ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 7, July 2019

Review on Server/Application Alerts Registration and Escalation Tool using Priority Queue Sweeping Algorithm

Gurpreet Kaur [1], Pankaj Sharma [2]

P.G. Student, Department of Computer Science & Engineering, Delhi Institute of Technology Management and Research, Faridabad, Haryana, India[1]

Associate Professor, Department of Computer Science & Engineering, Delhi Institute of Technology Management and Research, Faridabad, Haryana, India [2]

ABSTRACT: Server/Application Alerts Systems endeavor to give a section to hailing the event of an occasion and inciting contributed individuals through cautions or messages. Consistently, they bolster a structure for ceaselessly enrolling energy for explicit sorts of occasions, therefore clearing the need of setting up reliance relationship between occasion makers and customers at creation time. This specific development of notice is pressing: basic pass on and multicast exchanges systems enormous scale appropriated conditions in perspective on the sheer number of beneficiaries. An enlistment based association utilizing need line based similarity can diminish the required fan-out or load on the venture servers. Decoupling the creation and use of data in programming structures empowers extensibility by evacuating express conditions between parts. Expected "alarms/signs in" see structures are incorporated undirected creation and enlistment to occasions by their attributes as opposed to their source to determine the cautions adequately and precisely using priority queue sweeping algorithm.

KEYWORDS: Enterprise Server, Priority Queue, Sweeping Algorithm, Monitoring Services, Remote Procedure Call,

I. INTRODUCTION

Event/ alert monitoring or event logs monitoring plays an important role in modern IT infrastructure systems. There are different kinds of applications, operating systems, network devices, OSS systems available in the market which cater the IT requirement of an organization. In such highly usable IT environment, it becomes very important for an organization to have a healthy and Always ON IT system. For solving this purpose, now a days each and every organization which is based on the IT Infra takes care of their IT infrastructure. Organizations use different types of IT Infra monitoring systems available in the market. These tools may have their own flavor, own limitations and specialties. Some time more than one Monitoring tool is required by the organizations to monitor their IT environment because one tool cannot provide all the required functionalities. For instance, if we talk about IBM's Tivoli Monitoring System version 6.2, which has the capability to monitor Servers, applications, up to some level Network devices etc. it just connect with these different devices installed in an organization's IT environment and collect the health logs* of them which are further become available to the operator to work on them. For example if it is alerted that a particular mount point of a server it more than 90% full, operator will further escalate the issue to concerned team to take action in order to avoid any system break down. So such kinds of other System Monitoring tools are available in the market by other leading companies as well e.g. Nagios, Zabbix, HP's system management etc

This issue looked by the administrator while enrolling and heightening the cautions to the resolver gathering engineers. The administrator has for the most part to perform two noteworthy activities which are Registration of the alarms in the alarm register and raising the equivalent to resolver gathering. An issue territory for an administrator is to

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 7, July 2019

the enlistment of the alarms alongside every single fundamental detail which he would require promptly so as to heighten the caution to the privilege concerned group. On checking device windows, alarms shows up with the constrained data, for instance, if a Server Mount Pt. 90% filled alarm has shown up on the screen, it will contain just the data about the server like Host Name of the server, Mount Pt. which is 90% full, Occurrence Time, Tally (which is reiteration check of the alarm), and so on. By, default ready won't have the data of the resolver gathering engineer who is there in the move to chip away at this. It might not have some specific subtleties like if the alarm is a piece of some arranged action (where the alarm is created in known conditions for some reason), regardless of whether a specific alarm has been raised to up to which levels, an alarm will have the subtleties of the effects which it might do, and so on.

Consequently, as to raise any caution, the administrator is constantly expected to have these subtleties too. For example, Administrator will heighten to a specialist that for XYZ server mount point "/pick" has been filled over 90%, in the event that it isn't settled Billing application running on XYZ server may get affected and it would be a business misfortune. So here administrator has gathered the effect data, which was not there with the alarm, and raised it to resolver gathering, who might make the move as indicated by the need and criticality. At the point when an administrator goes for getting all the above-said data, it takes adequate time and enrolment/revealing/heightening of the alarm gets a postponement and this deferral at some point has huge results like a punishment from a client who's IT condition is being checked. Different outcomes like dependability misfortune on the apparatus, manual human mistake (since administrator needs to physically discover these subtleties) notwithstanding, under this survey we proposed the priority queue sweeping algorithm which will raise the blunder or alarms dependent on the effect factor level or the deadly which may jump out at server or venture application on the fly.

The priority queue is a data model that offers profitable access to the data thing with the humblest (or greatest) key. This is useful when key characteristics show the solicitation where things should be gotten to. Need lines may be used for undertaking getting ready for PCs, where a couple of ventures and activities should be executed sooner than others and are as such given a higher need. A need line is an Abstract Data Type (ADT) offering methodologies that grant departure of the thing with the most extraordinary (or least) key-worth, incorporation, and sometimes various activities. In like manner, with various ADTs, need lines can be completed using a combination of shrouded structures. As of now, we saw a need line executed as a display. The issue with that approach is that, regardless of the way that ejection of the greatest thing is developed in brisk O(1) time, expansion requires moderate O(N) time, because a typical of a huge part of the things in the display must be moved to install the improved one all together.

The alerts are stored in the logs depending on their persistent mode and later put into the priority queue. There are ten queues, one for each priority level 1 through 10. The data structure used for the queues is an array of queues. The size of this array is ten corresponding to the number of priority levels. Alerts are stored in the queue on the basis of their priority. So all alerts with the equal priority are stored in the same queue. Every new alert is always added at the beginning of its queue in order to preserve its semantics. The priority data field in the alert header is used for indexing into the array and getting the queue at that index. Therefore insertion of the alert into the data structure always takes a constant time. Queuing and distribution of alerts are two independent operations. The diagram depicts the scenario below:-

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 7, July 2019

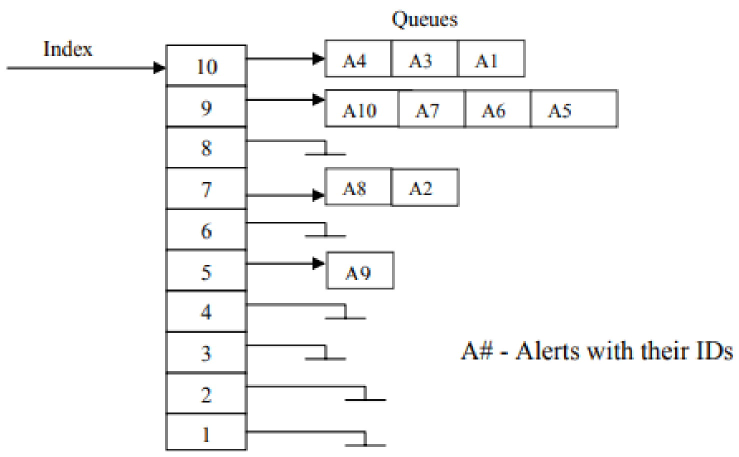

Figure 1: Queuing and Distribution of Alerts using Priority Based Indexing.

II. RELATED WORK

"iPlanet's Java Message Queue (JMQ) [11] thing is a standard based response for the issue of between application correspondence and reliable data transmission across over frameworks. With JMQ, structures running in different models and working structures can basically interface with the identical virtual framework to send and get information. This uses a disseminate/purchase in model. In perspective on Java Message Service (JMS) [12] open standard, JMQ applications are planned to be adequately conservative across over different PC models and working systems and to manage all data elucidation between application structures. Notwithstanding the way that, JMQ gives transport of messages dependent on need and supports consistent quality, it can't be used to move messages between different client applications (LED/GED) [1, 3] in light of the way that it doesn't have any mishap recovery instrument. Beside this, the continuous apportioning the messages are held in the line and the results of advantage flood are not dealt with. This ought to be tended to in a message-circling condition, as one can't anticipate unending resources, as a lot of messages ought to be dealt with whenever. Progressively completed, the heaviness of instatement of the lines and dispensing these lines to unequivocal topics (subjects) with the objective that messages are secured on the system falls on the executive. There is moreover no game plan for clients to set a part of the properties of the message like energy, need, etc. These attributes are set by send/get systems for the Java Message Queue."

"MQ Series [13] from IBM is one of the most settled illuminating things in the business. MQ Series has by and large been organized towards line based illuminating, yet updated in late 1999 to help disseminate/purchase in educating and a JMS interface. The educating clients talk with at any rate one line directors, which are executed in neighborhood code and are available for a wide extent of stages. A line executive is a program that gives illuminating organizations to applications. Applications that usage interface called Message Queue Interface (MQI) to put and get messages from lines. The line chairman ensures that messages are guided to a different line manager. The correspondence between them is through channels. There are two sorts of channels. A message channel is a unidirectional correspondence interface between two line boss that is used to move messages between them. A MQI channel is bi-directional and partners an application (MQI client) to a line head on a server machine for the trading of MQI calls and responses. Line executives are accumulated into gatherings. A pack [14] is a social occasion of in any event two line executives that are wisely related and can grant information to each other. Any line executive can establish a connection on some other line boss in a comparative gathering without the necessity for customer to set up a specific channel definition since this information is held in a storage facility to which all line chiefs in the bundle approach. This makes an issue if the PC on which the store uses DHCP (dynamic appropriation of IP Address) since MQ game plan uses IP address to find file. From this time forward, if the area changes, other line boss will never again have the alternative to find the chronicle. This still applies paying little heed to whether all line executives in the pack will be on a comparative PC, in light of the way that the IP address is up 'til now used to find the storage facility. Thusly, MQ course of action can't manage changes to sort out successfully as it is pre-intended to propel messages from a line to a different line. If any part tumbles in this system, the message coating writing computer programs is stuck until the shelled fragment is working precisely as it doesn't support recovery segment. Progressively completed, it requires in any occasion one guiding procedure on each machine. This isn't reasonable when passing on to various machines. This item moreover requires unprecedented advantages to present or run, as changes must be made to the working system bit on most supported stages. Despite these, MQ Series does not reinforce participation to messages dependent on ordinary verbalizations and crash recovery."

"SonicMQ [15], from Sonic Software, gives a middle point talked execution of JMS bar/sub ass well as point-point territories made inside and out in JAVA. SonicMQ in like manner suits JMS enlargements like XML messages and server gathering. The limit of an advising server to pass on messages at a consistent fierceness (paying little personality to distributer speeds and the amount of relationship with the server) depends to a huge degree on stream control figurings used by them. In an average illuminating condition, message creators are regularly faster than the buyers of the message. To ensure beyond what many would consider possible (memory, strings, etc.) inside server, the sending client must be throttled to maintain a strategic distance from the loss of messages or a help boss must be given on the server. A particularly critical piece of any stream estimation in a disseminate/purchase in educating server is to ensure that if a particular endorser backs off, various supporters on a comparative subject are not negatively impacted. That is, a singular moderate endorser should not ruin the entire structure. SonicMQ handles this issue by buffering messages inside and throttles distributers when inside pads flood. This truly impedes all distributers to the speed of the slowest purchaser. This prompts troublesome issues while using this thing in real application as the distributers are deterred endorsers get. Notwithstanding the way that, SonicMQ cases to help scattered trades, it might slump under specific conditions: if a convey call is a bit of a spread trade and the call squares ceaselessly, by then the Distributed Transaction Coordinator) would either break this trade or report a state of unclearness. Dynamically completed, SonicMQ uses pre-getting as usual, achieving deluding execution results a portion of the time. A couple of utilizations that don't think about pre-getting under the spreads can continue running into possibly troublesome issues at the course of action compose, when a couple of messages likely won't be passed on to some other authority since they are lined in a lone recipient (which may have pummeled/hung as a result of potential application dissatisfaction). This results in messages not getting dealt with even while some line beneficiaries are holding on for messages always. SonicMQ server resources (strings) are eaten up just holding on for moving toward data, including unnecessary overhead. In addition, as every additional client interfaces another string ought to be assigned on the server, provoking straightforwardly extending server load. Under such conditions the server over the long haul backs off to unacceptable levels or crashes. In average cases, designating more than 2000 strings is unfeasible on a singular JVM, notwithstanding the way that beyond what many would consider possible move dependent upon the gear and working systems stages used. SonicMQ uses Cloudscape, which is a Java based database the administrators structure as opposed to a report based data store to suffer and store messages. This declines the system speed as archive based data store passes on the messages 10-15 times faster than passing on messages from database."

"The Global Event Detector (GED) [1], made at ITLAB at UT Arlington, is a server subject to the notice/enrollment perspective. All messages passing is done in an intrigue driven mode. That is, no messages are sent to the server with the exception of if there is a client for that message. The server gets an event acknowledgment request from a client application and advances it to the contrasting creator exactly when it registers and the server. Exactly when the event of interest described in the creator application occurs, the producer application 17 instructs the occasion concerning event to the server. The server not simply progresses the occasion of the event to the looking at buyers, yet it is in like manner accountable for distinguishing any composite event reliant on that event. GED uses the ECA (Event-Condition-Action) rule perspective in order to help dynamic limit in an appropriated area. According to the ECA rule perspective, at whatever point the event occurs, the condition described in the standard (for that event) is surveyed and the relating movement is performed if proper. GED does not guarantee movement of events and realizes cautioning/participation perspective. This fine if there ought to emerge an event of event distinguishing proof anyway if the purchaser is essentially fascinated by the period of event and not the event acknowledgment then a convey/purchase in perspective would be even more better. It also does not give any filtering instrument with the objective that the server sends only

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 7, July 2019

the events that are imperative to the customer. There is furthermore no framework that supports need based movement and a way to deal with confirm that the event should remain on the server for a particular term i.e., a chance to live header. GED can be used in a space where in client applications need to recognize events. Nonetheless, in case the applications are as of late captivated that the event has occurred, by then this server may not be that fitting."

"Based on disseminate/purchase in perspective, event organization is one of the organizations from Common Object Request Broker Architecture (CORBA) [8]. It describes three employments (supplier, purchaser, and event channel) [16]. The suppliers and purchasers are decoupled, and clear from each other. Suppliers can push data to clients through the event channel. Also, clients can use event channel to pull the data from suppliers. The event direct fills in as a center individual among clients and suppliers. The event channel interface can be used for including customers, including suppliers, and moreover for destroying the channel. There are four models of part organized exertion in the event organization building: Push Model, Pull Model, Hybrid Push/Pull Model, and Hybrid Pull/Push Model. The Push model empowers suppliers to begin the trading of event data to purchasers. In the draw model, the purchasers request the event data from suppliers through event channel. The Hybrid Push/Pull model empowers the two purchasers and creators to begin the trading of event data. The event channel accept the activity of a uninvolved center individual. The dynamic clients can request data by methods for the event direct in which the dynamic supplier pushes the data. The Hybrid Pull/Push model, instead of the Hybrid Push/Pull model, allows the dynamic event channel to pull the data from suppliers and push them to buyers. Disregarding the way that the symmetry given by the COBRA event organization is all around arranged, it isn't achievable for buyers to purchase in just to events, which are of interest. Each event that is sent from each producer to the event channel is passed on to all the enrolled buyers. This anticipates that customers should filter through event data that isn't of interest. It incorporates additional overhead for the purchasers. This is presumably going to construct the framework use and cause deterred framework traffic. Besides, there is no idea of need based transport and standard explanation based participation for events. More there is in like manner no idea of mishap recovery and bearing the events. organized to propel messages from a line to a different line. If any part bombs in this methodology, the message covering writing computer programs is stuck until the besieged fragment is working precisely as it doesn't support recovery instrument. Logically completed, it requires at any rate one coordinating method on each machine. This isn't reasonable when passing on to various machines. This item moreover requires phenomenal advantages to present or run, as changes must be made to the working system parcel on most supported stages. Despite these, MQ Series does not support enrollment to messages dependent on standard verbalizations and crash recovery. Regardless of the way that the symmetry given by the COBRA event organization is all around arranged, it isn't practical for customers to purchase in just to occasion channel to pull the information from providers and push them to buyers. In spite of the fact that the symmetry given by the COBRA occasion administration is all around planned, it isn't feasible for shoppers to buy in just to occasions, which are of intrigue. Every occasion that is sent from every maker to the occasion channel is conveyed to all the enlisted buyers. This expects buyers to sift through occasion information that isn't of intrigue. It includes extra overhead for the customers. This is probably going to expand the system use and cause obstructed system traffic. Also, there is no idea of need based conveyance and normal articulation based membership for occasions. More there is additionally no thought of accident recuperation and persevering the occasions."

In [2] authors used average residual battery level of the entire network and it was calculated by adding two fields to the RREQ packet header of a on-demand routing algorithm i) average residual battery energy of the nodes on the path ii) number of hops that the RREQ packet has passed through. According to their equation retransmission time is proportional to residual battery energy. Those nodes having more battery energy than the average energy will be selected because its retransmission time will be less. Small hop count is selected at the stage when most of the nodes have same retransmission time. Individual battery power of a node is considered as a metric to prolong the network lifetime in [3]. Authors used an optimization function which considers nature of the packet, size of the packet and distance between the nodes, number of hops and transmission time are also considered for optimization. In [4] initial population for Genetic Algorithm has been computed from the multicast group which has a set of paths from source to destination and the calculated lifetime of each path. Lifetime of the path is used as a fitness function. Fitness function will select the highest chromosomes which is having highest lifetime. Cross over and mutation operators are used to enhance the selection. In [5] authors improved AODV protocol by implementing a balanced energy consumption idea

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 7, July 2019

into route discovery process. RREQ message will be forwarded when the nodes have sufficient amount of energy to transmit the message otherwise message will be dropped. This condition will be checked with threshold value which is dynamically changing. It allows a node with over used battery to refuse to route the traffic in order to prolong the network life. In [6] Authors had modified the route table of AODV adding power factor field. Only active nodes can take part in rout selection and remaining nodes can be idle. The lifetime of a node is calculated and transmitted along with Hello packets. In [7] authors considered the individual battery power of the node and number of hops, as the large number of hops will help in reducing the range of the transmission power. Route discovery has been done in the same way as being done in on-demand routing algorithms. After packet has been reached to the destination, destination will wait for time δt and collects all the packets. After time δt it calls the optimization function to select the path and send RREP. Optimization function uses the individual node's battery energy; if node is having low energy level then optimization function will not use that node.

III. Proposed Algorithm

The scheme proposes an order to clear the priority queue. This count diminishes the time in clearing the need priority queue structure and foresees the resending of a comparative alert to a comparable purchaser. Progressively over it ensures the transport of alerts dependent on their need. This is kept up even by virtue of extension or deletion of new buyers. The estimation achieves this by using the information held in the alert hash table (i.e., prepared once-over table property) and buyer records explained in before territories. Cautions (i.e., Alert Topic) in the lines reliably manual for a purchaser list in the client table data structure in case their subject is a USER or PROFILE and to plenty of records if the fact of the matter is a TAG. The prepared summary table in the preparation that is made when a caution thing is delivered, that is gotten to by the server and not by any client, holds the mapping between the customer list and the last buyer center point in the once-over that has gotten that alert. This information is significant in endeavoring to stop sending the message to the buyers that have successfully gotten the caution, thusly reducing the perfect open door for clearing the client records. Starting at now explained, each purchaser center in the buyer summary has exceptional ID attribute in that once-over and correspondingly every client list similarly has a stand-out ID quality in the customer table data structure. Buyer center point ID trademark assistants in demonstrating the arrival of the purchaser in that summary and customer list ID fills in as a record to the hash table in the caution message which tells that the alert has been sent till this client and ought to be sent to all of the purchasers before this client in the once-over. Since new customers are always added to the beginning of the once-over, the purchasers are reliably in reducing solicitation of their ID trademark as these characteristics show the arrival number in the summary. Thusly, the caution can be sent to all of the buyers with the IDs more imperative than the client ID of that once-over in the hash table present in the alert message, subsequently keeping from resending the message to a comparable customer more than once. At the point when the centers of the summaries are gotten, alerts are sent back using the information in the center. The count takes the line with the most essential need from the data structure and after that explores the line to send each alert in it to the enrolled purchasers. The two data structures that the count compasses may change, either as a result of prepared development or deletion from one of the lines or in view of extension or dropping of purchasers from one of the client records.

The proposed pseudo code is as under :-

```
Consumer Addition Mode
 Set all queues state to not sent.
 Get highest priority queue that has
 not been sent
 while( queue is not empty)
 runSweepingAlgorithm(queue)
 if (Mode is same) {
 Get the next higher priority
 queue that has not been sent
 }
 else if (Mode change) {
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 7, July 2019

```
return to calling function;
}
else {
Set mode to zero since no all
alerts have been sent;
}
```

IV. CONCLUSION

This proposition displays an advising system that grants versatile and reliable structure to move the alert messages based on priority but the modes and crucial level inculcate the sweep mechanism and modeled the queue again on the basis of priority. The proposed scheme also discusses the structure and execution of Alert Server that handles clients and is in like manner responsible for coursing alerts. It explains the arrangement and use of Server that handles clients request or alerts messages. It discusses the issues in an advising area and the arrangement choices made to deal with these issues. It similarly discusses the various alternatives open in addition, reason for the choice of a particular alternative. However, alert server has been completed using convey/purchase in model where clients can purchase in to three subjects TAG, USER and PROFILE. It reinforces typical verbalization based participation as TAG. It furthermore ensures transport of cautions dependent on need and keeps up trade logs for a thorough survey trail of transport of alerts, assertions, and receipts to appropriate objectives. It supports assurance of alerts by logging them on to a record based limit. The clients have the option of sending non-persisting cautions besides. Another logging and recuperation segment has been realized. This is more beneficial than the standard technique for securing the cautions. This is achieved by securing cautions dependent on their needs and securing extra information in the logs. A wide figuring has been expected to clear the data structures and find the clients for the alert using proposed sweeping algorithm.

REFERENCES

[1] Tanpisut, W., Design and Implementation of Event based subscription/notification paradigm for distributed environments. 2001, The University of Texas at Arlington.
[2] Dasari, R., Design and Implementation of a Local Event Detector in Java, in CISE. 1999, Univ. of Florida: Gainesville.
[3] Dasari, R., Events And Rules For JAVA: Design And Implementation Of A Seamless Approach, in Database Systems R&D Center, CIS Department. 1999, University of Florida: Gainesville.
[4] Bloomer, J., Power Programming with RPC. 2000: Reilly.
[5] R., R.B. Making the Most of Middleware. In Data Communications International 24. 1995.
[6] Vondrak, C., Message-Oriented Middleware. 1997.
[7] IONA, The OrbixWeb 3.0 Programmer's Guide. July 1997.
[8] OMG, The Common Object Request Broker: Architecture and Specification Version 2.0.
[9] Microsoft, The Common Object Model Specification Version 0.9. 1995.
[10] Microsoft, Distributed Component Object Model Protocol-DCOM/1.0, draft. 1997.
[11] SunMicrosystems, Java Remote Method Invocation Specification. 2000.
[12] SunMicrosystems, Java Message Service Specification Version 1.0.2b. 2000.
[13] IBM, MQ Series - An Introduction to Messaging and Queuing. 1999.
[14] IBM, MQ Series Queue Manager Clusters. 1999.
[15] FioranoMQ, FioranoMQ and Progress Sonic MQ Comparison. 2001.
[16] Schmidt, D.C. and S. Vinoski, The OMG Events Service. C++ Report. 1997
[17] Douglas, R. and Kelvin J. 2005. Essential SNMP. 2nd Edition. Sebastopol, CA: O'Reilly Media.
[18] Damianos, G. 2001. Mobile Software Agents for Network Monitoring and Performance Management. PhD thesis, University of Essex.
[19] Dev, H., Emery, H., Rustici, S., Brown, M., Wiggin, S., Gray, W. and Scott, P. 1996.
[20] Cabletron Systems Inc. Network management system using model-based intelligence. USA. US5504921.
[21] Ethan, G. 2011. Nagios Core "Ubuntu Quickstart Installation Guides". Available: http://nagios.sourceforge.net/docs/nagioscore/3/en/quickstart-ubuntu.html. Accessed 29 September 2011.
[22] Haojin, W. 1999. Telecommunications Network Management. NY: The McGrawHill.
[23] James, T. 2006. Pro Nagios 2.0 Administrations. Berkeley, CA: Apress.
[24] Jianguo, D. 2010. Advances in Network Management. Boca Raton, FL: Auerbach.
[25] Martin, P. 2003. Data Networks, IP and the Internet: Protocols, Design and Operation. Chicester, WS: John Willy & Sons.
[26] Michael, T. 2009. Monitoring Network and Service Availability with Open-Source Software. Winner of the LITA/Ex Libris Writing Award, 2009. Available: http://www.ala.org/ala/mgrps/divs/lita/publications/ital/29/1/silver.pdf. Accessed 27 July 2011.
[27] Munin Wiki 2011. Available: http://munin-monitoring.org. Accessed 29 September 2011.
[28] Nagios Community 2011. Available: http://nagios.sourceforge.net/docs/3_0/about.html#whatis. Accessed 29 September 2011.