A Control Theoretical Model for Quality of Service Adaptations

Baochun Li, Klara Nahrstedt*
Department of Computer Science
University of Illinois at Urbana-Champaign
b-li@cs.uiuc.edu, klara@cs.uiuc.edu

Abstract

In a distributed environment where multiple applications compete and share a limited amount of system resources, applications tend to suffer from variations in resource availability, and are desired to adapt their behavior to the resource variations of the system. We propose a Task Control Model to rigorously model the dynamics of an adaptive system, using the digital control theory. With our Task Control Model, we are able to quantitatively analyze the stability and equilibrium of the adaptive applications, while simultaneously providing fairness guarantees to other applications in the system. Our control algorithm has also been extended to the cases where no sufficient task state information are observable. We show that even under these circumstances, our Task Control Model can still be applied and our control algorithms yield stable and responsive behavior.

1 Introduction

In a heterogeneous distributed environment, a wide variety of distributed applications demand specific Quality of Service (QoS) from the underlying supporting platform to ensure user satisfaction. Critical qualities that are usually demanded are timeliness, data quality and reliability, other qualities such as security concerns and synchronization between data streams are also occasionally required.

In a typical system, it is normally observed that multiple applications compete and share a limited amount of resources. Due to variations in the underlying distributed environment, with respect to resource availability, applications may not be able to receive constant Quality of Service from the system. However, there are a variety of *flexible* applications that can accept and tolerate resource scarcity to a certain minimum limit, and can improve its performance if given a larger share of resources. If resources above the minimum requirements are shared among all applications, statistical multiplexing gain can be improved. In addition, for the flexible applications that involve interactive activities that cannot be predicted *a priori*, it may be hard or impossible to specify a maximum demand for QoS beforehand. Based on these observations, it is necessary to support QoS adaptation mechanisms in the system.

If we examine the adaptive behavior of applications in a more detailed fashion, it is very natural to map it to an adaptive control system. In a typical control system, there is a target system to be controlled. This target system takes appropriate actions to process the input. The input is determined by a *controller* according to a control algorithm, which monitors the states inside the target system, and compares them to the desired values referred to as the *reference*. Similarly, adaptation also needs to identify the current states

*This research was supported by the Air Force Grant under contract number F30602-97-2-0121, and National Science Foundation Career Grant under contract number NSF CCR 96-23867.

of the target system based on any parameters that can be observed, and to decide input values to the target system in the future.

This paper rigorously models adaptation behavior of a distributed heterogeneous environment, based on its analogy to the above described control system. The application of the adaptive control model onto a distributed adaptive system leads to a new Task Control Model. There are two major advantages of this approach: (1) With the Task Control Model we can utilize various adaptive control policies, and rigorously derive equilibrium, stability and responsiveness properties of these policies using control theoretical techniques. (2) The Task Control Model fits well into the distributed environment, because it allows us to associate various system adaptive mechanisms, such as buffer smoothing, scaling, filtering, prefetching and others, with policies in the adaptive controllers. This mapping between the adaptive control policies and system mechanisms will result in a flexible design and implementation of adaptive middleware. The control policies will guarantee equilibrium, stability and responsiveness properties, while the mapping between policies and system mechanisms will guarantee flexible adaptation and reconfiguration of different adaptive behavior. This paper addresses the first justification.

The rest of this paper is organized as follows. In Section 2, we discuss existing work related to QoS adaptation issues. In Section 3, we propose a Task Control Model based on the Task Flow Model [7] in order to analyze adaptation behavior using the control theory. In Section 4, we analyze the stability, fairness and responsiveness of a control theoretical approach that we propose for adaptive systems. In Section 5, we extend our approach and show that the Task Control Model also applies to the case where sufficient state information for the tasks is not available. Section 6 concludes the paper and discusses future work.

2 Related Work

It has been widely recognized that many QoS-constrained distributed applications need to be adaptive in a certain QoS range between $[QoS_{min}, QoS_{max}]$ [8], in order to provide a graceful reaction to dynamic resource availability in a distributed environment. Various schemes have been proposed [1] [3] [8] [12]. This work differs from other approaches in the sense that it concentrates on determining *adaptive policies*, rather than the development of *adaptive mechanisms*. We use digital control theory to quantitatively determine the states of the adaptive system, and based on these states we activate control algorithms to adapt to the dynamics of the system.

Our work is closely related to and utilizes the knowledge of dynamic resource allocations. Other approaches in the adaptive resource allocation area have also been presented in [9] [10]. The work in [9] focuses on maximizing the utility functions, while keeping QoS received by each application within a feasible range. The work in [10] focuses on a multi-machine environment running a single complex application, and the objective is to dynamically change the configuration of the application to adapt to the environment. Our work focuses on the analysis of the actual adaptation dynamics, rather than utility factors. We also focus on an environment with multiple applications competing for a pool of shared resources, which we believe is a common scenario easily found in many actual systems.

The essence of a closed-loop control system is its feedback path. Various software and distributed systems utilize feedback information for adaptation purposes. For example, the work presented in [2] uses software feedback mechanisms that enhance system adaptiveness by adjusting video sending rate according to on-the-fly network variations. However, the algorithms used in most of the software systems are heuristic in nature, and the analysis of various adaptation properties such as stability, steady-state fairness and responsiveness is not addressed.

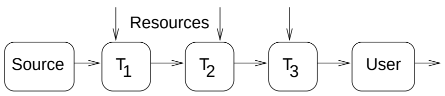
In recent research, control theories have been examined for QoS adaptation. In [11], the application of control theory is suggested as a future research direction to analyze adaptation behavior in wireless environments. In [4], a control model is proposed for adaptive QoS specification in an end-to-end scenario.

In [6], the time variations along the transmission path of a telerobotics system are modeled as disturbances in the proposed perturbed plant model, in which the mobile robot is the target to be controlled. Our work attempts to apply control theory to analyze the adaptation dynamics in a broader range of applications, and in a more rigorous fashion.

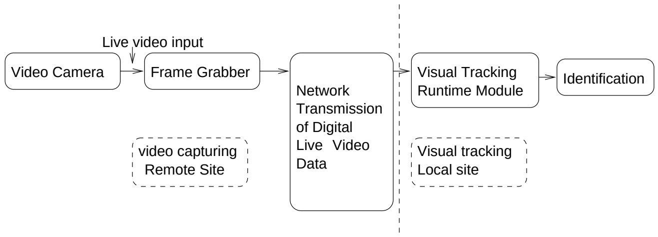
3 A Task Control Model for Quality of Service Adaptations

3.1 The Task Flow Model for Distributed Environments

In order to analyze the adaptation behavior using the control theory, we need a strict mapping between traditional control systems and the distributed environment that we study. As a start, we consider each application as an ensemble of functional components, which we refer to as *tasks*. Tasks are execution units that perform certain actions to deliver a result to other tasks or the end user.



(a) A Generic Task Flow Graph



(b) Task Flow Graph for a Visual Tracking System

Figure 1: Illustrations of the Task Flow Model for Distributed Environments

With the above definition of tasks, we utilize the Task Flow Model described in [7] and extend the model to address the modeling of adaptation behavior. In this model, the Task Flow Graph is a directed acyclic graph which consists of multiple tasks, and illustrates the consumer-producer dependencies among the tasks. A directed edge from task T_i to task T_j indicates that task T_j uses the output produced by task T_i. Tasks can be uniquely characterized by its *input quality*, *output quality* and *utilized resources*. Figure 1 illustrates a generic Task Flow Graph and its application to a distributed visual tracking system, whose objective is to track objects inside a sequence of images captured from a remote site and transferred via a network connection.

3.2 The Task Control Model

3.2.1 A Generic Task Control Model

Based on the Task Flow Model, the Task Control Model concentrates on one single task in the Task Flow Graph. This task is referred to as the *Target Task*, which is the task to be controlled. In addition, we introduce an *Adaptation Task*, which performs the adaptive control algorithm, as well as an *Observation Task*, which observes the states of the *Target Task* and feeds them back to the *Adaptation Task*.

The ideal objective of the Task Control Model is to achieve the following properties: (1) *Target Task* maintains the same output quality at the desired QoS level, regardless of variations in resource availability; (2) Dynamic changes in QoS requirements can be accommodated in a timely fashion; (3) Fairness is guaranteed among all competing tasks. These properties will be met by careful choice of adaptive control policies in the *Adaptation Task* and by assistance of the *Observation Task*.

- **Adaptation Tasks**: These tasks implement adaptive control policies and modify a set of *controllable* parameters according to a specific adaptive control algorithm. *Controllability* has a two-fold interpretation. First, a parameter is controllable means that by changing end-system configuration dynamically it is possible to affect its values; Second, it also means that by changing values of a parameter, it is possible to affect the internal states of the task and thus affect the output quality.

- **Task States**: In order to control the input quality to achieve the ideal range of the output quality, we need a precise analytical model to characterize the internal dynamics in the *Target Task*. We refer to the parameters in this model as *Task States*. The most important task states in any task are its parameters related to its resources. Any tasks have to consume resources in order to perform actions on input and produce output.

- **Observation Tasks**: *Adaptation Tasks* need knowledge about the current states of the Target Task, in order to perform the control. If the task states are observable, they are observed by the Observation Tasks and fed back into the *Adaptation Tasks*. Otherwise, if some related parameters can not be observed, the *Observation Task*s will estimate or predict the current states, based on the estimation algorithm of its choice.

Once the above notations are established, we are ready to present the Task Control Model based on the control theory. In the most generic fashion, we use the vector $\mathbf{s}$ for the vector of task states, the vector $\mathbf{u}$ for the vector of controllable input parameters, the vector $\mathbf{y}$ for the vector of observed output parameters of the task, the vector $\mathbf{v}$ for the uncontrollable variations in the task, and the vector $\mathbf{n}$ for the observation errors. Using the above notations, we model the *Target Task* in the Task Control Model with the following equations:

$$\frac{d\mathbf{s}(t)}{dt} \equiv \dot{\mathbf{s}}(t) = \mathbf{f}[\mathbf{s}(t), \mathbf{u}(t), \mathbf{v}(t), t] \tag{1}$$

$$\mathbf{y}(t) = \mathbf{h}[\mathbf{s}(t), \mathbf{n}(t), t] \tag{2}$$

With the above definition, the task is said to be at *equilibrium* when:

$$\dot{\mathbf{s}}(t) = 0 = \mathbf{f}[\mathbf{s}(t), \mathbf{u}(t), \mathbf{v}(t), t] \tag{3}$$

An equilibrium is *stable* if small disturbances do not cause the state to diverge and oscillate. Otherwise, it is an unstable equilibrium.

The above stated definitions are generic and can illustrate a wide variety of adaptation capabilities of the *Target Task*. According to these definitions, the *Target Tasks* may be continuous in time, non-linear and time-varying. In this paper we study a subset, namely, the tasks that can be approximated without loss of accuracy by discrete and linear equations as the following form:

$$\mathbf{s}(k+1) = \mathbf{G}\mathbf{s}(k) + \mathbf{H}\mathbf{u}(k) + \mathbf{v}(k) \tag{4}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{s}(k) + \mathbf{n}(k) \tag{5}$$

where $\mathbf{G}$, $\mathbf{H}$ and $\mathbf{C}$ are constant matrixes. We assume in later examples that the *Target Tasks* can be characterized accurately by discrete-time and linear equations in Equation (4) and (5).

3.2.2 An Example of the Task Control Model

The Task Flow Model stated in Section 3.2.1 can characterize a wide variety of tasks. To demonstrate a concrete example, we consider the following scenario. Let us assume multiple tasks competing for a shared resource pool with the capacity C_{max}. Each task makes *requests* for resources in order to perform their actions on inputs and produce outputs. These requests may be *granted* or *outstanding*. If a request is granted, resources are allocated immediately. Otherwise, the request waits in the outstanding status until it is granted. The system is granting requests from multiple tasks with a constant *request granting rate c*.

The mapping between the abstract notation *resource requests* and the actual services processing the resource requests varies among different types of system resources. For *temporal* resources, such as processing bandwidth and transmission throughput, where the resources are shared in a temporal fashion, outstanding resource requests may be mapped to the waiting queue, and granted requests may be mapped to allocated temporal resources, such as bandwidth. For *spatial* resources, such as volatile or non-volatile storage capacity, outstanding requests may be mapped to the actively used and occupied capacity, and granted requests may be mapped to the reclaimed capacity by the system due to inactivity. The framework presented in this section applies to both cases.

Figure 2 illustrates the above scenario and the mapping between the classic control and Task Control Model. Naturally, if the *Target Task* is greedy and makes an excessive number of resource requests in a short period of time, it is not fair to other tasks sharing the same resource pool. Thus, the request rate need to be throttled by the *Adaptation Task*. What the *Adaptation Task* tries to control is the resource request rate made by the *Target Task*, so that it does not exceed its fair share.

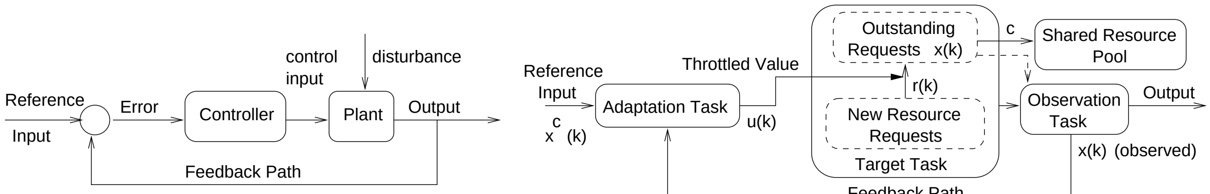


Figure 4a: A block diagram in a classic control system

Figure 4b: The Task Control Model used for QoS adaptation

Figure 2: The mapping between the classic control model and the Task Control Model

In this scenario, we define the following notions for the system with a *Target Task* T_i:

1. t_c is a constant sampling time interval, which is the time elapsed in interval $[k, k+1]$, k being time instants;

2. $r_i(k)$ is the number of requests made by T_i in $[k, k+1]$;

3. $u_i(k)$ is the number of requests in $[k, k+1]$ allowed by the *Adaptation Task* of T_i to enter the outstanding queue, which is referred to as the *adapted request rate*;

4. $x(k)$ is the *total number* of *outstanding resource requests* made by *all tasks* at time k;

5. $M(k)$ is the total number of active tasks competing for resources in the system;

6. $A(k)$ is the set of tasks at time k whose request rate is throttled by their respective *Adaptation Tasks*, $N(k)$ is the set of other tasks that are not affected by the *Adaptation Tasks*;

7. $l(k)$ is the number of tasks in $A(k)$, $M(k) - l(k)$ is the number of tasks in $N(k)$. We assume that both $M(k)$ and $l(k)$ stays constant within one time interval $[k, k+1]$.

8. w_i is the static weight of T_i showing its relative priority or importance compared to other tasks.

Using the above notation, the derivative of outstanding resource requests can be described as follows:

$$\dot{x} = x(k) - x(k-1) = \sum_{i=0}^{M(k-1)} u_i(k-1) - c \tag{6}$$

The Difference Equation (6) depicts the internal dynamics of the adaptive system.

The objective of the control is to maintain the number of outstanding requests x to stay around a specific *reference* value $x^c(k)$. Under the assumption that the dynamics of the adaptive system behave according to Equation (6), we can derive a control algorithm/policy in the *Adaptation Task* for T_i to calculate $u_i(k)$ values, which will lead to the desired values for x. For example, if a standard proportional-integral-derivative (PID) control [5] [1] is engaged, then $u_i(k)$ obeys the equation

$$u_i(k) = u_i(k-1) + \alpha[x^c(k) - x(k)] + \beta\{[x^c(k) - x(k)] - [x^c(k-1) - x(k-1)]\} \tag{7}$$

where α and β are configurable scaling factors. This is one of the many effective control algorithms illustrating the ability of the Task Control Model to capture the adaptation dynamics and map these dynamics onto a classic control model.

4 A Control Theoretical Approach for Quality of Service Adaptation

This section continues with a rigorous analysis of the stability, fairness and responsiveness of the Task Control Model characterized by the *Target Task* in Equation (6) and the PID control algorithm in Equation (7), in order to prove the validity of our approach using the Task Control Model.

We assume in our analysis that the controllable parameter of task T_i is the request rate $r_i(k)$. The Adaptation Task may control $r_i(k)$ at a lower rate $u_i(k)$, namely, $u_i(k) \leq r_i(k)$. The PID control algorithm presented in Equation (7) becomes:

$$u_i(k) = \Psi_{r_i(k)} \ \{ \ u_i(k-1) + \alpha(x^c(k) - x(k)) + \beta[(x^c(k) - x(k)) - (x^c(k-1) - x(k-1))] \ \} \tag{8}$$

Where $\Psi_t(x)$ is defined as:

$$\Psi_t(x) = \begin{cases} 0 & \text{if } x < 0, \\ t & \text{if } x > t, \\ x & \text{otherwise.} \end{cases} \tag{9}$$

In addition, since at time k we assume that, among all tasks, $l(k)$ tasks are throttled by their respective adaptation tasks, and $M(k) - l(k)$ tasks are not affected, we conclude that the total number of outstanding resource requests in the system shows the following dynamic property, which is an extension to Equation (6):

$$\begin{aligned} x(k) &= \Psi_{C_{max}} \left\{ x(k-1) + \sum_{T_i \in A(k-1)} u_i(k-1) + \sum_{T_i \in N(k-1)} r_i(k-1) - c \right\} && (10) \\ &= \Psi_{C_{max}} \left\{ x(k-1) + l(k-1)\bar{u}(k-1) + R(k-1) - c \right\}, \text{ where} && (11) \end{aligned}$$

[1]PID control is a classic control algorithm where the control signal is a linear combination of the error, the time integral of the error, and the rate of change of the error.

$$R(k) = \sum_{T_i \in N(k)} r_i(k), \text{ and} \quad (12)$$

$$u_i(k) = \gamma_i(k)\ l(k)\ \bar{u}(k) \quad (13)$$

where $\bar{u}(k)$ is the average rate for all $u_i(k)$ that satisfies $u_i(k) < r_i(k)$. In this equation, $\gamma_i(k)$ is the *dynamic weight* of task T_i which indicates priority for resource requests, and satisfies $\sum_{T_i \in A(k)} \gamma_i(k) = 1$. The dynamic weight of T_i can be derived from the *static weight* w_i of T_i, with the following calculation:

$$\gamma_i(k) = \frac{w_i}{\sum_{T_j \in A(k)} w_j} \quad (14)$$

Combining Equation (8) and Equation (11), we obtain the complete characterization of the adaptation system. Associated with each task T_i, we have a *static weight* w_i, and three internal task states:

$$\mathbf{s}_i(k) = \left\{ x^c(k) - x(k), x^c(k-1) - x(k-1), \gamma_i(k-1)\ l(k-1)\ [\bar{u}(k-1) - \frac{c - R(k-1)}{l(k-1)}] \right\}^T \quad (15)$$

Where $x^c(k)$ and $\frac{c-R(k)}{l(k)}$ are the equilibrium values of $x(k)$ and $\bar{u}_k$, respectively. The detailed analysis of the equilibrium states is given in Section 4.1.

4.1 Equilibrium Analysis

Now that we have established the control algorithm in the *Adaptation Task*, we start to analyze the exact value that the system stays at equilibrium. The ideal case is that $x(k)$ always stays the same as the reference $x^c(k)$. Let us assume that for a specific period of time $[k_1, k_2]$, $x^c(k)$, $l(k)$, $M(k)$ and $R(k)$ are all stable and stay at constants x_s^c, l_s, M_s and R_s, respectively. Then we show the following properties:

Theorem 1: Within $[k_1, k_2]$, The number of outstanding resource requests x in the system, established by Equation (8) and (11), will converge to an equilibrium value which equals to the reference value x_s^c. In addition, the system also fairly shares resources among competing tasks according to their static weights.

Proof: Let x_s and $\bar{u}_s$ be the equilibrium values corresponding to the system established by Equation (8) and (11).

$$x_s = \Psi_{C_{max}} \{x_s + l_s \bar{u}_s + R_s - c\} \quad (16)$$

$$\gamma_i\ l_s\ \bar{u}_s = \Psi_{r_i} \{\gamma_i\ l_s\ \bar{u}_s + \alpha(x_s^c - x_s)\} \quad (17)$$

Ignoring the threshold cases, the solution to Equation (16) and (17) is

$$\bar{u}_s = \frac{c - R_s}{l_s} \quad (18)$$

$$x_s = x_s^c \quad (19)$$

Equation (19) directly proves the first part of the theorem. Assume the stable set of throttled tasks is A_s, Equation (18) can be rewritten for task T_i at equilibrium as follows:

$$(u_i)_s = \gamma_i\ l_s\ \bar{u}_s = \frac{w_i\ l_s}{\sum_{T_j \in A_s} w_j} \frac{c - R_s}{l_s} = \frac{w_i\ l_s}{\sum_{T_j \in A_s} w_j} \left\{ \frac{c}{M_s} + \frac{(M_s - l_s)\frac{c}{M_s} - R_s}{l_s} \right\} \quad (20)$$

Equation (20) presents the following weighted max-min fairness property. Each task T_i can be granted at least a w_i share of the resources; In addition, if $M_s - l_s$ tasks request less than their fair share, namely, only l_s tasks are adapted, then the free portion $\frac{(M_s - l_s)c}{M_s} - R_s$ can be distributed among those who are throttled and thus need these resources, according to their static weights, which identify their relative priority and importance. This concludes the proof. □

4.2 Stability Analysis

The concept of *stability* has a two-fold meaning. First, in an environment of multiple tasks simultaneously sharing the limited availability of resources, the ensemble of the adaptation activities in all tasks need to be *stable*, which means that when the number of active tasks is fixed, system resources allocated to each task settle down to an equilibrium value in a definite period of time. This definition also implies that, if a new task becomes active, existing active tasks will adjust their resource usage so that after a brief transient period, the system settles down to a new equilibrium. Second, stability implies that with respect to variations in resource availability due to unpredictable and physical causes, for example a volatile wireless connection, adaptation activities do not suffer from oscillations, which are undesirable because they cause both fluctuations in user-perceptible qualities, and an excessive amount of adaptation attempts that may occupy too much resource to overload the system.

In order to converge to the equilibrium of the system regardless of disturbances and statistical multiplexing, we need to prove that the system is stable. Due to the nonlinear nature of the system given by Equation (8) and (11), we are unable to derive a global and absolute stability condition, which is the case for most systems with nonlinear properties. However, formal conditions for local asymptotic stability can be addressed analytically. We present the following theorem related to local asymptotic stability conditions.

Theorem 2: The adaptation system established by Equation (8) and (11) is asymptotically stable for task T_i around a local neighborhood, under the condition that $\alpha > 0,\ \beta > 0$, and $\alpha + 2\beta < 4\gamma_i$.

Proof: Given the states defined in Equation (15), we define

$$e(k) = x^c(k) - x(k) \tag{21}$$

$$\hat{u}_i(k) = \gamma_i\ l(k)\ [\bar{u}(k) - \frac{c - R(k)}{l(k)}] \tag{22}$$

In order to examine the asymptotic stability properties, we simplify the dynamic equations (8) and (11) in the neighborhood of equilibrium by: (1) removing the nonlinearities introduced by $\Psi_t(x)$ at both thresholds; (2) treating $l(k)$ and $R(k)$ as constants in the neighborhood of the equilibrium. Thus, Equations (8) and (11) become:

$$\hat{u}_i(k) = \hat{u}_i(k-1) + \alpha\ e(k) + \beta\ [e(k) - e(k-1)] \tag{23}$$

$$x(k) = x(k-1) + \frac{1}{\gamma_i}\hat{u}(k-1) \tag{24}$$

We perform z-transform on Difference Equations (23) and (24) to obtain $D_i(z)$ and $G_i(z)$, respectively. Thus, the transfer function $F_i(z)$ of the entire system is [5]:

$$F_i(z) = \frac{D_i(z)G_i(z)}{1 + D_i(z)G_i(z)} = \frac{\frac{1}{\gamma_i}[(\alpha+\beta)z - \beta]}{z^2 + (\frac{\alpha}{\gamma_i} + \frac{\beta}{\gamma_i} - 2)z - (\frac{\beta}{\gamma_i} - 1)} \tag{25}$$

We then consider the discrete characteristic equation of the above:

$$z^2 + (\frac{\alpha}{\gamma_i} + \frac{\beta}{\gamma_i} - 2)z - (\frac{\beta}{\gamma_i} - 1) = 0 \tag{26}$$

According to theorems in the digital control theory [5], in order for the system to be stable, all roots of Equation (26) need to be within the stability boundary, which is the unit circle. In other words, for any root z, we need $|z| < 1$. It can be proved that this property holds if the following condition is valid (the proof is omitted for space limitations):

$$\alpha > 0,\ \beta > 0,\ \text{ and } \alpha + 2\beta < 4\gamma_i \tag{27}$$

Equation (27) concludes the proof. □

It is obvious to see from Theorem 2 that the asymptotic stability of the adaptation for task T_i is determined by an appropriate choice of α and β. It then follows that in order to guarantee that the entire system is stable, we need to choose α and β so that for any task T_i with any static weight values w_i, stability is ensured.

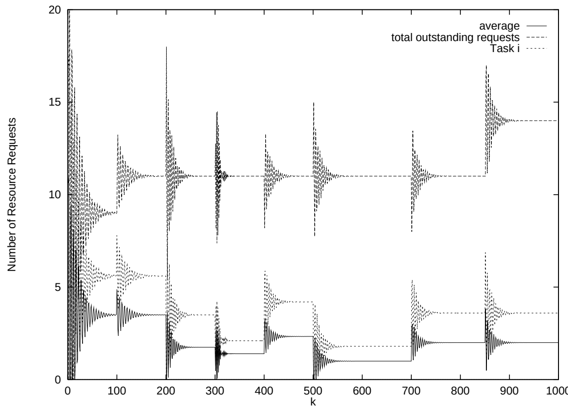
Corollary: There exist appropriate values of parameters α and β so that all the tasks in the system are stable, for any pre-determined static weight w_i for task T_i.

Proof: Assume w_{min} is the minimum value among all pre-determined w_i. α and β can be chosen to satisfy

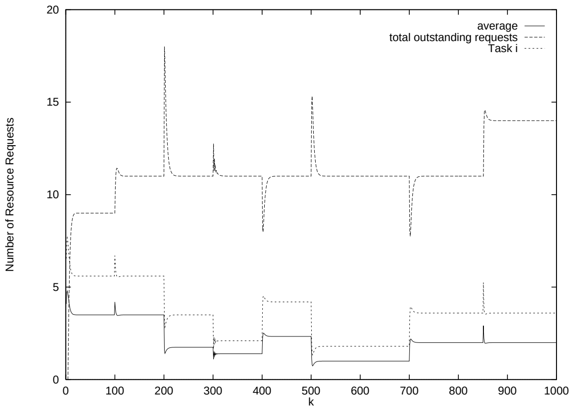
$$\alpha > 0,\ \beta > 0,\ \text{ and }\ \alpha + 2\beta < 4\,\frac{w_{min}}{\sum_{\forall i} w_i} < 4\,\frac{w_i}{\sum_{T_j \in A(k)} w_j},\ \forall i,\ k \tag{28}$$

It follows from Theorem 2 that if these conditions hold, the system will be stable for any task T_i with a static weight w_i. □

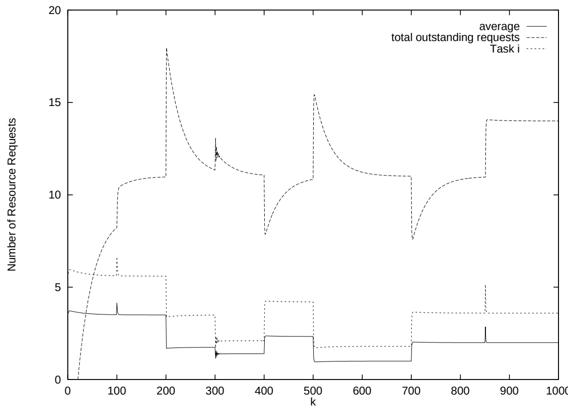
4.3 Responsiveness Configuration



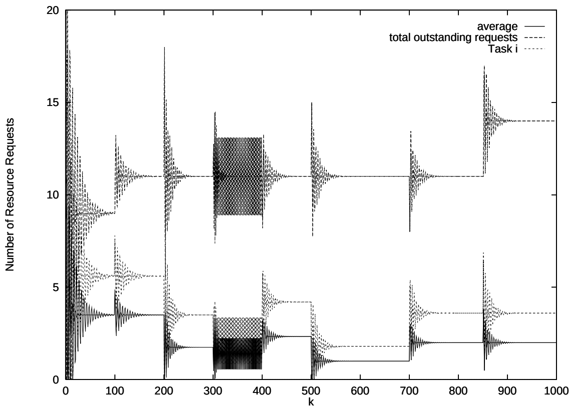
(a) $\alpha = 1,\ \beta = 0.09$



(b) $\alpha = 0.1,\ \beta = 0.45$



(c) $\alpha = 0.015,\ \beta = 0.5$



(d) $\alpha = 0.7,\ \beta = 0.25$

Figure 3: Illustrations of Configurable Responsiveness and Dynamic Responses

In addition to stability requirements, It is also desired that the system responds quickly to changes in both resource availability and QoS requirements of the tasks. Responsiveness is determined by the configurable parameters in control algorithm. In the case of a PID control algorithm in Equation (8), α and β are configurable as long as the stability conditions in Theorem 2 hold. The actual configuration is tailored to the needs of the system.

Four illustrations are given in Figure 3(a) - 3(d) to show the effects of different configurations on the system. In all four graphs, we simulated the system established by Equation (8) and (11) for 1000 time intervals. The changes of $l(k)$, $R(k)$ and $x^c(k)$ are given in Table 1:

Time k	*0-100*	*100-200*	*200-300*	*300-400*	*400-500*	*500-700*	*700-850*	*850-1000*
$l(k)$	2	2	4	5	3	3	3	3
$R(k)$	3	3	3	3	3	7	4	4
$x^c(k)$	9	11	11	11	11	11	11	14

Table 1: The values of $l(k)$, $R(k)$ and $x^c(k)$ used in the illustration

It can be seen from these illustrations that the dynamics of the system is affected significantly by different configurations of the *Adaptation Task*. In Figure 3(d), it even starts to oscillate in the interval $[300, 400]$. The configuration in Figure 3(b) reaches equilibrium much faster than Figure 3(c), and both of them do not show the oscillating transient response shown in Figure 3(a). Overall, the illustrations show that the *Adaptation Task* is configurable according to the needs of the system. For the PID control algorithm in Equation (8), there are two parameters to configure. However, for more complex control algorithms, there will be more dimensions for finer tuning of the transient responses.

5 An Extension to the Control Theoretical Approach

We proved that the control theoretical approach presented in Section 4 is stable, fair to all participating tasks and configurable with respect to responsiveness. However, this approach is only applicable if the *Observation Task* can observe full system states, and feed them back to the *Adaptation Task* in a timely fashion. This is true if the *Target Task* utilize local resources only, such as local processing power and local storage capacity. However, in a distributed environment, there may be cases that tasks share global resources in a distributed fashion. This renders it hard or impossible for the *Observation Task* to locate and observe full system states information.

In one example, the *Target Task* is a transmission task, which relocates data between different sites in a distributed environment. The resource it uses is transmission throughput, and it is shared by other tasks. The quantity and behavior of these tasks may be unpredictable and unknown to the *Observation Task*. In the approach proposed in previous sections, the *Observation Task* needs to provide $x(k)$, which is the total number of outstanding requests for *all tasks*. This is naturally impossible given an unpredictable distributed environment.

The Task Control Model still applies to this scenario. However, the fairness property cannot be guaranteed for the lack of global state information. In order to demonstrate this statement, we take the example that the transmission task is in charge of transmitting data between two end systems. By analyzing this example we conclude that the same PID control algorithm in Section 4 can still be applied, i.e. it preserves equilibrium and stability properties. Only the model for the *Target Task* needs to be slightly modified.

We assume that the transmission throughput allocated to the transmission task, T_i, is not fixed and will vary according to the global state of the distributed environment, such as activities of other tasks sharing the same resource. Rather than considering $x(k)$ as the total number of outstanding requests for all tasks, we consider $x(k)$ as the number of outstanding requests made by task T_i. This is the number of data units *in flight* in T_i before reaching destination.

Assume $u(k)$ is the number of new requests made in T_i for resources. This is modified by the *Adaptation Task*, if necessary, in order to control T_i so that $x(k)$ stays at the reference value $x^c(k)$. Also assume $c(k)$ is the number of granted requests in $[k, k+1]$. Because allocated throughput for T_i varies, $c(k)$ will change accordingly. In implementation, $u(k)$ can be mapped to the number of data units sent into the transmission path in $[k, k+1]$, and $c(k)$ can be mapped to the number of data units received at the destination in $[k, k+1]$.

We now can give the equation characterizing the *Target Task* T_i:

$$x(k) = x(k-1) + u(k-1) - c(k-1) \tag{29}$$

In order to apply the PID control algorithm proposed in Section 4, we only need the *Observation Task* to estimate $x(k)$. Implementation-wise, there are various mechanisms to solve this problem, such as the utilization of resource management cells in an ATM network. We take one of the solutions as an example. Assume that the destination acknowledges all the received data units to the source, and assume $E(k)$ is the total number of data units unacknowledged at source at time k. In addition, assume that p_{max} equals to the maximum propagation delay of one data unit between source and destination in either direction.

With these assumptions, we can estimate $x(k)$ in the *Observation Task* at source by:

$$x(k) = E(k) - \sum_{i=k-\lceil \frac{p_{max}}{t_c} \rceil}^{k-1} c(i) \tag{30}$$

This is because the total number of unacknowledged data units is either in flight from source to destination, or received by destination. Equation (30) characterizes the *Observation Task*.

The PID control algorithm for the *Adaptation Task* can thus be given as:

$$u(k) = u(k-1) + \alpha e(k) + \beta[e(k) - e(k-1)], \text{ where} \tag{31}$$

$$e(k) = x^c(k) - x(k) \tag{32}$$

Equations (29) and (31) fully characterize the adaptation system. We have the following theorems about equilibrium analysis and asymptotic stability around a local neighborhood.

Theorem 3: Within the time interval when $x^c(k)$ remains at a constant X_s^c, the number of data units in flight x in the system established by Equation (29) and (31) will converge to an equilibrium value which equals to the reference value x_s^c. In addition, u will converge to an equilibrium value which equals to c.

Proof: Let x_s, u_s and c_s be the values at equilibrium of x, u and c. We then have

$$x_s = x_s + u_s - c_s \Rightarrow u_s = c_s \quad \text{and} \tag{33}$$

$$u_s = u_s + \alpha[x_s^c - x_s] + \beta[x_s^c - x_s - (x_s^c - x_s)] \Rightarrow x_s = x_s^c \tag{34}$$

Equations (33) and (34) conclude the proof. □

Theorem 4: The adaptation system established by Equation (29) and (31) is asymptotically stable for task T_i around a local neighborhood, under the condition that $\alpha > 0,\ \beta > 0$, and $\alpha + 2\beta < 4$.

Proof: Around a local neighborhood, assume $c(k-1) = c(k) = c_l$. Let $\hat{u}(k) = u(k) - c_l$, we have

$$\hat{u}(k) = \hat{u}(k-1) + \alpha e(k) + \beta[e(k) - e(k-1)] \tag{35}$$

$$x(k) = x(k-1) + \hat{u}(k-1) \tag{36}$$

Equations (35) and (36) are identical to Equations (23) and (24). The proof follows from Theorem 2. □

To conclude, we believe that the Task Control Model still yields asymptotic stability and responsiveness configurability, despite of incomplete task state information. It is possible to apply the same PID control algorithm in the *Adaptation Task* to these situations, if the models for the *Target Task* and *Observation Task* are appropriately established.

6 Conclusions

Our work made two contributions. First, we established the Task Control Model based on the Task Flow Model presented in [7] and rigorously defined the mapping between classic control systems and the Task Control Model. Second, we proposed a PID control algorithm and analyzed fairness properties, asymptotic stability conditions, and responsiveness of the adaptation behavior. In addition, we also proved that the Task Control Model yields the same equilibrium and stability properties despite of incompleteness of task state information, as in the example of controlling a transmission task. Obviously, control theory itself is not new; our contribution is to propose a model that successfully applies the control theory to the practice of QoS adaptations. The design and implementation of an experimental adaptive QoS framework in the middleware layer based on the model presented in this paper is in progress.

References

[1] A. Campbell and G. Coulson. QoS Adaptive Transports: Delivering Scalable Media to the Desk Top. *IEEE Network*, 1997.

[2] S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole. A Distributed Real-Time MPEG Video Audio Player. *Proceedings of the 5th International Workshop on Network and Operating System Support of Digital Audio and Video (NOSSDAV'95)*, April 1995.

[3] Z. Chen, S. Tan, R. Campbell, and Y. Li. Real Time Video and Audio in the World Wide Web. *Proceedings of Fourth International World Wide Web Conference*, 1995.

[4] J. DeMeer. On the Specification of End-to-End QoS Control. *Proceedings of 5th International Workshop on Quality of Service '97*, May 1997.

[5] G. Franklin and J. Powell. *Digital Control of Dynamic Systems*. Addison-Wesley, 1981.

[6] F. Goktas, J. Smith, and R. Bajcsy. Telerobotics over Communication Networks: Control and Networking Issues. *36th IEEE Conference on Decision and Control*, 1997.

[7] D. Hull, A. Shankar, K. Nahrstedt, and J. Liu. An End-to-End QoS Model and Management Architecture. *Proceedings of IEEE Workshop on Middleware for Distributed Real-time Systems and Services*, December 1997.

[8] S. Lu, K.-W. Lee, and V. Bharghavan. Adaptive Service in Mobile Computing Environments. *Proceedings of 5th International Workshop on Quality of Service '97*, May 1997.

[9] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A Resource Allocation Model for QoS Management. *18th IEEE Real-Time System Symposium*, 1997.

[10] D. Rosu, K. Schwan, S. Yalamanchili, and R. Jha. On Adaptive Resource Allocation for Complex Real-Time Applications. *18th IEEE Real-Time System Symposium*, 1997.

[11] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. *Proceedings of the ACM Symposium on Principles of Distributed Computing*, 1996.

[12] N. Yeadon, F. Garcia, A. Campbell, and D. Hutchison. QoS Adaptation and Flow Filtering in ATM Networks. *Proceedings of the Second International Workshop on Multimedia: Advanced Teleservices and High Speed Communication Architectures*, 1994.