2002-09

Quality of Service Behavioral Model from Event Trace Analysis

Luqi

Quality of Service Behavioral Model from Event Trace Analysis, with J. Drummond, W. Kemple, M. Auguston, N. Chaki. Proc. of the 7th international Command and Control Research and

Cover page for the 2002 7th International Command and Control Research and Technology Symposium
16-20 September 2002, Quebec City, Quebec

Possible focus topic:

C2 Experimentation

Paper Title:

Quality of Service Behavioral Model From Event Trace Analysis

Authors:

John Drummond*
SPAWARSYSCEN
San Diego Ca 92152-5001
(619)553-4131, **drummond@spawar.navy.mil**

Valdis Berzins, Luqi, William Kemple, Mikhail Auguston, Nabendu Chaki
Naval Postgraduate School
555 Dyer Road, Monterey, CA 93943-5118 USA
(831)656-2610, **berzins@cs.nps.navy.mil**
(831)656-2735 **luqi@cs.nps.navy.mil**,
(831)656-2249, **kemple@cs.nps.navy.mil**
(831)656-2509, **auguston@cs.nps.navy.mil**
(831)656-2509, **nchaki@cs.nps.navy.mil**

* **This work sponsored by the Defense Advanced Research Projects Agency, Information Technology Office (DARPA-ITO)**

Quality of Service Behavioral Model From Event Trace Analysis

John Drummond[*]
SPAWARSYSCEN
San Diego Ca 92152-5001
(619)553-4131, drummond@spawar.navy.mil

Valdis Berzins, Luqi, William Kemple, Mikhail Auguston, Nabendu Chaki
Naval Postgraduate School
555 Dyer Road, Monterey, CA 93943-5118 USA
(831)656-2610, berzins@cs.nps.navy.mil
(831)656-2735 luqi@cs.nps.navy.mil,
(831)656-2249, kemple@cs.nps.navy.mil
(831)656-2509, auguston@cs.nps.navy.mil
(831)656-2509, nchaki@cs.nps.navy.mil

Abstract

The distributed command & control environment includes limited computer resources and numerous mission critical applications competing for these scarce resources. Additionally the stringent constraints and considerable complexity of distributed command & control systems can create a condition that places extreme demands upon the allocated resources and invites a potential for program errors. Consistent quality of service distribution can be a critical element in ensuring effective overall program completion while avoiding potential errors and process failures. The potential for errors and process failures can be understood and addressed by performing a practical analysis of the resource deployment procedures utilized within this environment. However, analyzing resource-based quality of service within a distributed command & control environment is a demanding endeavor. This difficult task can be simplified by directly examining specific quality of service actions that take place during program execution. Therefore, to

[*] **This work sponsored by the Defense Advanced Research Projects Agency, Information Technology Office (DARPA-ITO)**

pragmatically isolate these actions and develop a practical quality of service behavioral model, the research discussed in this paper has implemented an event trace approach to examine the exact quality of service execution path during program operation.

Introduction

The command & control environment is especially complex and may certainly exhibit dynamically changing attributes during its operation. The processing of command & control elements can exhibit perplexing difficulties such as abrupt mission changes, and dynamic tactical surprises[Harn 99]. Many critical applications within this environment could benefit from distributing the processing load. Distribution of the processing load does however also increase the overall complexity of the environment. Despite the expanded complexity, the augmentation to command & control environments of distributed processing is desired. The distributed processing environment can provide added benefits over the non-distributed approach due to the capacity for improvement in program accessibility, overall performance, additional sharing of limited resources, and the increased fault tolerance capabilities. However, this distributed command & control environment does present an expanded assemblage of requirements and constraints for effective computer resource control. The efficient allocation of computer resources can be considered a major element of these requisites. The direct implementation of resource control features into the distributed command & control infrastructure can be extremely advantageous for software programs that contain mission critical requirements. Implementing quality of service features into the distributed command & control infrastructure is not a trivial task. Additionally, subsequent to implementing the quality of service features, an examination must be performed upon the effectiveness of the implementation.

To properly ascertain that the essential quality of service based system resources are being reasonably utilized and efficiently shared among these programs some evaluation of the resource deployment method should be conducted. However, current analysis techniques for evaluation of resource deployment and control are somewhat lacking in that there is no exacting method to focus exclusively upon specific quality of service actions that take place during actual program execution. Therefore, the analysis of proper employment and dispersion of available resources is the focus of this research in the area of distributed command & control processing. This direct analysis can be carried out though the use of quality of service based behavioral models. The development characteristics of the behavior model are described in the next section.

Approach

The foundation for the approach to developing the quality of service behavioral model considers a centralized resource provisioning mechanism for control of all computer resources that can be found within the end-to-end pathway. Typically communication based quality of service analysis approaches have focused upon the network resource provisioning implementations that utilize a decentralized resource management technique to disperse the required communication bandwidth resources at numerous locations (e.g. ATM switches, etc). The scope of the research being

pursued in this paper expands this quality of service analysis of resource deployment to include computer resources that can consist of CPU, Network, Disk, I/O, and Memory. These resources can be considered critical elements within a true end-to-end distributed environment and have a direct bearing upon any quality of service capabilities.

This focus of efficient quality of service within the total end-to-end pathway is a much needed element for current DoD systems as stated by the Defense Advanced Research Projects Agency Quorum program manager [Koob 99] “While emerging network-level QoS mechanisms (such as RSVP) are an essential enabling technology for Quorum, they are insufficient in that they are limited to communications QoS. Quorum defines "end-to-end" as being the quality-of-service seen by the application, which calls for coordinated QoS management across middleware, operating systems, and networks.”

For the purpose of this research investigation the mechanism for controlling and coordinating these critical quality of service resources will be centralized within a singular system as can be found within the Linux/RK resource kernel [Rajkumar 98]. This research does not examine the multiple controller communication/network mechanism mentioned earlier. This investigation is accomplished by an in-depth look at various quality of service and resource deployment characterizations as well as the application of high level modeling and quality of service analysis. The detailed analysis is attained through the utilization of the SPAWAR System Center DARPA Quorum Integration Test & Exploitation project (Quite) testbed environment located at the SPAWAR System Center. Other specific logical conditions and constraints for this work include distributed systems, heterogeneous environment, multiple diverse quality of service levels essential for program execution (i.e. application requirements vary high/low needs), and available resources can include network bandwidth, CPU, memory, etc.

To achieve a precise analysis of quality of service procedures an approach has been implemented to examine the exact quality of service execution path during program operation. The evaluation approach utilized in this research is based upon an event trace concept employed by [Auguston 00] originally as an analysis tool for focusing upon correctness in C language programs. This event trace concept discusses the idea that testing and debugging are mostly concerned with the program run-time behavior, and states that developing a precise model of program behavior becomes the first step towards any dynamic analysis.

The method of performing the quality of service event trace analysis begins with an event trace of a targeted application. This event trace is utilized as the basis for developing the quality of service behavior that characterizes the targeted program. The quality of service based event trace approach allows for a detailed quality of service parameter examination. This event trace is utilized as the tool for collecting the predefined quality of service metrics and allows for in-depth analysis based upon these previously developed metrics. The specific events to be isolated within these parameters for this research are based upon actions that may have temporal properties (e.g. Event Start, Event Stop) or simply be atomic in nature (e.g. Initialization). This follows the event

trace work of [Auguston 98]: “Every event defines a time interval which has a beginning and end. For atomic events, the beginning and end points of the time interval will be the same.”

The procedures for performing the quality of service event trace include:

- Develop operative models of execution pathways (quality of service & resource control specific statement execution) within the target application based upon identifiable details such as resources requested, resources utilized, resources available, etc.

- Initiate the development of a working model of program behavior based upon quality of service factors. This is accomplished by producing abstractions of events that are fundamental to specific quality of service actions performed during program execution, which include:

    - Quality of service request statement execution that requests resource reservation.
    - Procedure execution that focuses upon the evaluation and negotiation of available resources to be applied to the originating resource request.
    - Software statement execution of procedures for proper utilization of the assigned resources.
    - Execution of statements responsible for the detection of any resource needs change within the application software.
    - Execution of procedures focusing upon the re-negotiation based on increase or decrease of available and previously assigned resources.
    - Execution of reallocation statements for specific resources by the resource controller.
    - Sending and receiving of quality of service related messages by both the application and resource controller software.

- Identify quality of service specific application program points that directly relate to appropriate resource deployment as illustrated in Figure 1. Such elements have direct consequences upon quality of service behavior and include:

    - QoS specific message passing
    - Application QoS violations
    - QoS negotiations
    - QoS resources and control of resources
    - QoS re-negotiations
    - QoS level

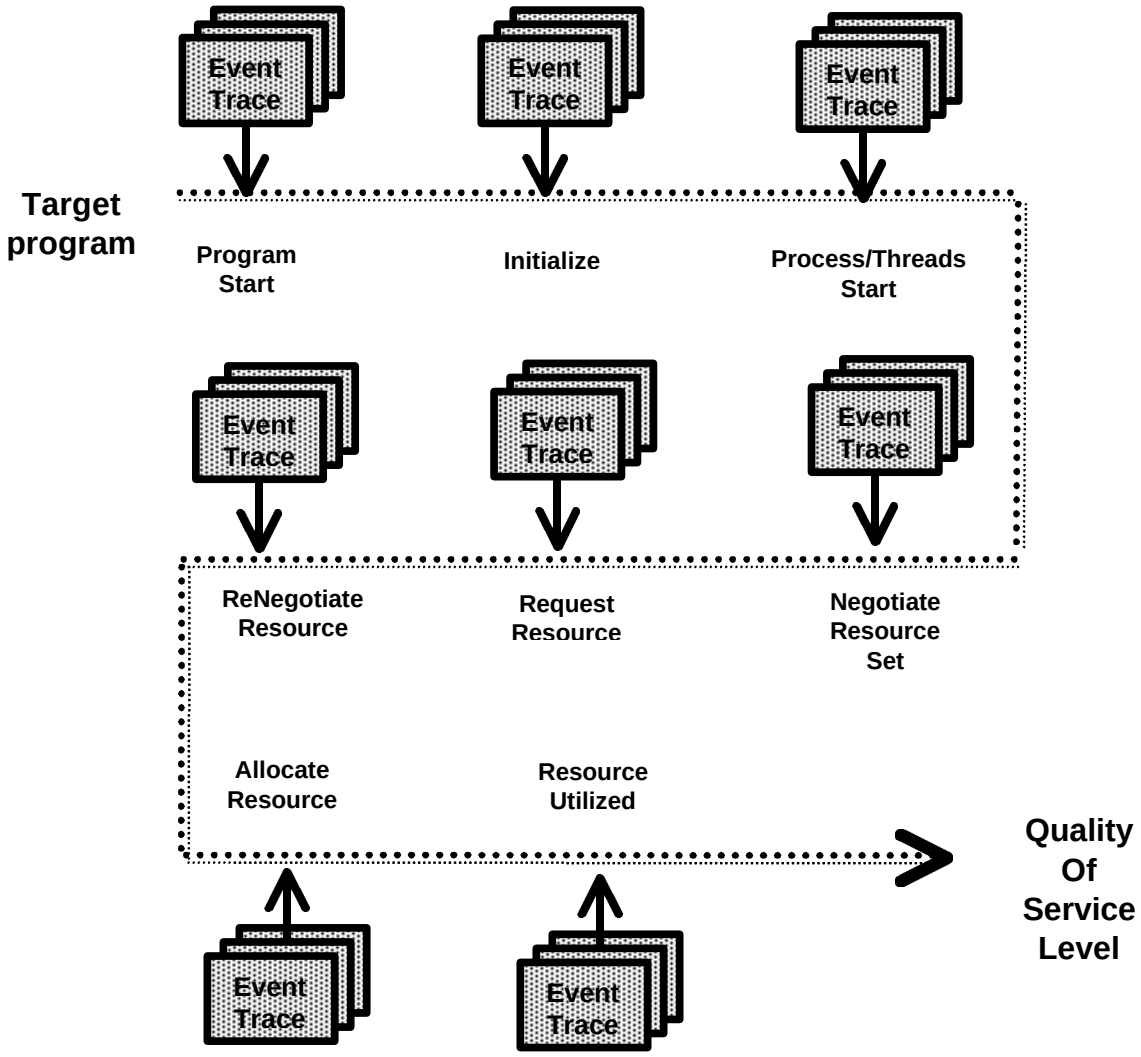

Figure 1. Quality of Service Program Points.

- Instrument the targeted program based upon these previously identified specific quality of service program points. This direct code instrumentation will allow for effective event trace recording at the precise location of the quality of service actions of interest.

At this point it is necessary to further expand upon the events of interest for quality of service analysis and the development of the behavior model. An event is a detectable action that influences the overall achievement of the desired quality of service level. The event is the smallest element of the quality of service behavior model. The discovery of this action is noted by the embedded instrumentation within the targeted program sources at the pre-defined program points as previously shown in Figure 1. The event attributes describe the event and include the process

or thread within which this event has occurred, and a boolean attribute denoting the associated quality of service action of success/failure.

The event model is constructed from a specific quality of service based action and all the attributes relevant to this action. The event model is applied to the event trace for the purpose of constructing the quality of service behavior. There are eight events of interest and their respective attributes that form the composition of the behavioral model.

The resource request event is critical to the development of the behavioral model because it represents the applications mechanism for acquiring the proper resources for successful program execution. Within the quality of service behavioral model every resource request event and subsequent failure/success attribute is indicative of the applications behavior. The resource request event model is composed of the action of requesting resources and the set of event attributes that include: depth level = DP, process type = TP, location = LC, path = PA, resource type = RT. The request resource event RQ = {DP, TP, LC, PA, RT}, this event model is illustrated in the next figure.

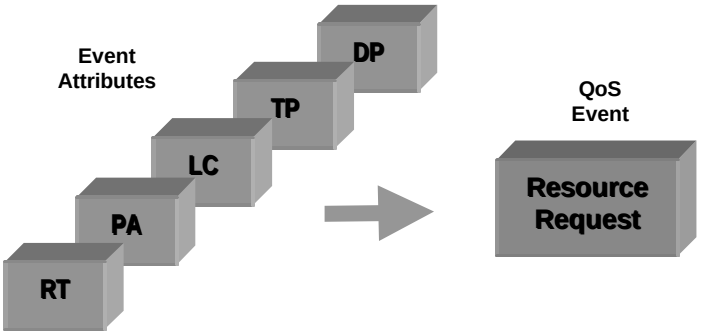

Figure 2. Resource Request Event Model.

The quality of service violation event is an important element of the behavioral model as it is representative of a quality of service fault. This failure event has a causal relation to the preceding quality of service associated attempt actions that include resource negotiation, resource request, resource re-negotiation, and resource assign. Within the composition of the quality of service behavioral model failure is indicative of the applications behavior. The quality of service violation event model consists of the resource request failure, or resource negotiation failure, or resource re-negotiation failure, or resource assigned failure actions and the set of event attributes: depth level = DP, process type = TP, location = LC, path = PA, resource type = RT. The quality of service violation event QV = {RT, DP, TP, LC, PA} and this event model is illustrated in the next figure.

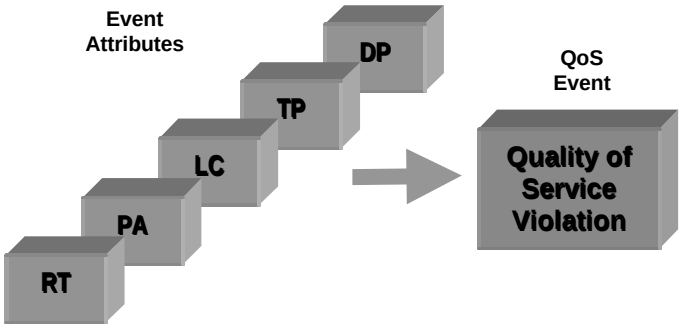

Figure 3. Quality Of Service Violation Event Model.

The quality of service level event supports the behavior model as it represents the action of the resource controller appropriating the requested resource. Within the quality of service behavioral model the appropriation of resources is a significant action in the attainment of proper quality of service and consequently characterizing the applications behavior. The quality of service level event model is composed of the action of resource reserve creation success action through the resource controller and the set of event attributes that include: depth level = DP, process type = TP, location = LC, path = PA, resource type = RT, resource size = SZ, resource period = RP, resource deadline = RD, and resource used = RU. The quality of service event QL= {DP, TP, LC, PA, RT, SZ, RP, RD, RU}, this event model is illustrated in the next figure.

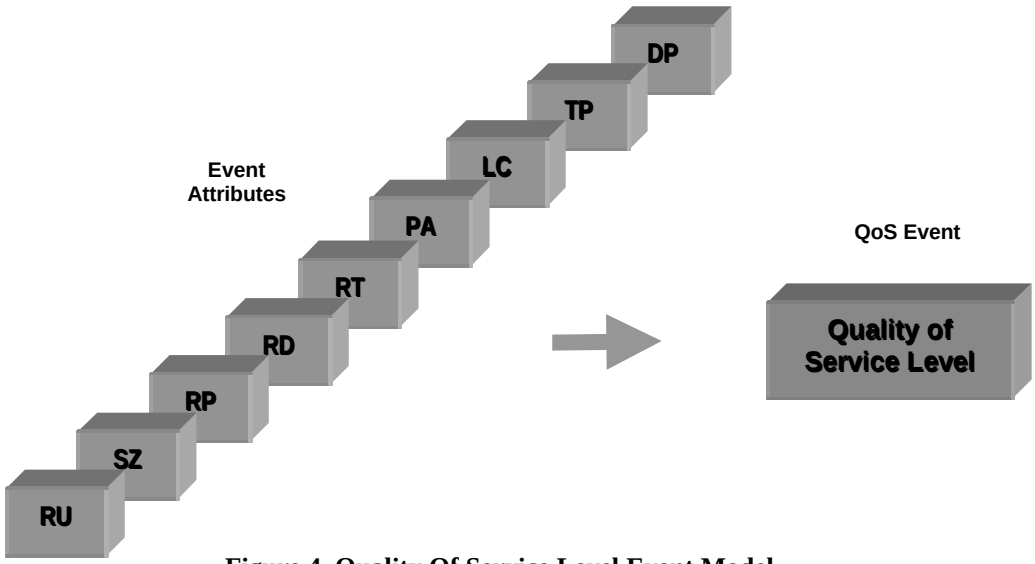

Figure 4. Quality Of Service Level Event Model.

The resource negotiation event is an important part of the behavior model because it represents the transaction of establishing a resource set with the resource controller. This event is significant in the acquisition of resources and therefore an important in the development of the applications quality of service behavior. The resource negotiation event model is comprised of the action of setting up this resource set and the event attributes that include: depth level = DP, process type = TP, location = LC, path = PA, resource type = RT. The resource negotiation event RN= {DP, TP, LC, PA, RT}, this event model is illustrated in the next figure.

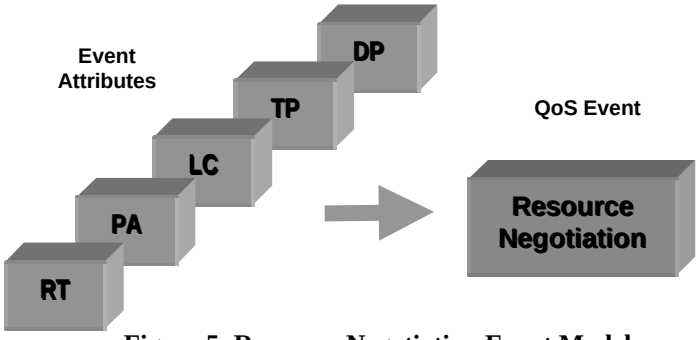

Figure 5. Resource Negotiation Event Model.

The system reservation event is a component in the behavior model because it represents the action by the system, and other applications not currently being targeted by the event trace, of requesting resources from the resource controller. When the focus is directed only at the target program for evaluation the system resource event simply represents a competing load application. This event is critical to the quality of service behavior model because it enables an evaluation of the target program under resource competition load. The system reservation event model consist of this resource reservation action by these competing users through the resource controller and the set of event attributes that include: process type = TP, location = LC, path = PA, resource type = RT, resource size = SZ, resource period = RP, resource deadline = RD, and resource used = RU. The system reservation event SR = {TP, LC, PA, RT, SZ, RP, RD, RU}, this event model is illustrated below.

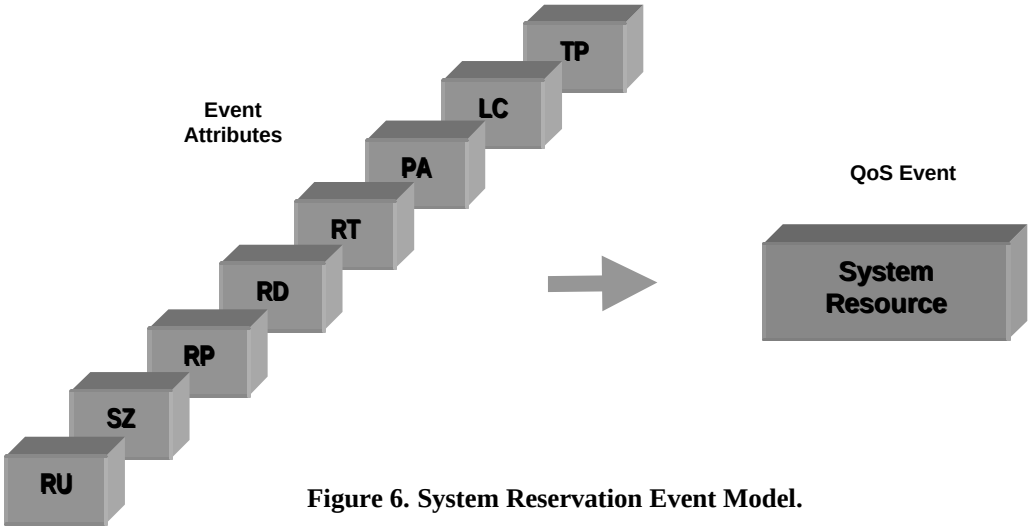

Figure 6. System Reservation Event Model.

The resource assignment event is a critical element in the quality of service behavior model because it describes the action of assigning resources by the resource controller to the requesting thread/process. The resource assignment event model is composed of the action of attaching the resource set to the specific process/thread through the resource controller and the set of event attributes that include: depth level = DP, process type = TP, location = LC, path = PA, resource

type = RT. The resource assignment event SR= {DP, TP, LC, PA, RT}, this event model is illustrated in the figure below.

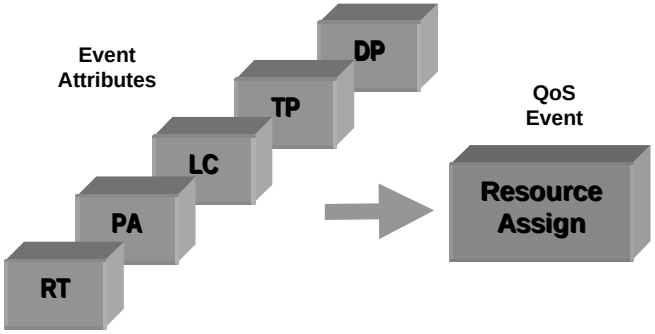

Figure 7. Resource Assign Event Model.

The path length event is part of the quality of service behavior model because it represents the action of traversing the quality of service path to a succeeding program point level within the event trace. The path length event model consists of the action of proceeding through program points and the set of event attributes that include: depth level = DP, process type = TP, location = LC, path = PA. The path length event PL = {DP, TP, LC, PA} and this event model is illustrated below.

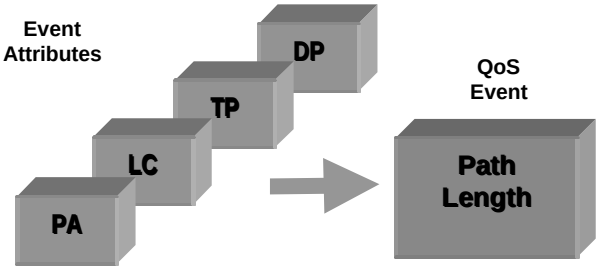

Figure 8. Path Length Event Model.

The resource re-negotiation event is critical to the quality of service behavior model because it is representative of the process/thread actions to correct a preceding quality of service violation. The resource re-negotiation event model is comprised of the action of re-negotiating the amount of resource requested through the resource controller and the set of event attributes that include: depth level = DP, process type = TP, location = LC, path = PA, resource type = RT, resource size = SZ, resource period = RP, and resource deadline = RD. The resource re-negotiation event RR = {DP, TP, LC, PA, RT, SZ, RP, RD}, this event model is illustrated in the next figure.

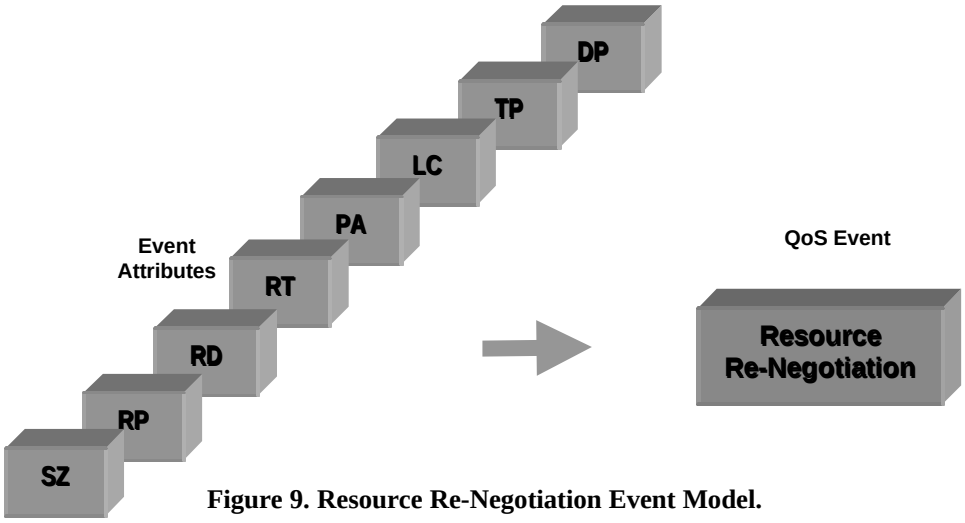

Figure 9. Resource Re-Negotiation Event Model.

These specific event trace quality of service actions which comprise the event models are informally structured with no imposed overall ordering structure as they occur asynchronously. However, there is a partial ordering within a specific thread/process execution as denoted by the thread/process depth level attribute, and a causal ordering between request events (resource negotiation, request resource, resource re-negotiation, resource assign) and fault event (quality of service violation). After the event occurs the event trace notes the specific attributes of the quality of service action such as type (quality of service resource), level (path depth), path (aggregate progression of the event trace), ptype (process or thread), and loc (within the process/thread). This data is noted and a boolean evaluation process determines its success/failure attribute.

The behavioral model is composed of resource request events ‘RQ, resource negotiation events ‘RN, resource assign events ‘RA, quality of service events ‘QV, resource re-negotiation events ‘RR, quality of service level events ‘QL, system reservation events ‘SR, and path length events ‘PL. Potential failure behaviors are comprised of the following sets of events: {RN, QV}, {RQ, QV}, {RQ, QV, RR, RR, RR, QV}, {RR, QV}, and {RA, QV}. Typical success behaviors are composed of sets of events that include: {RN}, {RQ, QL, RA}, {RR, QL, RA}, and {RA}. An example of a quality of service behavior model that characterizes quality of service failure is illustrated in the next figure.

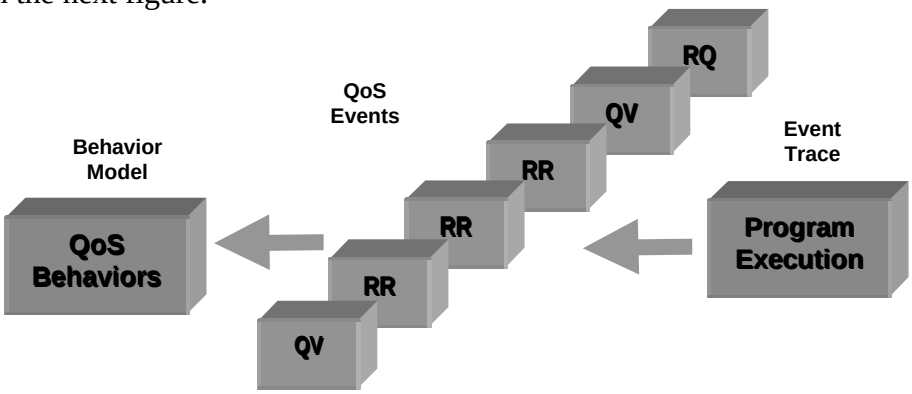

Figure 10. Quality Of Service Behavior Model.

The behavioral model can be utilized to isolate specific quality of service behaviors. For example in the failure occurrence illustrated in above with the events {RQ, QV, RR, RR, RR, QV} the program point of failure can be isolated through examination of the event trace results. This search can be achieved through a retrace of the execution path to the specified depth level of the distinct (named) thread/process, and by examination of the event attributes associated with the failure event quality of service violation QV = {DP, TP, LC, PA, RT}.

The quality of service event trace on the targeted application collects these events and based upon this information the quality of service behavior can be constructed. The behavior model is partially ordered set of event types (e.g. resource request) and event attributes (success/fail boolean). The quality of service metrics of the event trace are calculated based on quality of service actions. These calculations include elements such as the number of resource re-negotiation events that have occurred, and the number of application quality of service violations. The results of these event trace metrics include the lists of specific processes/threads identities and their resource specific events. This information provides the necessary data to construct the quality of service behavior.

Applying The Event Trace

This discussion has thusfar focused upon the composition of the quality of service event trace. The model of this generalized (general case) approach can be applied to the specific case implementation of a pre-selected application. The employment of the quality of service event trace analysis is based upon the event models applied to a target application program that has been instrumented for accurate feedback. This analysis is then utilized to develop an overall quality of service behavior for characterization of the command & control systems application that can be applied to mitigation of discovered quality of service related efficiency problems. Through direct examination of the event trace results, specific potential failure regions (QV events) can be isolated to specific path locations and thread/process depth levels that denote the distinct program point within the targeted application. The potential error region can then be adjusted to improve the quality of service level achievement probability.

The selection of the target program/environment for this work has been subjectively inspired through the research of the DARPA Quorum program. The specific emphasis of the DARPA-ITO sponsored Quorum program is the development of computing environments with quality of service attributes, controls, and guarantees on local to global scales[Koob 99]. This Quorum program is a multi-million dollar collaboration of fifty top research groups representing universities, industries, and SPAWAR System Center. The quality of service event trace research reported in this paper leverages from this DARPA project focus and this association has provided a high degree of valuable input.

A failure detection program was isolated as a candidate target application based upon the logical conditions and constraints discussed earlier. This fast failure detection program has as its primary objective the efficient and prompt detection of node failures within group communication

software as utilized within distributed mission critical systems of the AEGIS environment. The fast failure detection program was designed and developed under the DARPA-ITO Quorum Integration, Testbed and Exploitation (Quite) project efforts.

As noted in [Drummond 02] this program can be set up to take advantage of a resource management system based upon quality of service procedures or operate as a simple non-quality of service application “The Fast Failure Detector can be built and executed on its own or it can be executed while taking advantage of facilities like Linux/RK [Oikawa 98] and Ensemble[Birman 00] group communication.” For the purpose of this event trace analysis research the fast failure detection program has been implemented using both the Linux/RK kernel and the Ensemble group communication software.

The specific area of concentration within this targeted environment is based upon the following problem domain characteristics: distributed computing environment, multiple heterogeneous systems, network medium connections, software applications with specific requirements (quality of service resource needs), centralized resource control software (with quality of service awareness), metrics data gathering instrumentation software. This specifically isolated computing environment can be readily encountered within the AEGIS system, as well as in many other DOD and commercial systems. AEGIS is a combat system architecture that contains a computer-based command & decision component. The core element of the AEGIS architecture provides simultaneous operations capability. These operations include measures against multi-mission threats including anti-air, anti-surface, and anti-submarine. The problem space of this domain requires specific levels of performance to operate correctly.

Explicit quality of service program points that directly relate to resource utilization have been isolated within the target program. Based upon these program points the target application has been instrumented. This has been accomplished by direct source code instrumentation that allows for effective event trace recording at the actual location where the quality of service specific actions take place. The analysis results can be utilized to develop an overall characterization of the fast failure detection program for any mitigation of discovered quality of service related efficiency problems. The specific event trace analysis has examined the quality of service events and related quality of service characteristics of this fast failure detection program within a distributed environment.

For this application of the quality of service event trace analysis approach to the specific case of the selected targeted application the following distinct events and attributes have been recorded within each quality of service event trace execution. These events, their actions and attributes follow the event models described earlier.

EVENT	ACTION	ATTRIBUTE
RES_NEG	Resource Set Negotiation	RES_TYP, PATH, LOC, LEVEL, PTYPE
REQ_RES	Resource Request	RES_TYP, PATH, LOC, LEVEL, PTYPE
RES_ASG	Resource Assignment	RES_TYP, PATH, LOC, LEVEL, PTYPE
PATH_LN	Quality of Service Program Point Traversal	PATH, LOC, LEVEL, PTYPE
QOS_VIO	Quality of Service Violation	RES_TYP, PATH, LOC, LEVEL, PTYPE
RES_RNG	Resource Re-Negotiation	RES_TYP, PATH, LOC, LEVEL, PTYPE, SIZE, PERIOD, DEADLINE.
QOS_LEV	Resource Appropriation	RES_TYP, PATH, LOC, LEVEL, PTYPE, SIZE, PERIOD, DEADLINE, USED
SYS_RES	System/Application Resource Reservation	RES_TYP, PATH, LOC, PTYPE, SIZE, PERIOD, DEADLINE, USED

Table 1. Quality of Service Events.

These specific event types (shown in table 1) that have been included within the quality of service event trace of the target failure detection program were chosen because they represent distinct actions during program execution that have a direct influence resource utilization. These event types include attributes that are closely associated with and help describe these actions. The quality of service events occur asynchronously within the quality of service event trace and are informally structured with no overall strictly imposed ordering, however there is a partial ordering within each executing thread.

The RES_TYP notation represents the event trace attribute that denotes the type of resource (RT) reservation that is requested. This event attribute is not utilized during the analysis of this target failure detection program, as the sole resource that has been reserved by this program is the CPU resource. When it is used, the other possible resources that this attribute can represent include Disk, Network, and Memory. The PATH attribute references the total event trace quality of service path (PA) length that has been recorded during the application execution. This path element represents a simple integer value that is dynamically updated and recorded as the event trace proceeds. This integer value indicates the aggregate progression of the event trace. The quality of service event trace attribute labeled LEVEL is similar to the Path element. However, this element reflects the specific process or thread execution path depth (DP) as it proceeds though the operations necessary to attain a specific level of quality of service. This element is also a simple integer that is dynamically updated and recorded as the process or thread executes. The LOC attribute references the specific processing location (LC) that the quality of service event trace is recording from within the specific process or thread. This attribute is a simple char type and includes FFDMAIN, FFDINIT, KSYSTEM, THREAD1, THREAD2, and THREAD3. The next attribute in the quality of service event trace output data is titled PTYPE. This attribute indicates the specific task type (TP) that has been recorded. Two of the possible task types

include process, and thread which are directly related to the failure detection application. The SIZE attribute reflects the resource size (SZ) being requested. The SIZE attribute is measured in resource units. The PERIOD attribute indicates the period (RP) that the resource is utilized within. The DEADLINE attribute relates to specific information (RD) utilized by Deadline Monotonic and Earliest deadline First scheduling policies. The USED attribute reflects the event of resources being allocated (RU). The TOTAL element indicates the additive figure of all resources that have been allocated. The AVAL element is representative of the total resource available as indicated by the resource kernel. This element is also measured in resource units.

For this case study additional fabricated competing application tasks RS1, RS2, and system processes such as DISK were executed during the event trace. The competing application is a simple CPU resource load program that requests large amounts (400.0 & 200.0 units) of this resource. Its sole purpose is to present the target program with a resource competitor for evaluation under load. The system process is produced by the resource kernel for continuous disk access and requests a nominal amount of CPU resource (0.299 units). The resource kernel also indicates a setback of minimum 90 resource units that cannot be allocated.

Conclusion

The concluding results of this examination have produced high-level quality of service behavior representations of the fast failure detection program. This quality of service event trace analysis has shown the capacity to specifically reveal various failure points, potential resource re-negotiation inefficiencies, and excessive quality of service paths. All of these elements have a direct bearing upon the quality of service based resource deployment efficiency for the distributed command & control fast failure detection application program and environment.

The events of interest from the resulting case study examination demonstrate a typical success behavior composed of {RN}. Also discovered were potential failure behaviors illustrated as a progressive pattern of potential for failure that concludes with a quality of service violation and final failure. This progression can be see in patterns composed of {RQ, QV, RR, QL, RA}, {RQ, QV, RR, RR, QL, RA}, {RQ, QV, RR, RR, RR, QV}. For this final failure the program point of failure can be isolated through examination of the event trace results and the event attributes as shown in Table 2 below. This examination includes a retrace of the execution path to the specified depth level of the distinct (named) thread/process, and by isolation of the event attributes associated with the failure event quality of service violation = {DP, TP, LC, PA, RT}. Where DP= 7-11, TP=THREAD, LC=THREAD2, PA=34-38, RT=CPU.

ETTYPE	PATH	LEVEL	LOC	PTYPE	SIZE	PERIOD	DEADLINE	TOTAL	AVAIL
QOS_VIO	34	7	THREAD2	THREAD				610.299	90
RES_RNG	35	8	THREAD2	THREAD	6.000	50.000	50.000	610.299	90
RES_RNG	36	9	THREAD2	THREAD	4.000	25.000	25.000	610.299	90
RES_RNG	37	10	THREAD2	THREAD	2.000	12.000	12.000	610.299	90
QOS_VIO	38	11	THREAD2	THREAD				610.299	90

Table 2. Event Trace Results.

It is interesting to note in performance analysis that this complete denial of resources could have a catastrophic consequence for the requesting thread or process. This result could also translate into an uncertain outcome for the total program execution resulting in total program failure. This instance of a potential for total program failure was evident in the case study of the fast failure detection program. During the quality of service event trace analysis execution that included competition for resources that produced the data found in Table 2, the fast failure detection program exhibited a total collapse of the THREAD2 task. This failure of the specific thread to reserve necessary resources from the resource kernel in turn resulted in an abort of the program.

Thus far this work has specifically been directed towards the area of developing quality of service behavior models for targeted distributed command & control programs. This utilization of the quality of service event trace approach to behavioral modeling can be further expanded for inclusion into a development/analysis framework.

References

[Auguston 00] Auguston, M., *Assertion Checker For The C Programming Language Based On Computations Over Event Traces*, Fourth International Workshop on Automated Debugging, AADEBUG2000, Munich, Germany, August 2000.

[Auguston 98] Auguston, M., *Building Program Behavior Models*, Proceedings of the European Conference on Artificial Intelligence ECAI-98, Workshop on Spatial and Temporal Reasoning, Brighton, England, August 23-28, 1998.

[Birman 00] Birman, K., et al., "*The Horus and Ensemble Projects: Accomplishments and Limitations*", Proceedings of the DARPA Information Survivability Conference & Exposition (DISCEX '00), Hilton Head, South Carolina, January 2000.

[Drummond 02] Drummond, J., Wells, D., Rahman, M., *Detecting Failure Within Distributed Environments*, SPAWAR Technical Paper TR1884, Space and Naval Warfare Systems Center, San Diego, Ca. 2002.

[Harn 99] Harn, M. Berzins, V. Luqi, Kemple, W., *Evolution of C4I Systems*, Command & Control Research and Technology Symposium, 1999.

[Koob, 1999], Koob, G., *Background for DARPA-ITO Quorum Mission Statement*, Defense Advanced Research Projects Agency Information Technology Office, 1999.

[Oikawa 98] Oikawa, S., Rajkumar, R., *Linux/RK: A Portable Resource Kernel in Linux*, IEEE Real-Time Systems Symposium Work-In-Progress, Madrid, December 1998.