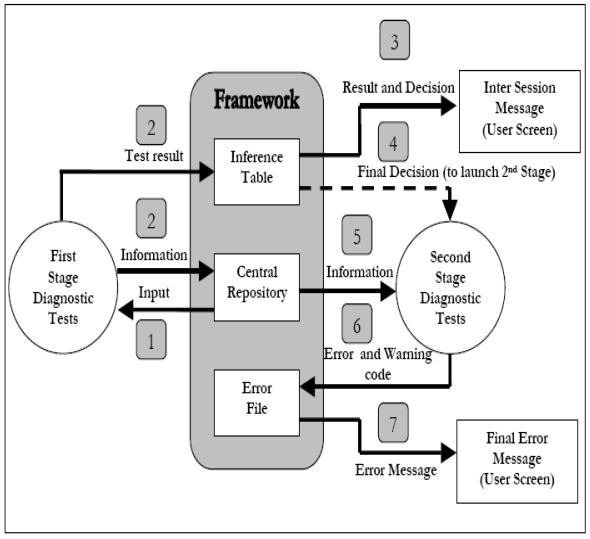

Fig. 2. ANFIT diagnosis architecture

A. Two-layered Diagnostic Tests

ANFIT consists of a two stage tests in order to reduce time in executing all the diagnostic tests. In the first stage, ANFIT can infer the location of network fault meaning, which layer is having problem. As in the second stage, ANFIT will produce detail information about the possible cause of failure. This kind of structure is appropriate because to detect and to analyze are two different tasks that require different kinds of approach respectively. Apart from that, the varied time needed to accomplish each task since the latter consumes more time. Intersession message will be displayed in between the two stages.

1) First stage diagnostic test. This stage adopts a breadth style of diagnosing which will execute all the diagnostic tests sequentially to gather all the results. Each diagnostic test is designed in such way that it is sufficient enough to detect where the error lies, as no aspect of thoroughness is needed yet at this point. It is also at this stage that inference table holds a significant role by facilitating us in mapping the diagnostic test results and decisions, determining the final decision that eventually concludes which problem layer should be launched in the second stage. The breadth style will ensure ANFIT can accomplish running all diagnostic tests in short time yet possess enough and useful information. Essentially, this information is needed to generate the intersession message and to launch the second stage. After careful consideration, the diagnostic tests that have been chosen are comprised of checking the physical line status, checking the host configuration (IP address, default gateway and subnet mask), checking the validity of web server IP address that we obtained from DNS, checking IP level connectivity between the host and remote server, checking TCP connection with web server, checking HTTP connection with web server and checking the existence or availability of the URL.

2) Second stage diagnostic test. This stage adopts a more comprehensive mechanism for each detailed diagnostic test since ANFIT will only execute these tests on the problem layers that have been detected having error as determined by the first stage. These detailed diagnostic tests which will run in parallel, assimilate a more exhaustive analysis and more combination of tools and technologies. It is at this stage that failure scenarios are referred as guideline to design the detailed diagnostic tests. Failure scenarios have been identified to consist of general and specific cases that each should reveal one or more errors. Realizing that, detailed diagnostic tests need to be crafted in order to ensure a higher level of certainties in determining what the cause of failure is and why it occurs.

B. Framework

Framework architecture is further subdivided into three components which are Inference Table, Central Repository and Error File. However, only Inference Table will be further explained in this section.

1) Inference table. The inference table is made up of a series of diagnostic tests and problems that can be identified from the results of the tests. The table is structured in such a way that the first stage diagnostic tests fill the column of the table and problem categories that were devised from failure scenarios gathered, form the row as shown in Table 1.The intersection between test and failure case is called decision pair. It holds the mapping between the result of a test and the corresponding decision.

TABLE 1
INFERENCE TABLE

First Stage Tests / Problem Categories	Configuration Test	Default Gateway Connectivity Test	Root DNS Servers Connectivity Test	DNS Consistency Test	Remote Server IP/TCP Connectivity Test		Remote Server Application Test	
					IP	TCP	HTTP	URL
Local Host Problem	CN - LP OK - UP NO - PR MO - UP MN - LP NE - LP	CN - LP OK - UP NO - LP MO - UP MN - LP NE - LP	CN - LP OK - UP NO - LP MO - UP MN - LP NE - LP	NR	NR	NR	NR	NR
Network Problem	NR	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	NR	NR	NR	NR	NR
DNS Problem	NR	NR	NR	CN - NR OK - NP NO - PR MO - UP MN - LP NE - LP	CN - NR OK - NR NO - LP MO - NR MN - NR NE - NR	CN - NR OK - NR NO - LP MO - NR MN - NR NE - NR	CN - NR OK - NR NO - LP MO - NR MN - NR NE - NR	CN - NR OK - NR NO - LP MO - NR MN - NR NE - NR
Reachability Problem	NR	NR	NR	NR	CN - NR OK - NP NO - PR MO - UP MN - LP NE - LP	CN - NR OK - NP NO - NR MO - UP MN - NR NE - NR	NR	NR
Remote Server Problem	NR	NR	NR	NR	NR	CN - NR OK - NR NO - LP MO - NR MN - LP NE - LP	CN - NR OK - NR NO - LP MO - NR MN - LP NE - LP	CN - NR OK - NP NO - LP MO - UP MN - LP NE - LP

V. ANFIT DIAGNOSTIC PROCESS

In this section, we will highlight on how we infer which layer is the likely cause of failure.

A. Framework

Each diagnostic test will return either of the results as shown in Table 2. We have designed them in such way because diagnostic result cannot be too decisive. The result is not as simple as yes or no situation but lies in those probabilities.

B. Decision Based on Diagnostic Test Results

Each diagnostic test's result will be mapped to either of the decisions as shown in Table 3. The same reason applies here; the each result needs to be mapped to a decision that is flexible.

TABLE 2
DIAGNOSTIC TEST RESULTS

Abbrev.	Meaning	Description
CN	Cannot be tested	The diagnostic test cannot be carried out due to not enough information from the previous test(s) or does not meet the pre-conditions
OK	OK	Explicit indicator of success from that diagnostic test
NO	Not OK	Explicit indicator of error from that diagnostic test
MO	Maybe OK	Implicit indicator of success from that diagnostic test
MN	Maybe Not OK	Implicit indicator of error from that diagnostic test
NE	Neutral	Balanced indicator of error and success from that diagnostic test

TABLE 3
DECISION BASED ON DIAGNOSTIC TEST RESULTS

Abbrev.	Meaning	Description
NR	Not Related	The diagnostic test's result is not associated to the corresponding layer.
PR	Problem	The corresponding layer is having problem.
NP	No Problem	The corresponding layer has no problem.
LP	Likely Problem	The corresponding layer may have problem.
UP	Unlikely Problem	The corresponding layer may not have problem.

C. Preferences of Final Decision

Final decision will determine which layer to be launched in the second stage test. The decision of each diagnostic test will be evaluated to get the final decision. The final decision is derived from the rules as set in Table 4. The leftmost which is the Problem (PR) will have highest priority while the rightmost, which is Not Related (NR) carries the least priority.

TABLE 4
PREFERENCES OF FINAL DECISION

PR >> NP >> LP >> UP >> NR

1) Inferring process. To better understand how the inferring process works, let's look at an example. For instance, if the problem category is Reachability Problem and the results of the diagnostic tests are as follows:

- Remote Server IP Connectivity Test (RSICT) is NO
- Remote Server TCP Connectivity Test (RSTCT) is MN

Therefore, as shown in Table 1, the corresponding decision for the RSICT is PR and the decision for RSTCT is NR. Accordingly, based on our preferences for final decision (Table 4), since PR is higher priority than NR, so we infer that there is a reachability problem with the network. Therefore, this problem will be further checked and second stage diagnostic tests will be launched as to find the cause of failure.

Fig. 3 shows the output from ANFIT given a problematic URL. For every diagnostic test run, the obtained result and corresponding decision will be imparted to the user. After all the tests have been run, a summary of the results will be displayed to the user as to inform where the cause of the fault is. Since this is the first phase of the research, therefore abbreviations are still used despite its original word. More comprehensible output will feature in the second stage of this research.

- Enter the URL: **www.goole234.com/index.html**
- Using domain name: www.goole234.com
- Using port: 80
- End is: index.html
- ----------
- **Running Configuration Test...**
- RESULT OK
- Decision: UP Problem: Local Host Problem
- ----------
- **Running Default Gateway Connectivity Test...**
- RESULT OK
- Decision: UP Problem: Local Host Problem
- Decision: UP Problem: Network Problem
- ----------
- **Running Root DNS Servers Connectivity Test...**
- RESULT OK
- Decision: UP Problem: Local Host Problem
- Decision: UP Problem: Network Problem
- ----------
- **Running DNS Consistency Test...**
- Added error ERR-DNS1_www.goole234.com
- NXDOMAIN
- Added error ERR-DNS1_www.goole234.com
- RESULT NO
- Decision: PR Problem: DNS Problem
- ----------
- **Running Remote Server IP Connectivity Test...**
- RESULT CN
- TCPCONN RESULT CN
- Result CN
- Decision: NR Problem: DNS Problem
- Decision: NR Problem: Reachability Problem
- ----------
- **Running Remote Server TCP Connectivity Test...**
- RESULT CN
- Decision: NR Problem: DNS Problem
- Decision: NR Problem: Remote Server Problem
- ----------
- **Running Remote Server Application Test(HTTP)...**
- RESULT CN
- Decision: NR Problem: DNS Problem
- Decision: NR Problem: Remote Server Problem
- ----------
- **Running Remote Server Application Test(URL)...**
- ==========
- Launching 2nd stage of: DNS Problem:PR
- Local Host Problem : UP
- Network Problem : UP
- DNS Problem : PR
- Reachability Problem : NR
- Remote Server Problem : NR
- ====ERROR RESULTS======
- ERR-DNS1 : The host name for the web page(www.goole234.com) that you requested does not exist. Please check your spelling and try again

VI. Implementation

Two different programming languages were needed to cater for two distinct parts of the project which are the Framework and diagnostic tests (First and second stage). Framework and diagnostic tests are of disparate nature that Java has been picked as the language of choice for developing framework while Python was deployed in implementing diagnostic tests.

D. Java for Framework

The Framework development requires a system programming language like Java, in order to optimize the performance of ANFIT. Furthermore, Java facilitates a good data structure for the framework with its object-oriented feature. As a consequence, it provides ease of integration or interaction among objects such as diagnostic tests objects, problem objects and etc. Java also has support for other languages, so Python commands (in order to execute diagnostic tests) can be run through it. Moreover, the main feature of Framework is the inference table of which is portrayed using XML file structure. XML documents tend to have a very explicit structure that is easily addressed by a language like Java. Although there are available implementations of standard XML parsers in many languages, including C, C++, Tcl, Perl and Python but the XML parser we are using which is SAX (Simple API for XML) is designed in and for Java. SAX which is an event-based interface parser reads the document and tells the program about the symbol it finds, as it finds them. For example, it will notify the application when it finds a start tag, when it finds character data and when it finds the end tag. Thus, the implementation of Inference Table is made feasible by using this approach, hence making Java as the ideal language to use.

E. Python for Diagnostic Tests

Scripting language was decided as the ultimate type of language for developing diagnostic tests as it possesses such traits as providing fast build-cycle turnaround (no compilation needed), facilitating dynamic typing(no declaring of variables needed) , and offering interactive environment where we can create, view or change objects at runtime. Scripting language is also better for rapid development and reusing code which are apt for test automation. All these significant features are very useful to ensure the efficiency and flexibility of the diagnostic tests. All these traits can be found in Python.

VII. Conclusion

ANFIT is designed in two-layered architecture in order to efficiently troubleshooting the network fault. Troubleshooting requires detection and localization of the fault and these two different tasks require different approach. Therefore ANFIT deploys optimal strategy where detailed diagnosis will be performed to the identified faulty components only. In the first layer, ANFIT will perform breadth style of search in detecting the network fault.

Apart from that, ANFIT is intended to integrate isolated yet related existing probing tools/applications into one unify Web Service tool while in the end providing the users with simple and practical suggestions where necessary.

In this paper, we also disclose how our diagnosis work where we have a created an inference table. The table was formulated based on understanding on how normal network works and classes of network components that might fail. Therefore ANFIT is a tool that attempts to calibrate how normal network behavior works and if something fails, it will be able to answer such as question as “Why I can’t get to the website?”.

However, further improvement could be made such as extending ANFIT to diagnose other Internet applications such as FTP, email service or etc. Apart from that, ANFIT may be modified to become platform independent in order to ensure smooth running on all platforms. Another feature that can be added is to provide level of certainty to all responses given to the user. This will act as an indicator for degree of accuracy of the answer.

References

[1] M. Brodie, I. Rish, and S. Ma, “Intelligent probing: A Cost-Efficient Approach to Fault Diagnosis in Computer Networks,” *IBM Systems Journal, 41*, 3, 2002, pp. 372-385.

[2] Y. Zhao, Y. Chen, and D. Bindel, “Towards Unbiased End-to-End Network Diagnosis,” in *Proc. of ACM SIGCOMM*, Sep. 2006.

[3] N. Hu and P. Steenkiste, “Emodis - An End-Based Network Monitoring and Diagnosis System,” Technical Report CMU-CS-05-146, Carnegie Mellon University, 2005, pp. 187-192.

[4] S. Kandula, D. Katabi and J-P. Vasseur, “Shrink: Tool for Failure Diagnosis in IP Networks,” . In ACM SIGCOMM *Workshop on mining network data (MineNet-05)*, Philadelphia, PA, August 2005.

[5] G. Lee, “CAPRI: A Common Architecture for Autonomous, Distributed Diagnosis of Internet Faults using Probabilistic Relational Models,” In *Proceedings of the First Workshop on Hot Topics in Autonomic Computing (HotAC I)*, 2006.

[6] G. Lee, S. Bauer and P. Faratin, “A Scalable Architecture for Network Fault Diagnosis in the Knowledge Plane,” In *Proceedings of the CSAIL Student Workshop (CSW '05)*, 2005.

[7] P. P. C. Lee, V. Misra, and D. Rubenstein, “Toward Optimal Network Fault Correction via End-to-End Inference,” In Proceedings of IEEE INFOCOM, Anchorage, Alaska, May, 2007.

[8] H. Li, and J. S. Baras, “A framework for supporting intelligent fault and performance management for communication networks,” In Proceedings of 4th IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MNS ’01),, 2001.

[9] R. Mahajan, N. Spring, D. Wetherall and T. Anderson, “User-level internet path diagnosis,” In ACM SOSP, 2003.

[10] D. Oppenheimer, A. Ganapathi and D. A. Patterson, D. “Why do internet services fail, and what can be done about it? ,” In Proceedings of 4th Usenix Symposium on Internet Technologies and Systems (USITS ’03), 2003.

[11] N. Spring, D. Wetherall and T. Anderson, “Scriptroute: A public internet measurement facility,” In Proceedings of USENIX Symposium on Internet Technologies and Systems (USITS), 2003.

[12] D. G. Thaler, and C. V. Ravishankar, “An architecture for inter-domain troubleshooting,” *Journal of Network and Systems Management*, 12, 2, 2004.