Optimal State Prediction for Feedback-Based QoS Adaptations

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

Abstract

In heterogeneous network environments with performance variations present, complex distributed applications, such as distributed visual tracking applications, are desired to adapt themselves and to adjust their resource demands dynamically, in response to fluctuations in either end system or network resources. By such adaptations, they are able to preserve the user-perceptible critical QoS parameters, and trade off non-critical ones. However, correct decisions on adaptation timing and scale, such as determining data rate transmitted from the server to clients in an application, depend on accurate observations of system states, such as quantities of data in transit or arrived at the destination. Significant end-to-end delay may obstruct the desired accurate observation. We present an optimal state prediction approach to estimate current states based on available state observations. Once accurate predictions are made, the applications can be adjusted dynamically based on a control-theoretical model. Finally, we show the effectiveness of our approach with experimental results in a client-server based visual tracking application, where application control and state estimations are accomplished by middleware components.

1 Introduction

When complex distributed applications demand a particular level of Quality of Service (QoS) from the underlying system in a heterogeneous network environment, only those systems that provide system-wide end-to-end QoS guarantees (via CPU and network resource reservation and admission control) are able to meet such demands. If this is not the case, such as the Internet with best-effort services, applications may experience significant variations in resource availability. These variations are caused by either physical limitations, in the case of wireless links, or dynamic multiplexing of multiple competitive access to a limited pool of resources.

However, there exist *flexible* applications that present the following characteristics: First, they can accept and tolerate resource scarcity to a certain minimum bound, and can improve its performance if given a larger share of resources. Second, they are willing to sacrifice the performance/quality of some application-level services in order to preserve the performance/quality of critical functions. For these flexible applications, it is possible to adapt to the availability variations and still manage to preserve QoS for critical parameters. Application-level adaptation support is also necessary when it is hard to specify an upper bound of QoS demand for reservation, e.g., for interactive applications.

The client-server based visual tracking application is an example of flexible applications. A visual tracking server grabs live video frames in real time from a camera, and sends them over the network to the visual tracking client. The client executes a computationally intensive tracking algorithm, which tracks the coordinates of interested objects. Our interests focus on key application QoS parameters such as the *precision* of tracking algorithms, which depends on video quality, network bandwidth availability, jitter and other QoS parameters. As long as tracking precision is preserved, other parameters in the application, such as video quality, can be dynamically tuned, adjusted and reconfigured.

In order to adjust the application appropriately, and to decide when, how and to what extent for the application to adapt, accurate identification of current system states is needed, based on observable parameters. This *observe and control* process resembles a control system, where control signals are determined by a *controller*, based on the current state estimates. Our previous work takes advantage of control theory to model this process. As a consequence, a Task Control Model was developed and

*This research was supported by the Air Force Grant under contract number F30602-97-2-0121, National Science Foundation Grant under contract number NSF EIS 98-70736, and National Science Foundation Career Grant under contract number NSF CCR 96-23867.

theoretical results were given to prove stability and fairness properties in the model [11]. The objective of the approach was to optimally adjust the internal parameters and semantics of flexible applications, with a centralized control algorithm.

The effectiveness of a control algorithm depends on accurate observations of system states. However, in a distributed application with the presence of end-to-end delays, many system states are not directly observable in the end system, thus need to be estimated. For example, in the visual tracking application, in order to control the application and dynamically adjust the data rate transmitted from server to client, we need to obtain system states such as quantities of data in transit or arrived at the client. Significant end-to-end delay may obstruct the desired accurate observations, when estimations are used instead.

The key contributions of this paper are the following. (1) *An extended Distributed Task Control Model*: The Task Control Model introduced in [11] is extended from a model focusing on local resources such as CPU, to a distributed model focusing on bandwidth availability in Transmission Tasks. The tradeoff is that the fairness property that was previously proved can no longer be guaranteed. (2) *A linear model for Transmission Tasks*: To characterize the Transmission Tasks in a distributed application, we develop a linear model with concrete coefficients, on which state estimation techniques are based. (3) *Optimal state prediction mechanisms*: Accurate control signals are based on precise estimates of system states. With the presence of significant end-to-end delay that obstructs accurate state observations, we present an optimal prediction approach to estimate current system states based on available observations. We adopt optimal estimation theories such as the Kalman Filter in our approach. Assisted by optimal state prediction techniques, the application can be optimally controlled to adapt to dynamic variations. (4) *Verification with visual tracking*: We show the validity of our approach by experiments with a client-server based visual tracking application, where tracking precision is the crucial QoS parameter considered, and state prediction mechanisms are implemented as middleware components.

The rest of this paper is organized as follows. In Section 2, we discuss existing related work. In Section 3, we present an overview of the middleware control architecture, in which optimal estimation algorithms contribute a key functionality. In Section 4, we focus on modeling the Transmission Task in a distributed application. In Section 5, we present our optimal prediction approach to consider significant delays in the state estimation process. In Section 6, we show experiments with the client-server based visual tracking application. Section 7 concludes the paper and discusses future work.

2 Related Work

In recent research, control theories have been examined for QoS adaptation. In [3], a control model is proposed for adaptive QoS specification in an end-to-end scenario. In [4], 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. In our previous work [11], theoretical proofs are given for various properties applying control theory to model QoS adaptation.

The issues related to application-level QoS adaptation has also been studied by various previous work. 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. In [5], the authors proposed application adaptation at the configuration level, which carries out transparent transition from primary components to alternative components, as well as at the component level, which redistributes resources in different components so that a QoS tradeoff can be made. In [1], a software framework was proposed for network-aware applications to adequately adapt to network variations. Similarly to the above, our approach models applications as a series of tasks and assisted by the feedback loop. However, we differ in the sense that we take the view that middleware components control the adaptation behavior of applications, and we propose a separation of control and estimation algorithms. With respect to control, proper choices are made on adaptation timing, scale and methods used, which balances between the frequency and responsiveness of adaptation actions within the application; With respect to estimation, optimal predictions are being made to obtain the best possible estimate of actual states.

Optimal control and estimation theories, e.g., Kalman Filters, have been previously applied to flow control in high-speed networks. In [9], a Kalman Filter was given for state estimation in a Packet-Pair flow control mechanism. In [10], Kalman Filter was also used to shape traffic in a collection of VC sources in one VP of an ATM network, in this case the system state is the number of active transmission sources. Our work distinguishes itself from previous work in the following way: (1) Kalman Filter is used as a optimal prediction algorithm, instead a filtering or smoothing algorithm; (2) We use a different and a macroscopic system model to interpret the dynamics of the Transmission Task; (3) Application-specific adaptation are performed in a slower time scale (in multiples of round-trip delays) and (4) The middleware components make control decisions in the application level, and based on application-specific semantics, rather than on the packet level via a traffic shaper.

3 The Middleware Control Architecture

We have adopted a middleware solution in order to implement a centralized control of all active applications. We present the general architecture of the middleware solution in this section, followed by detailed discussion of the model and algorithms proposed for optimal state predictions, which participate in the design of the overall architecture.

A major objective of the architecture is to implement the *observe and control* process, i.e., to observe current system states in the distributed environment and produce control signals to the complex distributed applications. These signals determine the actual adaptation actions (reconfigurations or parameter tuning) within the applications. The architecture consists of two parts: the *Adaptors* and *Configurators*. In an end system, each Adaptor corresponds to a single type of resource, such as CPU or transmission bandwidth, and consists of an *Adaptation Task* and an *Observation Task*. Each Configurator corresponds to a single target application, and makes control decisions based on the output of Adaptors corresponding to several types of resources. The interaction among various middleware components and applications is made through available specific service enabling platform, such as CORBA or DCOM. Our middleware architecture uses *active* components, in the sense that these components call external interfaces in the applications in order to control them. Figure 1 shows an overview of the architecture.

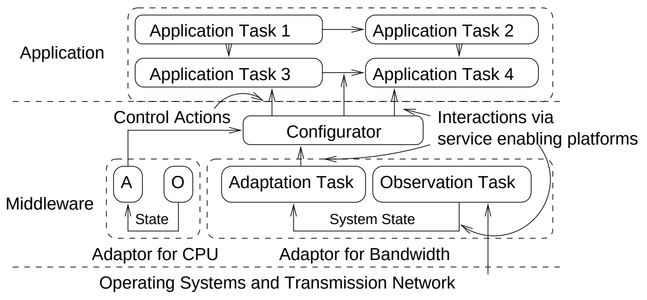


Figure 1: The Middleware Control Architecture

The optimal prediction algorithms presented in this paper are implemented in the Observation Task, a part of the Adaptor. The goal of the design is to accurately obtain system states at a specific instant in a distributed environment, with the presence of significant end-to-end delays. The mechanisms below show that the middleware components can optimally estimate the states without soliciting actions in underlying layers, which may lead to layering violations. Accurately estimated states lead to appropriate control decisions made by the Adaptation Task [11] in the Adaptor, which are interpreted by the Configurator and delivered to the applications, where proper adaptation actions are performed.

4 Modeling Transmission Tasks

We view each application as a series of connected *tasks*, with each task as a concrete component that performs operations on the input, generates output and consumes resources. We represent the relationships among tasks within an application using a directed acyclic graph, with each directed edge indicating the producer-consumer relationship between output of the upstream task and input of the downstream task. The directed acyclic graph is referred to as the task flow graph [7].

With the above view of an application, a distributed application has one or more *Transmission Tasks* in its task flow graph, which transmits application data between two end systems. Since multiple end systems are involved, the middleware control architecture shown in Figure 1 resides in all end systems. In the Observation Task, we observe the current system states, such as available bandwidth in the Transmission Task. These observations are delivered to the Adaptation Task, which calculates control signals according to the control policy [11]. The distributed view is shown in Figure 2 in the example of a client-server based application.

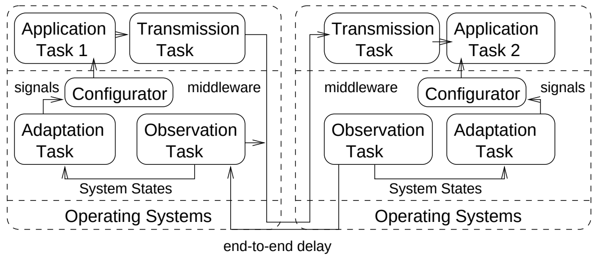


Figure 2: The Middleware Control Architecture with a Distributed Application

The control algorithms introduced in our previous work

[11] apply the control theory in the practice of calculating control signals in the Adaptation Tasks. In the PID control algorithm introduced as an example, we were able to prove that, if priority weights are given to each task, the system fairly allocates resources among competing tasks according to the weighted max-min fairness property. We also proved that the system converges to equilibrium, and stability of control is preserved around a local neighborhood. With an appropriate model for the transmission task, we can extend this work to apply to a distributed environment, with the presence of significant end-to-end delays.

4.1 State Observation in the Transmission Task

The accuracy of control signals calculated by the Adaptation Task relies on precise observations of system states. However, in the distributed environment where observing system states in a Transmission Task is necessary, end-to-end propagation delay poses serious difficulties to observe and capture such information.

An important state to observe is available bandwidth within the Transmission Task T_i, with i being an index in the set of tasks within the application. We take a client-server application as an example, and assume that the Observation Task located on the client can observe the number of received data units[1] during the time $[k, k+1]$ (k being discrete time instants), $z_i(k)$. In reality, we assume that $y_i(k)$ is the number of data units actually received during $[k, k+1]$. On the server, the actual number of data units sent by T_i during $[k, k+1]$, denoted by $u_i(k)$, is controlled by the the Adaptation Task. Finally, $x_i(k)$ is the number of data units *in flight* in the Transmission Task T_i. Note that the Transmission Task T_i itself is distributed on both client and server side, therefore, both $x_i(k)$ and $y_i(k)$ are internal states in T_i, while $y_i(k)$ is also the output of T_i. The above scenario is illustrated in Figure 3.

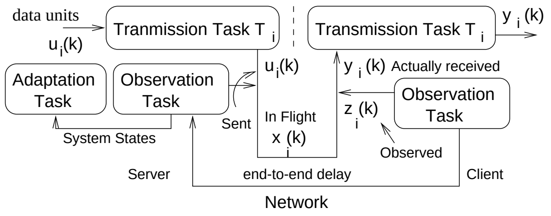


Figure 3: States in the Transmission Task

The challenge is the following. Since the middleware control architectures are situated in different end systems, if end-to-end delays are present, at any particular time instant k, the server Observation Task can only obtain previously observed states by the client Observation Task, with the lag equivalent to the end-to-end delay. This calls for an estimation algorithm in the server Observation Task to compensate the observation error, and to predict the states for the current time instant k.

[1]Data units are defined based on application-specific semantics. For example, in a video-on-demand application, a data unit may be defined as a video frame.

4.2 A Linear Model for the Transmission Task

As a preparation for later applications of analytical techniques in the optimal estimation theory to estimate system states in the distributed Transmission Task, we present a precise analytical model to characterize the internal dynamics of the Transmission Task T_i.

4.2.1 Abstract Model for A Generic Application Task

In order to control any Application Task, we identify several key parameters in this task, referred to as *Task States*. If we use a vector $\mathbf{x}$ denotes task states, $\mathbf{u}$ denotes input to the task, $\mathbf{y}$ denotes task output, $\mathbf{w}$ denotes system noise within the task, $\mathbf{z}$ denotes observation, and $\mathbf{v}$ denotes observation error, we examine a linear and discrete-time model described by the following form:

$$\mathbf{x}(k) = \mathbf{\Phi}\mathbf{x}(k-1) + \mathbf{\Gamma}\mathbf{u}(k-1) + \mathbf{w}(k-1) \quad (1)$$

$$\mathbf{y}(k) = \mathbf{H}\mathbf{x}(k) \quad (2)$$

$$\mathbf{z}(k) = \mathbf{y}(k) + \mathbf{v}(k) \quad (3)$$

where $k = 1$ to k_{max}, and $\mathbf{\Phi}$, $\mathbf{\Gamma}$, and $\mathbf{H}$ are known transition matrices without an error. In later discussions, we develop a concrete analytical model based on the above generic linear model, which is frequently used within the state space approach in control systems.

4.2.2 A Concrete Model for the Transmission Task

In order to develop a concrete model for the distributed Transmission Task T_i, we consider two types of noises in the system. First, the data units in flight from server to client, $x_i(k)$, may suffer from random and unpredictable variations and disturbances $w_i^x(k)$, caused either by physical unstable conditions (in the case of wireless links) or statistical multiplexing of network connections. Consequently, the received quantity of data units during $[k, k+1]$, $y_i(k)$, may also suffer from random disturbances $w_i^y(k)$. These are obviously *system noises* caused by external dynamics in the Transmission Task. Second, the Observation Task itself is also subject to random errors, which can be characterized as the *observation noise* $v_i(k)$. Assume that the observed value is $z_i(k)$, we have

$$z_i(k) = y_i(k) + v_i(k) \tag{4}$$

In Equation (1), $\mathbf{u}(k)$ is actually a scalar $u_i(k)$. Following the analogy of Equation (4) with (2) and (3) combined, we have $\mathbf{y}(k) = y_i(k)$, and $\mathbf{x}(k)$ contains $y_i(k)$. We thus can compute $\mathbf{H}$ based on Equations (2) and (3):

$$\mathbf{z}(k) = z_i(k) = [1 \quad 0] \begin{bmatrix} y_i(k) \\ x_i(k) \end{bmatrix} + v_i(k) \tag{5}$$

and

$$\mathbf{H} = [1 \quad 0]\,, \mathbf{x}(k) = \begin{bmatrix} y_i(k) \\ x_i(k) \end{bmatrix} \text{ and } \mathbf{v}(k) = v_i(k) \tag{6}$$

In addition, from the definition of $x_i(k)$, $y_i(k)$, $w_i^x(k)$ and $w_i^y(k)$, we have

$$x_i(k) = x_i(k-1) - y_i(k-1) + u_i(k-1) + w_i^x(k-1) \tag{7}$$

$$y_i(k) = y_i(k-1) + w_i^y(k-1) \tag{8}$$

It follows from Equations (7), (8), (1) and (6) that

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}, \mathbf{\Gamma} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ and } \mathbf{w}(k) = \begin{bmatrix} w_i^y(k) \\ w_i^x(k) \end{bmatrix} \tag{9}$$

This concludes the state space representation of the linear model for Transmission Task T_i.

4.3 Extending Control Algorithms to Distributed Environment

In our previous work [11], a PID[2] control algorithm was given, and a weighted max-min fairness property was proved. A prerequisite for the fairness property to hold is that the Observation Task has the ability to observe complete global system states. For example, if the resource being observed is CPU usage in the same end system, global states can be observed for all competing tasks.

However, this is normally not the case in a distributed environment, when observing task states within the Transmission Task. Since the Observation Tasks reside as middleware components in the end system, it has no ability to obtain states corresponding to all other connections sharing the network. In such cases, the only observable states are the parameters used or allocated by the Transmission Task itself, such as occupied bandwidth. Therefore, while the control algorithm still adapts to variations in resource availability and shows stability and convergence properties, it lacks crucial observations to guarantee any global fairness properties.

In distributed applications, the PID control algorithm adopted in the Adaptation Task may be modified as follows:

$$u_i(k) = u_i(k-1) + \alpha[x_i^c(k) - x_i(k)] + \\ \beta\{[x_i^c(k) - x_i(k)] - [x_i^c(k-1) - x_i(k-1)]\} \tag{10}$$

where x_i^c is the reference value expected at equilibrium, $u_i(k)$ is the actual number of data units sent by T_i, $x_i(k)$ is the number of data units in transit from server to client, and α and β are configurable scaling factors. The stability and convergence proofs still hold as in the previous work.

However, with the presence of end-to-end delays, it is inherently not trivial to accurately estimate $x_i(k)$ directly at the server, since the number of data units in transit in $[k, k+1]$ is not directly observable.[3] $y_i(k)$, the number of data units received, is directly observable, but *only* at the client side. Therefore, at the server side, the available values for $u_i(k)$ computation in the control algorithm (Equation (10)) are imprecise, as the $y_i(k)$ values (needed for $x_i(k)$ computation) previously observed by the client are received by the server only after an end-to-end delay from the time of observation. This leads us to the following approach. Instead of deriving $y_i(k)$ using only the available observed values transmitted from the client to the server with an end-to-end delay, we will adopt optimal state prediction techniques to estimate $y_i(k)$ at the current time instant, which forms discussions in the next section.

5 Optimal Prediction of Task States In Transmission Tasks

In this section, we present an optimal prediction approach to optimally predict the current task states in the Transmission Task, based on observed task states in previous time instants before the end-to-end delay. The optimal prediction algorithms are implemented in the server Observation Task, while the actual observation is made in the client Observation Task. Optimality in the prediction algorithms guarantees that the relative error between the prediction and actual values of task states is minimized,

[2]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.

[3]Equation (7) is part of the linear model of the Transmission Task, but it can not be easily utilized for the estimation of $x_i(k)$ since it is not observable directly.

i.e., a *best possible guess* is obtained. We adopt the optimal control and estimation theory [12] to develop the proposed algorithms, and associate the theoretical solutions with the practical cases in complex distributed applications, focusing the Transmission Task.

5.1 The Need for Prediction

It is obvious from Equation (4) that the client Observation Task is able to observe $y_i(k)$ as $z_i(k)$, with an observation noise $v_i(k)$. However, from the control algorithm expressed in Equation (10), we note that $x_i(k)$ is actually used in the Adaptation Task. In order to derive $x_i(k)$ on the server from the observed values $z_i(k)$ on the client, we assume that the client acknowledges all received data units to the server, and that the server Observation Task has the knowledge of the total number of data units *unacknowledged* at the server up to the time instant k, denoted by $y_i^s(k)$. Then, we have $z_i^s(k) = y_i^s(k) + v_i^s(k)$ as the observed values of $y_i^s(k)$ with an observation noise $v_i^s(k)$. Naturally, $y_i^s(k)$ represents the total number of unacknowledged data units which are either in flight from server to the client, which is $x_i(k)$, or received by the client, but acknowledgments not yet received by the server. We thus have

$$x_i(k) = y_i^s(k) - \sum_{t=k-\lceil \frac{d_i}{t_c} \rceil}^{k-1} y_i(t) \qquad (11)$$

where d_i is the end-to-end transmission delay from client to server, t_c is the sampling time interval between $[k-1, k]$, assuming $d_i \geq t_c$. Ideally, if $y_i(t), \forall t \in [k - \lceil d_i/t_c \rceil, k-1]$ is known, $x_i(k)$ can be computed and then used in the control algorithm of Equation (10). However, the end-to-end delay, represented by d_i, prevents the knowledge of $y_i(t),\ \forall t \in [k - \lceil d_i/t_c \rceil, k-1]$. The last available observation is $z_i(k - \lceil d_i/t_c \rceil)$. The need of predicting these values of $y_i(t)$ in the server Observation Task before calculating $x_i(k)$ arises from this lack of knowledge. Figure 4 illustrates the above scenario.

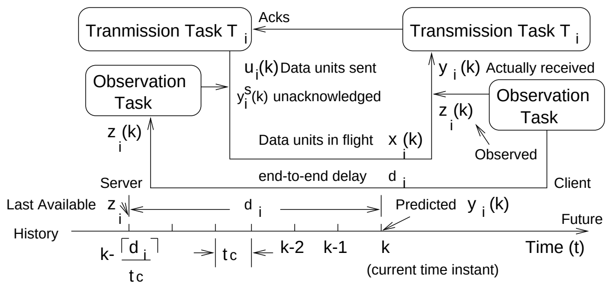


Figure 4: State Prediction in Transmission Tasks

We use $\hat{y}_i(k)$ to denote the predicted values of $y_i(k)$. Assuming that $\hat{y}_i(k)$ is already obtained optimally, we can estimate $x_i(k)$ by the following Equation:

$$x_i(k) = z_i^s(k) - \sum_{t=k-\lceil \frac{d_i}{t_c} \rceil}^{k-1} \hat{y}_i(t) \qquad (12)$$

The problem then shifts to the development of appropriate mechanisms to obtain $\hat{y}_i(k)$.

5.2 Mechanisms for Optimal Prediction

5.2.1 Definition of Optimality

Based on the Separation Principle [12], for a linear stochastic system where an observer is used to estimate the system state, the parameters for the observer and controller are determined separately. Informally, this means that we can develop an optimal prediction algorithm for $y_i(t), \forall t \in [k - \lceil \frac{d_i}{t_c} \rceil, k-1]$ in the server Observation Task, while still retaining complete freedom for adopting alternative control algorithms in the server Adaptation Task.

With regards to the prediction accuracy, we prefer to design an *optimal* prediction algorithm that minimizes the sum of squared errors between the predictions and values being estimated, i.e., a least-squares estimate. More precisely, if $\epsilon(k) = y_i(k) - \hat{y}_i(k)$, where $\hat{y}_i(k)$ is the predicted values of $y_i(k)$ at time k, we try to minimize the quadratic error cost function $J(\mathbf{y}) = J(y_i(k)) = \frac{1}{2}\epsilon^2(k)$. The optimal prediction approach, e.g., Kalman Filter, presented in this section is designed to minimize $J(\mathbf{y})$.

5.2.2 Requirements of an Optimal Solution

The optimal prediction problem is generally hard if the linear stochastic system is in its generic form. However, it is proved in optimal control theory [13] that simplified prediction algorithms can be adopted as an optimal solution in a special case, with two prerequisites. First, the system random disturbances $\mathbf{w}(k)$ and observation noises $\mathbf{v}(k)$ are uncorrelated white Gaussian-Markov sequences with zero mean. This can be interpreted that: (1) random vectors in the same stochastic sequence are independent of each other; (2) they can be uniquely characterized by a joint Gaussian probability density function; (3) this density function has zero mean expectation, and (4) random vectors in different stochastic sequences are uncorrelated with each other. Second, the initial system state vector $\mathbf{x}(0)$ is also a Gaussian random vector with zero means.

We assert that the system states and noises in the Transmission Task T_i observe such nature. This assertion is based on the following characteristics.

(1) The states in the Observation Task and the Transmission Task are not correlated, since the Observation Tasks are implemented separately in the middleware level, while the Transmission Task is part of the application. This observation guarantees that the observation noise $\mathbf{v}(k)$ and the system noise $\mathbf{w}(k)$ are uncorrelated.

(2) Within the transmission path, when the number of simultaneous connections N sharing the same physical communication channel (statistical multiplexing in intermediate switches) is large, we expect that in a time interval of length t_c, the changes in N is very small compared to N itself. This leads to the fact that changes in $x_i(k)$ due to activities of other connections will be small. Thus, when we model $x_i(k)$ as a process given by Equation (7) $x_i(k) = x_i(k-1) - y_i(k-1) + u_i(k-1) + w_i^x(k-1)$, the term $w_i^x(k-1)$, which is the dynamic disturbances caused by activities of other connections, can be modeled as a zero-mean Gaussian white noise [9]. Even though when $x_i(k)$ is small and the connection is in a starting stage, the possibility of an increase is larger than a decrease, this assumption of zero mean is justifiable when $x_i(k)$ is sufficiently far from 0. The same observation also applies to $y_i(k)$ and $w_i^y(k)$. This concludes that the random noise $\mathbf{w}_i(k)$ is a white Gaussian-Markov sequence with zero mean.

We conclude that random disturbances of the Transmission Task satisfy the requirements of applying the simplified prediction algorithms, such as the Kalman Filter prediction algorithm that follows.

5.2.3 Parameters in the Kalman Filter

We now apply the frequently used optimal estimation algorithm, Kalman Filter, to solve the prediction problem of Task States in the Transmission Task.

Equation (9) shows that both $\mathbf{\Phi}(k-1)$ and $\mathbf{\Gamma}(k-1)$ in Equation (1) are constants without error. In addition, $\mathbf{u}(k-1)$ are also known in the server Observation Task without error in the interval $0 \leq k \leq k_{max}$. We introduce the definition of the following terms:

(1) The *expected values*, or *expectations*, $E(\mathbf{x})$ of any random vector $\mathbf{x}$ is defined as the mean vector of $\mathbf{x}$. Formally, $E(\mathbf{x}) = [E(y_i(k)), E(x_i(k))]^T$, where $E(x)$ for a random variable x is defined as $E(x) = \int_{-\infty}^{\infty} xp(x)dx$, if $p(x)$ is the probability density function of x.

(2) The *error covariance matrix* $\mathbf{P}$ of $\mathbf{x}$ in the Transmission Task is defined as:

$$\mathbf{P}(k) = E[[\mathbf{x}(k) - \hat{\mathbf{x}}(k)][\mathbf{x}(k) - \hat{\mathbf{x}}(k)]^T] \tag{13}$$

(3) The dynamic system disturbance $\mathbf{w}$ is a white, zero-mean Gaussian random sequence showing the following properties, where $\mathbf{Q}(k)$ is the system noise covariance matrix:

$$E[\mathbf{w}(k)] = 0 \tag{14}$$

$$\mathbf{Q}(k) = E[\mathbf{w}(k)\mathbf{w}(k))^T] \tag{15}$$

$$E[\mathbf{w}(k)\mathbf{w}(j))^T] = 0, (j \neq k) \tag{16}$$

(4) Similarly, in Equation (2) and (3), $\mathbf{H}$ is a constant and the observation noise is modeled as a white, zero-mean Gaussian random sequence that is uncorrelated with the system disturbance:

$$E[\mathbf{v}(k)] = 0 \tag{17}$$

$$\mathbf{R}(k) = E[\mathbf{v}(k)\mathbf{v}(k))^T] \tag{18}$$

$$E[\mathbf{v}(k)\mathbf{v}(j))^T] = 0, (j \neq k) \tag{19}$$

$$E[\mathbf{n}(k)\mathbf{w}(j))^T] = 0(\text{all } j \text{ and } k) \tag{20}$$

where $\mathbf{R}(k)$ is the observation noise covariance matrix. According to Equation (6), $\mathbf{v}(k)$ is a scalar $v_i(k)$, it follows that $\mathbf{R}(k)$ is the variance of $v_i(k)$, $var(v_i(k)) = \sigma^2$, when $v_i(k)$ is a Gaussian distribution (m, σ).

In practice, it is necessary to determine $\mathbf{Q}(k)$ and $\mathbf{R}(k)$ offline. These covariance matrices indicate the level of confidence in the system model and observations, respectively. If one were to increase $\mathbf{Q}$, this would indicate that stronger noises are driving the dynamics. Consequently, the rate of growth of the elements of the error covariance matrix $\mathbf{P}(k)$ will also increase, which increases the filter gain $\mathbf{K}(k)$ (refer to Appendix for formal presentations), thus weighing the measurements more heavily. Therefore, by increasing $\mathbf{Q}$, we in effect put less confidence in the system model. Similarly, increasing $\mathbf{R}$ indicates that the observations are subject to a stronger corruptive noise, and therefore should be weighed less by Kalman Filter.

5.2.4 Operations of Kalman Filter

Based on these definitions, Kalman Filter operates recursively in a *predict-update* manner as informally described in the following phases. Refer to Appendix for the formal equations.

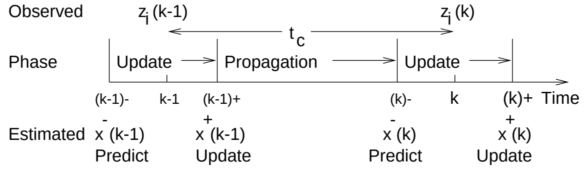


Figure 5: The Kalman Filter in operation

(1) *Prediction Phase* occurs at time k^-, that is, before observations are made at time k. State predictions $\hat{\mathbf{x}}^-(k)$

are made for states $\mathbf{x}^{-}(k)$, and error covariance predictions $\mathbf{P}^{-}(k)$ is also made.

(2) *Kalman Filter Gain Computation Phase* occurs between k^{-} and k^{+}, which is the time after k. The Kalman Filter gain matrix $\mathbf{K}(k)$ is computed to be used later in the Update Phase.

(3) *Update Phase* occurs at time k^{+}. The Kalman Filter gain matrix $\mathbf{K}(k)$ is used along with the new observation $\mathbf{z}(k)$. The error covariance matrix $\mathbf{P}^{+}(k)$ is also updated from previously predicted $\mathbf{P}^{-}(k)$ in the Prediction Phase.

These phases are executed repetitively till the time when the latest observation is available from the client Observation Task. After this time instant, we can deploy a linear-optimal predictor to predict the state and its error covariance on the basis of all the information that is available without observation. Denoting the time of latest available observation on the client as $k - \lceil \frac{d_i}{t_c} \rceil$ for Transmission Task T_i, where k is the present time instant on server, the linear-optimal predictor starts with the latest state estimate update phase using Kalman Filter, i.e., $\hat{\mathbf{x}}^{+}(k - \lceil \frac{d_i}{t_c} \rceil)$, and then recursively applies the state prediction phase to calculate $\hat{\mathbf{x}}(t) = \hat{\mathbf{x}}^{-}(t), \ \forall t \in [k - \lceil \frac{d_i}{t_c} \rceil, k-1]$. According to Equation (6), we have the following for Transmission Task T_i:

$$\hat{y}_i(t) = \mathbf{H}\hat{\mathbf{x}}(t), \ \forall t \in [k - \lceil \frac{d_i}{t_c} \rceil, k-1] \qquad (21)$$

Equation (21) concludes our prediction mechanisms utilizing the Kalman Filter. When the estimated values of $y_i(k)$ are applied to Equation (11), $x_i(k)$ can be obtained and thus applied to the control algorithm in the Adaptation Task as presented in Equation (10).

6 Experiments with Visual Tracking

Based on the algorithms developed in previous sections, we have implemented a preliminary middleware control architecture to control a client-server based visual tracking application, adopting Kalman Filter as a optimal prediction mechanism in the server Observation Task, with the presence of end-to-end delay.

6.1 Overview of the Visual Tracking Application

We use a client-server based visual tracking application as an example of complex applications to evaluate our approach. Figure 6 shows an overview of its architecture. Based on the original XVision [6] project in Unix, we have completed the implementation of the client-server based visual tracking application on the Windows NT 4.0 platform in Visual C++ 5.0, using Windows Sockets 2 API for the network transmission.

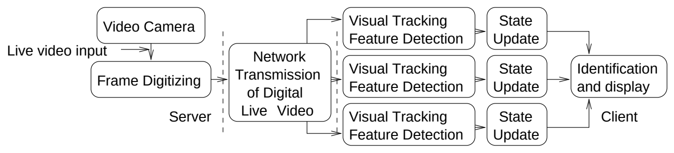

Video Camera
Live video input
Frame Digitizing
Server
Network Transmission of Digital Live Video
Visual Tracking Feature Detection
Visual Tracking Feature Detection
Visual Tracking Feature Detection
State Update
State Update
State Update
Identification and display
Client

Figure 6: The Client-Server Visual Tracking Application

We have implemented the Adaptors shown in Figure 1 in C++ and Java as middleware components, including both the Adaptation Task, using Equation (10) as the control algorithm, and the Observation Task, using the Kalman Filter as the optimal state predictor. All middleware components interact among one another and with the application using CORBA. We use ORBacus 2.0.4 [8] as our CORBA implementation. Figure 7 shows the main tracking window of the application with three trackers (SSD, line and corner) running simultaneously. By enabling adaptation, the primary QoS parameter that we focus on is the *tracking precision*. For quality assurance of this parameter, other QoS parameters such as the image size can be sacrificed as a tradeoff.

We tested our system in a varying network environment, in order to experiment application-level adaptation on transmission bandwidth requirements. In order to simulate bandwidth fluctuations in a typical distributed environment over wide-area networks, we have also implemented a simple network simulator, which simulates packet delay through a transmission path of multiple network routers, each of them implementing the FIFO scheduling algorithm. Because of the bursty nature of cross traffic, throughput fluctuations may occur at various times over the connection.

For the purpose of repeating the same set of experiments and for measurements of *tracking precision*, we use a computer generated image sequence, in which the object moves at fixed speed and path. For the experimental results shown in Figure 8, the moving speed of the rectangle is set at a constant 3 screen pixels per second continuously. In addition, we assume there are no other CPU intensive process running in the background on the same platform. This is for the purpose of separating the experiments on bandwidth requirements from those on CPU requirements.

In Figure 8, the three graphs on the left are in the case without any adaptation. The three graphs on the right are in the case with adaptation support from the middleware framework, with integrated optimal prediction mechanisms in action in the server Observation Task to

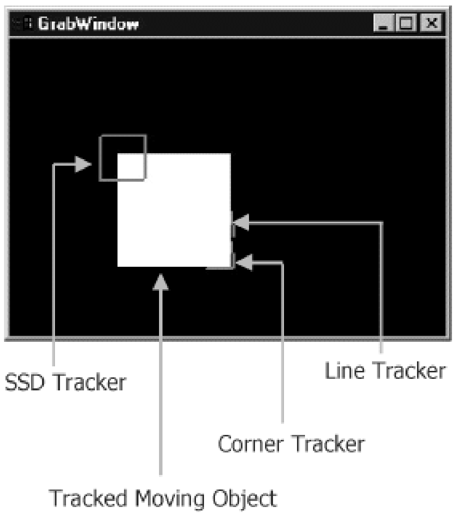
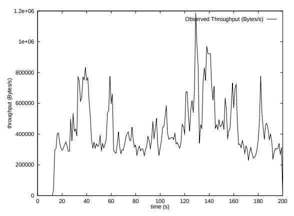


Figure 7: Visual Tracking on Client: The Main Tracking Window with three trackers

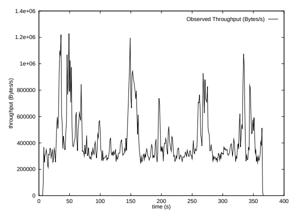
overcome end-to-end delay, as well as the PID control algorithm in the Adaptation Task. We can observe that by changing the frame size of the visual tracking application, the tracking precision will be preserved without any tracking error at all times during the connection. In contrast, without any adaptation, when the network throughput degrades to a certain degree, the tracking algorithm is not able to keep track of the object, the error accumulates rapidly verifying that the tracking algorithm loses the object. This prove-of-concept system proves that the approach we have taken is effective in preserving tracking precision in a distributed environment with fluctuating bandwidth and significant end-to-end delay between the client and server.

7 Conclusions

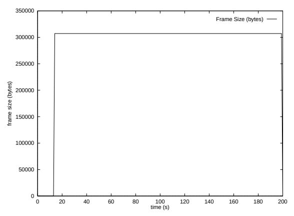
In this work, we focus on the scenario when flexible applications, such as client-server based visual tracking, need to adjust themselves to adapt to resource variations and preserve critical QoS parameters, such as tracking precision. We have extended the Task Control Model in a distributed environment, modeled the Transmission Task in a state-space representation, presented an optimal state prediction mechanism to overcome end-to-end delay in distributed state observations, and presented preliminary results with our client-server visual tracking experiments to verify our approach. The optimal prediction mechanism proposed in this paper is integrated in the server Observation Task, as middleware components and part of the middleware control architecture in a larger scale. Ongoing and future work involves application-configurable adaptation and estimation algorithms via a configuration scripting language, as well as QoS assurance and adaptation in a multicasting environment.



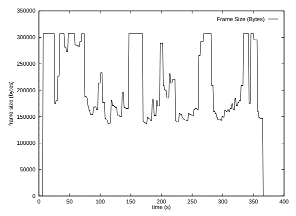
(a) Observed Throughput



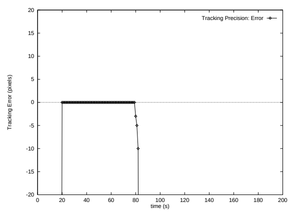
(b) Observed Throughput



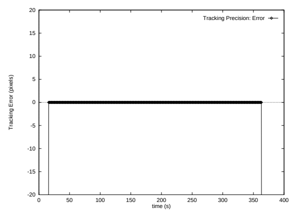
(c) Application-level Adaptation: Original Image Size



(d) Application-level Adaptation: Chopped Image Size



(e) Tracking Precision



(f) Tracking Precision

Figure 8 (Left): Experiments without adaptation support

Figure 8 (Right): Experiments with adaptation support

Figure 8: Experiments with the Client-Server Based Visual Tracking Application

Appendix: The Kalman Filter

In the following equations, we distinguish between estimates made before and after the updates. $\hat{\mathbf{x}}^-(k)$ is the state estimate that results from the prediction equation (22) alone (i.e. *before* the observations are considered), and $\hat{\mathbf{x}}^+(k)$ is the corrected state estimate that accounts for the observation made. $\mathbf{P}^-(k)$ and $\mathbf{P}^+(k)$ are defined similarly. For completeness, the initial conditions are $\hat{\mathbf{x}}^+(0)$ and $\mathbf{P}^+(0)$.

- State Estimate Prediction Phase:
$$\hat{\mathbf{x}}^{-}(k) = \mathbf{\Phi}\hat{\mathbf{x}}^{+}(k-1) + \mathbf{\Gamma}\mathbf{u}(k-1) \tag{22}$$
$$\mathbf{P}^{-}(k) = \mathbf{\Phi}\mathbf{P}^{+}(k-1)\mathbf{\Phi}^{T} + \mathbf{Q}(k-1) \tag{23}$$
- Kalman Filter Gain Computation Phase:
$$\mathbf{K}(k) = \mathbf{P}^{-}(k)\mathbf{H}^{T}[\mathbf{H}\mathbf{P}^{-}(k)\mathbf{H}^{T} + \mathbf{R}(k)]^{-1} \tag{24}$$
- Update Phase:
$$\hat{\mathbf{x}}^{+}(k) = \hat{\mathbf{x}}^{-}(k) + \mathbf{K}(k)[\mathbf{z}(k) - \mathbf{H}\hat{\mathbf{x}}^{-}(k)] \tag{25}$$
$$\mathbf{P}^{+}(k) = [\mathbf{P}^{-}(k)^{-1} + \mathbf{H}^{T}\mathbf{R}(k)^{-1}\mathbf{H}(k)]^{-1} \tag{26}$$

References

[1] J. Bolliger and T. Gross. A Framework-Based Approach to the Development of Network-Aware Applications. *IEEE Transactions on Software Engineering*, 1998.

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

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

[5] A. Hafid and G. Bochmann. Quality of Service Adaptation in Distributed Multimedia Applications. *ACM Springer-Verlag Multimedia Systems Journal*, 6, 1998.

[6] G. Hager and K. Toyama. The XVision System: A General-Purpose Substrate for Portable Real-Time Vision Applications. *Computer Vision and Image Understanding*, 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] Object Oriented Concepts Inc. ORBacus for C++ and Java. *ftp://ftp.ooc.com/pub/OB/3.1/OB-3.1b1.pdf*, 1998.

[9] S. Keshav. A Control-Theoretic Approach to Flow Control. *Proceedings of ACM SIGCOMM 91*, 1991.

[10] A. Kolarov, A. Atai, and J. Hui. Application of Kalman Filter in High-Speed Networks. *Proceedings of IEEE GLOBECOM 94*, 1994.

[11] B. Li and K. Nahrstedt. A Control Theoretical Model for Quality of Service Adaptations. *Proceedings of Sixth International Workshop on Quality of Service*, 1998.

[12] J. Meditch. *Stochastic Optimal Linear Estimation and Control*. McGraw-Hill, 1969.

[13] R. Stengel. *Optimal Control and Estimation*. Dover Publications, 1994.