

Diagnosis of End-to-end Paths in Dynamic, Distributed, Real-time Systems

Charles D. Cavanaugh
 Computer Science Department
 University of Missouri-Rolla
 Rolla, MO 65409
charlesc@umr.edu

Lonnie R. Welch
 School of Electrical Engineering and Computer Science
 Ohio University
 Athens, OH 45701
welch@ohio.edu

Behrooz A. Shirazi
 Department of Computer Science and Engineering
 The University of Texas at Arlington
 Arlington, TX 76019
shirazi@cse.uta.edu

Abstract

Dynamic, distributed, real-time systems must control a changing environment in a timely manner with a workload that is difficult to predict at design time. A quality of service (QoS) manager enables such systems to do so by monitoring the timing and intelligently identifying components that need more or fewer resources. In particular, the distributed system consists of collections of time-constrained and precedence-constrained applications that communicate and work together to form a pipeline, or path, which is divided into subpaths. The QoS manager monitors these subpaths in order to diagnose paths of dynamic, distributed, real-time systems.

1. Introduction

Today's safety-critical and mission-critical software systems, from air-traffic control to battleship defense systems, operate on tight deadlines [5][16]. The input data comes from sensors that monitor the real-world environment, such as the radars that monitor the airspace around a busy airport or a battleship, so the volume of data (the workload) changes dynamically depending on the conditions. A quality of service (QoS) manager maintains QoS in such systems by monitoring the timing and intelligently identifying components of the software that need more or fewer resources.

The distributed application consists of collections of time-constrained and precedence-constrained applications that communicate and

work together to form a pipeline, or path [16][17]. The path further consists of subpaths, which are components of the path, either programs or logical communication links between programs. The QoS manager monitors and analyzes these subpaths in order to diagnose paths of dynamic, distributed, real-time systems. For example, figure 1 shows a system that processes raw data from air-traffic control radar and alerts the air-traffic controller if aircraft come dangerously close to one another. The subpaths must work together to process and move the data to the display quickly before the next radar sweep, or the display will lag behind the radar. Path-based computing is the major approach that the DeSiDeRaTa (dynamic, scalable, dependable, real-time) project uses [16][17][14][5][6][7][8]. Figure 2 shows DeSiDeRaTa's set of monitoring, load balancing, and support programs comprising the middleware architecture.

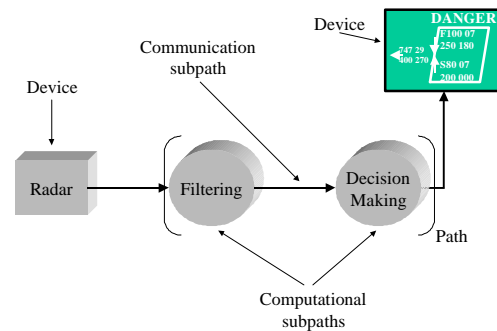


Figure 1. Radar data processing path and subpaths.

In this paper, we propose a new approach to QoS diagnosis and action selection and experimentally show its effectiveness in identifying the cause of and resolving poor QoS. Section 2 presents the key concepts of our solution and the algorithm for diagnosis and action selection, and section 3 shows experimental results using the solution in the actual QoS manager. Section 4 summarizes previous work in QoS management, and section 5 concludes the paper and indicates future work.

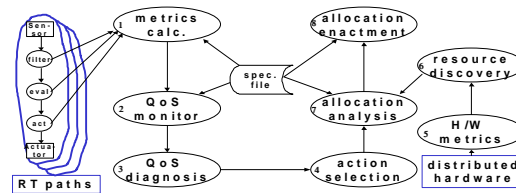


Figure 2. QoS management middleware architecture.

2. Solution

In our solution to diagnosing paths, we isolate the unhealthy subpaths and determine what caused the subpaths to become unhealthy. If a subpath has the maximum excess latency of all subpaths, then that subpath is unhealthy. There are two situations that may cause a subpath to be unhealthy: contention and workload. In the case of contention, another process or data stream uses the same resource as the subpath, causing the subpath to experience excess latency as a result of processor or network queuing delay. There is some room for the execution time to increase (that is, there is some queuing delay time contributing to the excess latency, as shown in figure 3). There are plenty of resources available to ensure adequate throughput, so the path is not saturating its resources. In this case, the best solution is to get away from the contended resource. One of two actions should be taken: reduce the contention (and thus the queuing delay), or circumvent the contended resource by removing the need for it (as in the case of combining the endpoints of a communication subpath). For processes, moving to a host whose load is lighter than the current is best. For communication subpaths, either using a more lightly loaded network or nullifying the communication time by combining computational subpaths onto the same host is the best solution.

In the case of workload, the sheer amount of data being sent or processed using a resource does not allow the data to go through the resource in the required amount of time. At this point, the required subpath latency is *less* than the profiled execution/communication time for a given workload, so the profiled execution/communication time *must* be reduced in some way. In the example shown in figure 4, there is very little total queuing delay, but the path is violating the requirement because it is saturating the resources. In this case, the best solution is to reduce the sheer amount of data that goes through the subpath or to increase the throughput. This can be done by reducing the workload or increasing the throughput (by using hardware that can handle the workload more quickly, be it network or processor load). For computational subpaths, copying the subpath onto a suitable host or moving the subpath to a host with a higher *SPEC* rating than the current host's is a solution. For communication subpaths, the best solution is to either use a higher-speed network or copy one endpoint of the computational subpaths. (While the latter

option would reduce the size of the data stream traveling over the link, the option would have a net effect of zero in Ethernet, since nodes in the network share the physical link.) If neither option is available, combining the endpoints will reduce the communication latency by eliminating the need for the communication network resources. The diagnosis algorithm distinguishes workload-related increased latency from contention-related increased latency through the use of queuing delay (the difference between profiled time and actual time) and slack (the relationship between profiled time and the requirement).

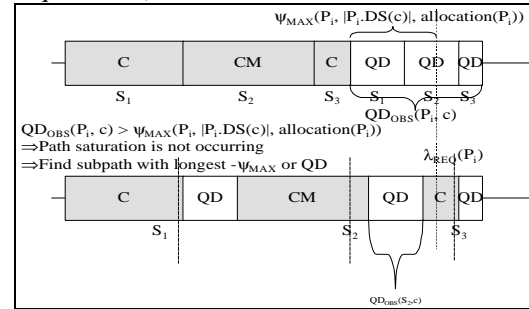


Figure 3. Path saturation not occurring.

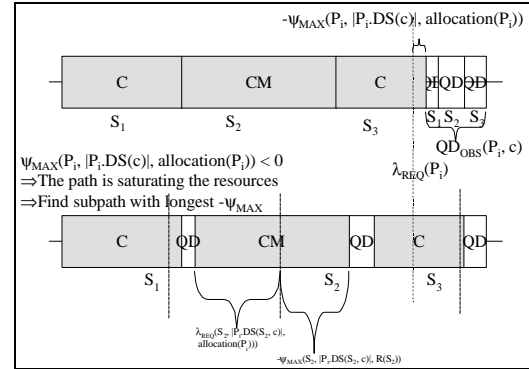


Figure 4. Path saturation occurring.

We now show our algorithm in high-level form.

Diagnosis:

- Monitor computational and communication subpaths
- Identify unhealthy subpaths
- Distinguish cause: workload or contention
- Recommend action

Subpath diagnosis algorithm:

- Queuing delay = observed time - profiled time without contention
- Maximum slack = required time - profiled time without contention (maximum allowable queuing delay)
- Maximum slack < 0: heavy workload

- scale up/combine subpath
- Queuing delay > maximum slack: heavy contention
- move/combine subpath

3. Experimental Results

This section reports on results of evaluating the diagnosis technique in actual use with the DynBench real-time benchmark [14] using the hardware configuration shown in figure 5. The experimental results show the effectiveness of the diagnosis technique for detecting and correcting unhealthy computational and communication subpaths compared to a technique without profiling and communication subpath support.

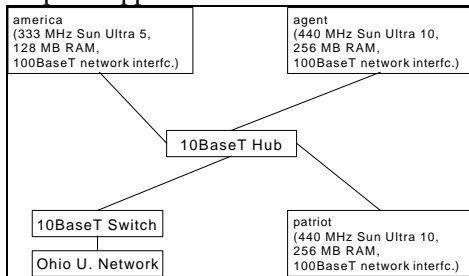


Figure 5. Hardware configuration for experiments.

The diagnosis techniques were evaluated using two assessment metrics: percentage of cycles above minslack, and average positive minslack difference. Percentage cycles above MinSlack is the percentage of cycles in the experiment in which the path latency violates the requirement, thus lower measurements are better. Average positive minslack difference is the average latency above the required latency, which measures the magnitude of violations. Again, lower measurements are better. Scenarios were chosen to cause varied workload (data stream size) and contention on the resources (with computational and communication load programs), as in figure 6. Then, the assessment measurements for all runs of a scenario using each technique were grouped by technique, and the general linear models procedure was run on SAS. The techniques evaluated were an old technique without profiling or communication subpath diagnosis, or the new algorithm with profiling and full subpath diagnosis. The “t” groupings of the means of each assessment metric identified which measurements were significantly different at the 0.05 alpha level. Figure 7 shows the mean percentage of each scenario in which violations occurred. In conclusion, experimental statistical evidence supported that the new algorithm is as effective

as the old algorithm in correcting computational loading due to workload and contention and significantly better than the old algorithm in correcting communication loading.

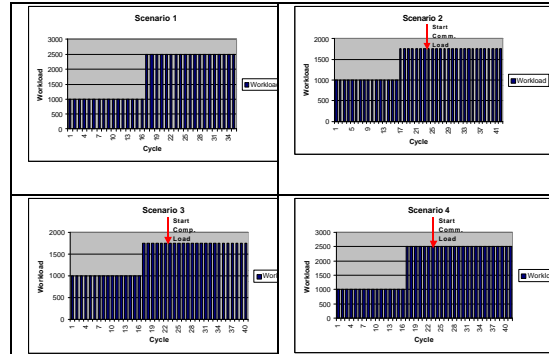


Figure 6. Workload/contention graphs.

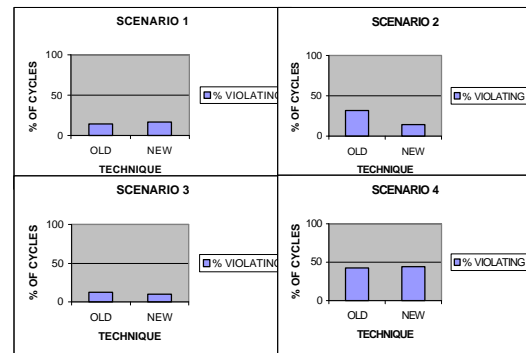


Figure 7. Experimental results comparing new and old diagnosis techniques.

4. Previous Work

RTPOOL [1], DQM [2][3][4][11][12], and EPIQ [10][13] are similar to each other in their use of discrete QoS levels. On the other hand, DeSiDeRaTa’s continuous model of resource requirements defines resource requirements in terms of the workload, which is continuous, instead of the algorithm type, which is discrete. Furthermore, DeSiDeRaTa’s resource manager dynamically adjusts its profiles of execution time depending on the current workload, rather than assuming a fixed execution time. Unlike QUASAR’s software feedback toolkit [9][15], DeSiDeRaTa’s QoS manager triggers actions that do not require a certain scheduler; rather, the resource management components carry out the actions and let the operating system’s scheduler handle the CPU-level scheduling. Rather than intercept procedure calls as QuO’s delegate objects do [18], DeSiDeRaTa’s middleware maintains a high-level view of the system by monitoring, diagnosing, and adapting paths.

5. Conclusions and Future Work

In this paper, we proposed a new technique for complete, generic computational and communication subpath diagnosis and validated it through extensive scientific experimentation. Statistical analysis supports that the new technique exhibits improved performance under contention for communication resources and comparable performance under high workload or contention for computational resources. Future work involves refining the QoS manager's ability to detect stability between bursts of QoS violations in rapid succession. As dynamic, distributed, real-time systems become more commonplace, there will be an increasing need for such intelligent QoS management techniques.

6. References

- [1] Abdelzaher, Tarek F., Ella M. Atkins, and Kang G. Shin. 1997. QoS negotiation in real-time systems and its application to automated flight control. In *Proceedings of the Third IEEE Real-Time Technology and Applications Symposium*:228–238. Los Alamitos, CA: IEEE Computer Society.
- [2] Brandt, Scott, Gary Nutt, Toby Berk, and Marty Humphrey. 1998a. Soft real-time application execution with dynamic quality of service assurance. In *Proceedings of the Sixth IEEE/IFIP International Workshop on Quality of Service*:154–163. Los Alamitos, CA: IEEE Computer Society.
- [3] Brandt, Scott, Gary Nutt, Toby Berk, and James Mankovich. 1998b. A dynamic quality of service middleware agent for mediating application resource usage. In *Proceedings of the Nineteenth IEEE Real-Time Systems Symposium*:307–317. Los Alamitos, CA: IEEE Computer Society.
- [4] Brandt, Scott, Gary Nutt, and Ken Klingenstein. 1998. A discrete and dynamic approach to application/operating system QoS resource management. In *Proceedings of the First Internet2 Joint Application/Engineering QoS Workshop*: 1–5.
- [5] Cavanaugh, Charles D., Lonnie R. Welch, and Carl Bruggeman. 1999. *A path-based design for the air traffic control problem*. Arlington, TX: The University of Texas at Arlington Department of Computer Science and Engineering. Technical Report, TR-CSE-99-001.
- [6] Cavanaugh, Charles D., Lonnie R. Welch, Farhad Kamangar, and Behrooz A. Shirazi. 1999. Quality of service forecasting for distributed, dynamic real-time systems. In *Proceedings of the Work-in-Progress Session of the 1999 IEEE Real-time Systems Symposium*.
- [7] Cavanaugh, Charles D. 2000. *Quality of service management for dynamic, distributed, real-time systems*. Ph.D. dissertation, The University of Texas at Arlington.
- [8] Cavanaugh, Charles D., Lonnie R. Welch, Behrooz A. Shirazi, Eui-Nam Huh, and Shafqat Anwar. 2000. Quality of service negotiation for distributed, dynamic real-time systems. In *Proceedings of the Eighth Workshop on Parallel and Distributed Real-time Systems (WPDRTS'00)*: 757–765. Berlin: Springer-Verlag.
- [9] Goel, Ashvin, David Steere, Calton Pu, and Jonathan Walpole. 1999. Adaptive resource management via modular feedback control. In *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems, submitted*. Los Alamitos, CA: IEEE Computer Society.
- [10] Liu, Jane W. S., Klara Nahrstedt, David Hull, Shigang Chen, and Baochun Li. July 22 1997. *EPIQ QoS characterization: draft version*. Accessed September 15 1999. Computer file. Available from <http://epiq.cs.uiuc.edu/qos-970722.pdf>.
- [11] Nutt, Gary, Toby Berk, Scott Brandt, Marty Humphrey, and Sam Siewert. 1997. Resource management for a virtual planning room. In *Proceedings of the Third International Workshop on Multimedia Information Systems*: 129–134.
- [12] Nutt, Gary, Scott Brandt, Adam Griff, Sam Siewert, Marty Humphrey, and Toby Berk. 1999. Dynamically negotiated resource management for data intensive application suites. *IEEE Transactions on Knowledge and Data Engineering*, to appear.
- [13] Shankar, Mallikarjun, Miguel DeMiguel, and Jane W. S. Liu. 1999. An end-to-end QoS management architecture. In *Proceedings of the Fifth Real-Time Technology and Applications Symposium*. Los Alamitos, CA: IEEE Computer Society.
- [14] Shirazi, Behrooz, Lonnie Welch, Binoy Ravindran, Charles Cavanaugh, and Eui-Nam Huh. 2000. DynBench: a benchmark suite for dynamic real-time systems. *Journal of Parallel and Distributed Computing Practices* (Accepted).
- [15] Steere, David C., Ashvin Goel, Joshua Gruenberg, Dylan McNamee, Calton Pu, and Jonathan Walpole. 1999. A feedback-driven proportion allocator for real-rate scheduling. *login*: 24, no. 3.
- [16] Welch, Lonnie R. 1997. Architectures of Large-Grain, Distributed Real-Time Systems. In *Proceedings of the Fifth Workshop on Parallel and Distributed Real-Time Systems*:22-26. Los Alamitos, CA: IEEE Computer Society.
- [17] Welch, Lonnie R., Binoy Ravindran, Behrooz Shirazi, and Carl Bruggeman. 1998. Specification and modeling of dynamic, distributed real-time systems. In *Proceedings of the Nineteenth IEEE Real-Time Systems Symposium*:72–81. Los Alamitos, CA: IEEE Computer Society.
- [18] Zinky, John A., David E. Bakken, and Richard E. Schantz. 1997. Architectural Support for Quality of Service for CORBA Objects. *Theory and Practice of Object Systems* 3, no. 1.