Model-Predictive Controllers for Performance Management of Composable Conveyor System

Shunxing Bao,
Aniruddha Gokhale
EECS, Vanderbilt Univ
Nashville, TN, USA
Email: {shunxing.bao,a.gokhale}@vanderbilt.edu

Sherif Abdelwahed
ECE, Mississippi State Univ
Starkville, MS, USA
Email: sherif@ece.msstate.edu

Shivakumar Sastry
ECE, Univ of Akron
Akron, OH, USA
Email: ssastry@akron.edu

***Abstract*—The increasing complexity of Cyber Physical Systems (CPS) found in a variety of domains, and the unforeseen fluctuations in operating conditions caused by the open nature of these systems makes it significantly challenging to appropriately configure and adapt the operating parameters of CPS to ensure reliability and desired quality-of-service (QoS). Material handling and packaging is one such domain experiencing these difficulties where the plants employ composable and reconfigurable conveyor systems that enable adaption to changing demands and business processes. Plant operators, however, face a daunting task of ensuring that the system throughput is maximized in the face of fluctuating workloads, while at the same time operation costs incurred in operating the plant are minimized. Plant operators require that these capabilities continue to be available even after the conveyor layouts change. To address these duo of challenges, this paper develops and compares five different model-predictive controller designs all of which have an inherent quality of being composable. The five controller designs comprise a single-level, completely decentralized controller with two alternatives on how belt speed is estimated; a centralized controller; a two-level controller; and a time-abstracted centralized controller. The composable nature of all the controller designs lends towards addressing the situation when the layouts change. An evaluation of our controller designs in the context of a case study is presented comparing to a baseline case with a static system configuration.**

***Keywords*—Model-predictive Control, Composable conveyors, CPS**

I. Introduction

Material handling and packaging systems are excellent examples of widely-used engineered systems that embody many characteristics of cyber physical systems (CPS). Such systems have applications in warehouses, manufacturing plants, package sorting facilities (e.g., FedEx and UPS), and even in front-line logistics for military deployments. The composable and reconfigurable nature of these systems make them attractive in application scenarios because they can adapt to changing processes and environmental demands. These systems are also attractive for the study of CPS principles because the techniques designed for such systems to enable composability, scalability and performance can be applied to a larger spectrum of CPS.

The composable conveyor systems (CCS) [1] we consider comprises multiple instances of two kinds of units — namely, Segment and Turn. A Segment has a belt whose speed and direction can be controlled. A Turn is a reconfigurable merger/splitter unit that can be juxtaposed with up to four Segment instances. These conveyor systems can be controlled by regulating the speed of the belt on the individual units. The load in the system, *i.e.*, the number of packages handled by different units in the system can be regulated by dynamically routing the packages over different end-to-end paths in the system. Configuring and synchronizing the speeds of the units is unlikely to be scalable if the decisions are made in a central location. The speeds must also be adjusted in response to variabilities in the arrival rates of the packages at the different inputs to the system. Further, the speeds and routes must be adjusted in response to failures and repairs of conveyor units that can occur throughout the lifecycle of the system.

Many automation systems are becoming increasingly open and more connected to the supply chain [2]. CCS is a representative example in which the impact of such connectivity can be studied. Workloads in the system can fluctuate substantially; *e.g.*, holiday season may experience dramatic increase in packages that must be sorted and shipped to their destinations. In such situations, statically-defined preset speeds for the entities of the CCS may not be sufficient to effectively operate the system. Similarly, unforeseen disruptions in the supply chain can make any fixed schedule ineffective. Plant operators are thus faced with the task of addressing two challenges. First, they must be able to dynamically adapt the operation of their plant to maximize the throughput and minimize energy consumption while adapting to the workload fluctuations. Second, such dynamic adaptation capabilities must remain available when the conveyor system topology or layout of the plant floor undergoes change due to business-specific and other logistical reasons.

To address such challenges, we present the design and evaluation of five model-predictive control approaches as follows:

1) Our first approach relies on a single level of distributed control where individual controllers reside in each unit of the CCS. Each controller predicts the incoming workload and adjusts its local speed.
2) Our second approach is based on a centralized management of individual controllers in the first approach. predicts and makes decisions for all the belts in the system.
3) Our third approach presents a decentralized design that uses local controllers for each unit as in approach #1, however, these controllers predict the arrival rate taking into account the speed of the preceding unit.

4) Our fourth approach provides a hierarchical design in which the first level comprises the above mentioned decentralized control units; however, the individual controllers make short-term predictions of workloads. These controllers are in turn managed by a second level controller that makes longer-term forecasts of expected workloads and fine tunes the performance of the system.
5) Our fifth approach illustrates a time abstract centralized controller where the controller can make both short-term and long-term prediction and come up with which speed level that belts should use.

Our prior work for CCS has explored the use of model driven engineering tools to reason about different properties of CCS conveyor layouts prior to their actual deployment [3]. More recently we adapted the classical priority inheritance protocol to resolve priority inversions in CCS [4]. None of these efforts investigated the use of model predictive control. On the other hand, we have designed a model-predictive, two-level controller for the adaptive performance management of computing systems [5], however, this solution was applied to a purely cyber-only system. In the current work we focus on cyber physical systems and account for both the physical dynamics and the cyber interactions of these sysetms. Our two-level model-predictive controller design for a system of CCS is indeed inspired by our work in [5] but its design has a number of differences that are explained later.

This paper makes the following contributions:

- We present the design of five model predictive controllers to manage the performance of CCS.
- Using a case study we compare and contrast the performance of each controller, and compare their performance to CCS with static configuration.

The rest of the paper is organized as follows: Section II describes related work comparing them with our work; Section III provides a background on model-predictive control; Section IV describes the design of the controllers; Section V describes results of our simulations comparing the performance of the two controllers with the baseline; and finally Section VI offers concluding remarks alluding to opportunities for future work.

II. Related Work

In this section we survey some recent efforts developing controller designs for CPS focusing mostly on related use cases and with a focus similar to ours. Various other related efforts also exist but we have not described them here.

A recent synergistic effort [6] is concerned with energy efficiency of the conveyors. In this work the authors compare an open-loop optimal control approach with a model-predictive closed-loop control. In their case, the authors tested their system in a coal processing plant. They conclude that the model-predictive approach was better able to handle fluctuations and uncertainties in workload prediction, however, in their case rather than incoming workloads, their problem related to predicting the consumption of coal which was driven by the quality of coal.

Another use case involving coal is reported in [7]. In this work the authors argue that one must also pay attention to system constraints and external constraints such as tariffs and storage capacities. Thus, this work seeks to find the optimal scheduling of belt conveyors to improve energy efficiency by taking into account all the additional constraints.

Fuzzy logic control is used as a control strategy for the system of belt conveyors with adjustable speed drives [8]. In this work the use case is an open pit mine where conveyors are used to transport the mined material. The authors in this work refer to the long distances covered by the conveyors which require intermediate stations. To make the system efficient, the authors have developed fuzzy logic controller.

Centralized monitoring and control is necessary to reliably and safely operate the belt conveyor. The authors in [9] take the belt conveyor for coal mine as the background, and design the monitoring system based on PLC technology according to a distributed controller structure model. In summary, each substation in mining area can independently realize controlling a single belt conveyor. Centralized control center can manage them centralized.

Airport baggage handling is a representative package system for centralized control. Authors in [10] propose a multi-agent control approach for a baggage handling system (BHS) using IEC 61499 Function Blocks. In particular, the work focuses on demonstrating a decentralized control system that is scalable, reconfigurable, and fault tolerant. The authors demonstrate the effectiveness of the agent-based control system and present a utility for real-time viewing of these systems.

A flexible robotic assembly system decentralized architecture is presented in [11]. The system consists of conventional manipulators and a belt conveyor. It is designed to achieve high reconfigurability so that it can adapt to changes in manufacturing environment; a new robot can be easily installed to the system and execute assembly tasks immediately with the help of other devices. For easier reconfiguration, a semi-automated calibration method for positions of newly installed robots is integrated into the system. The authors demonstrate the system will not be optimal on reconfigurability until it has effective task allocation.

Small scale, multi-directional conveyor modules which are smaller than the goods to be transported provide outstanding flexibility. The authors in [12] present a decentralized, self-organizing control for these modules based on cellular automata. To fulfill the transportation task, each module participates in the solution process. The presented control makes the system easy to reconfigure and scalable. The authors illustrated that a complex system behavior emerges from simple rules defining the cellular automaton.

III. Background on Control-based Performance Management

Our five controller designs are based model-predictive concepts with Limited Lookahead Control (LLC) [5]. This section first briefly reviews some key LLC concepts. Figure 1 shows the key components of a self-managing computing system: 1) the system model and its QoS specification, 2) the environment-input forecaster, and 3) the controller.

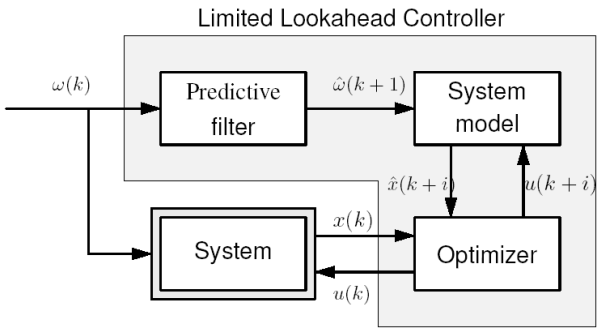

Fig. 1. Key components of a self-managing computing system.

A. Forecasting Environment Inputs

In systems that operate in open and dynamic environments, the corresponding inputs to the controller are generated by external sources whose behavior typically cannot be controlled, e.g., web-page requests made to a server by Internet clients. It such systems, workloads of interest show strong and pronounced time-of-day variations [13], [14], [15], and that key workload characteristics such as request arrival rates can change quite significantly and quickly—usually in the order of a few minutes. In most situations, however, such workload variations can be estimated effectively using well-known forecasting techniques such as the Box-Jenkins ARIMA modeling approach [16] and Kalman filters [17]. A forecasting model is typically obtained via analysis or simulation of relevant parameters of the underlying system environment and has the following form:

$$\hat{\omega}(k) = \phi_k(\underline{\omega}(k-1, r)) \tag{1}$$

where $\hat{\omega}(k)$ denotes the estimated value at time k and $\underline{\omega}(k-1, r)$ is the set of r previously observed environment vectors $\{\omega(k-1), \ldots, \omega(k-r-1)\}$. The other estimation parameters—for instance, the covariance matrix in the Kalman filter—are assumed to be embedded in the model ϕ_k. These parameters are typically obtained by training ϕ_k using test data representative of actual values observed in the field. We also assume that the estimation error is bounded and known with a certain probability distribution. Therefore, we can write, $\omega(k) = \phi_k(\underline{\omega}(k-1, n)) + e(k)$ where $e(k) \in E$ is a bounded random variable reflecting the effect of the estimation error.

From the implementation point of view, environment inputs are handled by a prediction module that continuously samples environment inputs and estimates their future values. The predictor module can be instantiated with different forecasting techniques and can be trained offline with representative input data. Signal and parameter estimators may also be added to extract information about, and build an accurate model of the operating environment.

B. System Model, Performance Specification, and Constraints

As noted in Section I, the control approach proposed in this paper targets a general class of cyber-physical systems with finite control-input set. The following discrete-time equation describes the general dynamics of such a system:

$$x(k+1) = f(x(k), u(k), \omega(k)) \tag{2}$$

where $x(k) \in \mathbb{R}^n$ is the system state at time step k, and $u(k) \in U \subset \mathbb{R}^m$ and $\omega(k) \in \Omega \subset \mathbb{R}^r$ denote the control inputs and environment parameters at time k, respectively. The system model f captures the relationship between the observed system parameters, particularly those relevant to the QoS specifications, and the control inputs that adjust these parameters. This model could be in the form of difference equations for simple systems, and in the form of an approximation structure such as a neural network for more complex systems whose dynamics cannot be easily described from first principles.

Since the current value of $\omega(k)$ cannot be measured until the next sampling instant, the system dynamics can only be captured using a model with uncertain parameters, as follows:

$$\hat{x}(k+1) = f(x(k), u(k), \hat{\omega}(k))$$

This estimated value of x is the one used by the controller to evaluate applicable control options. From the dynamic point of view, we can rewrite the system model as follows,

$$x(k+1) = f(x(k), u(k), \phi_k(\underline{\omega}(k-1, r)), e(k))$$

In the above equation, $e(k)$ is the only stochastic variable. In analyzing the feasibility of the predictive control approach, one needs to consider the value of e leading to the maximum deviation from the objective state. We will introduce an algorithm to check the feasibility of the control approach for a bounded error domain E.

1) Performance Specifications: Computing systems must achieve specific QoS goals while satisfying certain operating constraints. A basic control action in such systems is *set-point regulation* where key operating parameters must be maintained at a specified level or follow a certain trajectory. The controller, therefore, aims to drive the system to within a close neighborhood of the desired operating state $x^* \in X$ in finite time and maintain the system there. A general form of such specification can be expressed using the cost function

$$J(x, u) = \|x - x^*\|_P + \|u\|_Q + \|\Delta u\|_R \tag{3}$$

where $\|\cdot\|_A$ is a proper norm with weight A. The above performance measure takes into account the cost of the control inputs themselves and their change. It is also possible to consider transient costs as part of the operating requirements, expressing the fact that certain trajectories towards the desired state are preferred over others in terms of their cost or utility to the system.

2) Operating Constraints: The system must also operate within strict constraints on both the system variables and control inputs. In general, such constraints can be expressed as a feasible domain for the composite space of a set of system variables, possibly including the control inputs themselves. Such operating constraints can be generally captured as $\psi(x) \leq 0$ and $U(x) \subseteq U$ where $U(x)$ denotes the permissible input set in state x and $\psi(x) \leq 0$ defines reachable states.

To summarize, the primary objective of the controller is to drive the computing system to the desired state x^* while minimizing the control and transient costs in "reasonable" time using an admissible trajectory, defined by the constraints $\psi(x) \leq 0$ and $U(x)$, and maintain it close to x^*.

C. Predictive Controller

The basic concept behind predictive control is to solve an optimization problem over a future time horizon, and then roll this horizon forward at regular intervals, re-solving the control problem.

Algorithm 1 The Predictive Control Algorithm: PControl(k)

input: $x(k)$, $\underline{\omega}(k-1, r)$
$s_o := x(k)$
for $i = 0$ **to** $N-1$ **do**
  $\hat{\omega} := \phi_{k+i}(\underline{\omega}(k+i-1, r))$
  $s_{i+1} := \varnothing$
  for all $x \in s_i$, $u \in U(x)$ **do**
    $\hat{x} := f(x, u, \hat{\omega})$
    $s_{i+1} := s_{i+1} \cup \{\hat{x}\}$
    Compute Cost($\hat{x}$) based on $J(\hat{x}, u)$.
  end for
end for
$x_{min} := \arg\min\{\text{Cost}(x) \mid x \in s_N\}$
return $u^*(k) :=$ initial input leading from $x(k)$ to x_{min}

Algorithm 1 shows the details of the predictive control technique. At each time instant k, it accepts the current operating state $x(k)$, and starting from this state, the controller constructs a tree of all possible future states up to the specified prediction depth N. The relevant parameters of the operating environment are first estimated and then the next set of reachable system states (subject to both state and input constraints) are generated by applying all applicable control inputs from the set $U(x)$. The total cost function corresponding to each estimated state is then computed based on the cost function J. The state x_{min} with the minimum cost at the end of the tree is then selected and the first input leading to this state, $u^*(k)$, is applied to the system while the rest are discarded. The above search is repeated at each sampling step.

IV. Model Predictive Controller Designs for CCS

Based on the preliminaries on model predictive control described in Section III, we now describe the design of five different controllers we built for the composable conveyor systems.

A. System Model of CCS

The composable conveyor systems comprise four main components: The input bins are the places that receive the packages, and the packages will be finally sent to the output bins. Turnarounds are used to switch package flow while segments enable the flow of packages on their belts. A sample topology of a CCS is shown in Figure 2.

In the figure, In_1 and In_2 are two input bins, S1,S2 ... S8 represent Segment belts. T1, T2 ... T4 are Switches and Out_1 and Out_2 are output bins.

B. System Assumptions

Our modeling and subsequent simulation approach used to evaluate the models are based on several assumption as follows:

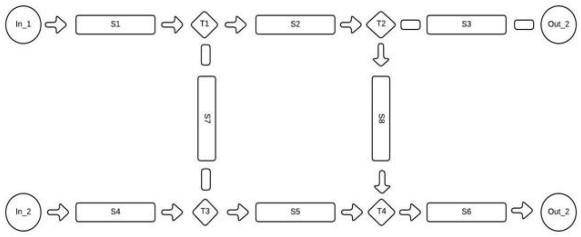

Fig. 2. Sample Topology for a Composable Conveyor System

1) Data relevant to control is collected and simulated within one time unit.
2) All packages have the same physical attributes including their dimensions, weight and coefficient of friction with conveyor belt's surface.
3) Speed of a belt remains constant within a given time unit, i.e., it cannot be changed during this interval of time.
4) Even if the number of packages in a certain time unit is zero, the system still runs at the lowest possible speed.
5) We do not consider power loss when there is a switching of the speed levels for a belt.
6) The belt energy consumption is directly proportional to the current speed of the belt. We will need to modify this assumption in future work, especially when considering the scenario when the speed is changed rapidly and when considering physical attributes of the packages such as weight and dimension differences.

C. System Variables used in the Controller Designs

The following variables related to segment belts are used in the controller design formulation.

- **Input variables:**

- **Arrival rate** λ **–** is the actual package workload from the belt input side. The current arrival rate is equivalent to the previous component's throughput. It is assumed that incoming packages are monitored by an embedded sensor and the data is collected.

- **Control input** v **–** is the control value which is generated by the controller. This value is used by the belt to choose its speed level for that time unit.

- **State variables:**

- **Belt speed** u **–** is the speed of the belt, which is fixed for a given time unit. In our model, we assume that the speed can be set from among multiple levels: low, medium, high. According to the control value signal v from the controller, the segment sets its own current speed. u_max is the maximum velocity that the belt can reach. At time k,

$$u(k) = v(k) \cdot u\_max \tag{4}$$

- **Throughput** y **–** is the package output rate per time unit incident on the next connected component. At

time k the throughput depends on the previous time unit's belt capacity $q(k-1)$ and the current adjusted speed $u(k)$, where T is the time unit.

$$y(k) = \min[q(k-1), u(k) \cdot T] \quad (5)$$

- **Belt capacity queue** q **–** is the total number of packages on a belt in a sample time unit period. It is related to previous capacity, current speed and arrival rate. We use q_max to represent maximum number of packages that the segment can hold.

$$q(k) = \max[q(k-1) + (\lambda(k) - y(k)) \cdot T, 0] \quad (6)$$

- **Energy consumption** e **–** is the belt energy consumption, which is directly proportional to the current belt velocity as shown in the equation below where α is a parameter.

$$e(k) = \alpha \cdot u^2(k) \quad (7)$$

D. Limited Lookahead Controller for CCS

As discussed in Section III, our actual system (belt segment) can generate the real-time system state and apply optimized control input $u(k)$ using a Limited Look-ahead Controller (LLC). We also define a utility function within the optimizer to measure the QoS. This function takes into account 1) belt energy consumption, 2) belt current throughput, and 3) the cost of the control inputs themselves and their switching change.

In the predictive filter (see Figure 1) we predict for the next several time steps the package arrival rate, and send to system model to generates several future system state. We use ARIMA model to make the estimation and adjust the parameter based on the patterns of arrival. At time k, assuming the look ahead step is 1, the prediction model will be:

$$\hat{\lambda}(k+1) = \beta \cdot \lambda(k) + (1-\beta) \cdot \lambda\_avg \quad (8)$$

where $\lambda(k+1)$ is the estimated arrival rate and the parameter alpha is used as a weighting factor. A high β value would make the estimated value of the next arrival rate closer to the current $\lambda(k)$, while a low β value will make it closer to the average λ_avg. λ_avg will be calculated based on the history of arrival rate, which is maintained in a file buffer (up to a certain limit, for example, the last 100 values).

The system model can traverse all possible speed values $\hat{u}(k+1)$, generate the predicted state variables, and then send the state to the optimizer shown in Figure 1. The predicted values for the three state variables are:

- Throughput in the next time unit instant

$$\hat{y}(k+1) = \min[q(k), u(k) \cdot T]$$

- Queue level in the next time unit instant

$$\hat{q}(k+1) = \max[q(k) + (\hat{\lambda}(k+1) - \hat{y}(k+1)) \cdot T, 0]$$

- Energy consumption in the next time unit instant

$$\hat{e}(k+1) = \alpha \cdot \hat{u}^2(k+1)$$

In the Optimizer block, the controller can then use the system predicted state from the system model and calculate the utility function, and finally choose the speed which can maximize the next utility function, maximize energy savings and maximize the throughput. This function also needs to smooth the control process via minimizing $\Delta u = j\hat{u}(k+1) - u(k)j$, which is a threshold that reduces the speed switching time over all belts in the system (e.g., it is less expensive to switch from low to medium than from low to high).

$$\hat{J}(k+1) = max[-w\_1\hat{e}^2(k+1) + w\_2y^2(k+1) - w_3\Delta u] \quad (9)$$

where $w1$, $w2$ and $w3$ are weight value to smooth the system utility.

E. State Chart for Belt Speed Adjustment

As mentioned earlier, our model considers three different speed options for the segment belt: low, speed and high. Figure 3 depicts the finite state machine for belt speed adjustment based on a look ahead step of 2. Note that the number of states in the state machine depends on the number of look ahead steps. For example, when the lookahead step is 3, the total states in the state chart will grow to be 27; when the number of look ahead steps is 4, the total number of states is 81 etc. Thus, the state space increases exponentially.

As shown in the figure, for a 2-step look ahead we deal with a total of **6 speed** combinations for a belt (e.g., belt at low speed can remain low in the next instant or go to medium or high). Thus, when the control step is executed as an online algorithm, the computation tree will expand significantly. In our algorithm (and hence the simulations conducted for validation), when executing a trace of a 3-step look ahead, we use Depth-First-Search and traverse 27 paths, and obtain the lowest total cost among all such traversed paths, and return the speed value computed at the first level of the look ahead tree.

F. Controller Designs and Expected Results

We now present the design of our five different model predictive controllers that use the LLC approach. We also discuss the static configuration, which is used as the baseline for evaluation purposes. We qualitatively compare their properties relative to a baseline static (fixed) configuration design. We use 6 parameters to evaluate our control designs: 1) Memory cost 2) CPU time 3) throughput 4) energy consumption 5) belt speeding switching cost 6) system utility.

1) Static Configuration (SC): For the static configuration case, we use three fixed values for conveyor speed: low, high and medium. When the belts use low speed, although the energy consumption is obviously low, the system can accumulate large packages and lower the throughput, so the total system utility will be low. When belts apply high speed to maximize the throughput, it will waste energy especially when incoming package rate is small. Altogether, the system utility of applying low and high speed should be low, and for configuring medium speed case, the utility depends on the workload.

2) Independent Distributed and Decentralized Controllers: Figure 4 shows the decentralized, distributed controller approach. In this case, each belt has a local controller which predicts arrival rates independently of each other. This approach is easy to extend but lacks precision and incurs instability

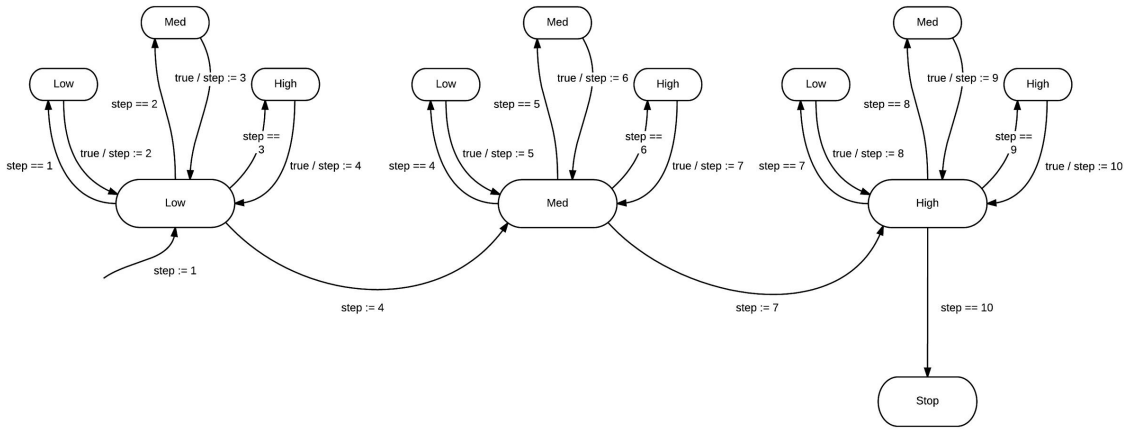

Fig. 3. Finite State Machine with Look Ahead Step of 2

with large variability in arrival rate. With the value of look ahead step growing, the system utility result could be larger or smaller depending on the weight value we set; CPU time and memory utilization should increase because of time and space complexity becoming larger. If we make system's throughput high enough so that throughput can act as a static variable in the utility function, then the function depends on the weight values of QoS parameters. In such a case, there will be one of the parameters among energy consumption and belt speed switching cost that performs better over the other.

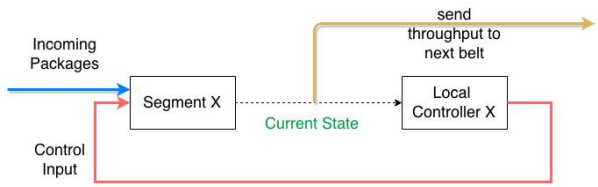

Fig. 4. Distributed controllers model design

3) Centralized Controller: Figure 5 depicts the centralized controller design. The system has one controller, which is responsible for prediction of speeds for all belts and decision making. This controller requires extensive centralized communication. Compared with the decentralized, distributed controllers, the behavior for throughput, belt speed switching cost and system utility should be similar. We need to further evaluate CPU time and memory usage because in distributed controllers, all controllers are self-optimized so that the control decisions are also made simultaneously. However, in a centralized controller, the control input for all belts emerge sequentially.

4) Decentralized Controllers With Dependency: Figure 6 depicts our third controller design, which is similar to the decentralized, distributed controller design. The differences is that each belt in system has a local controller that predicts the arrival rate of packages taking into account the speed of the preceding unit. This is more accurate than the previous approach but requires additional communication. When the look ahead steps is larger than 1, the CPU time for executing this controller should be smaller than the independent distributed controllers. For instance, in our system model, only $S1$ and

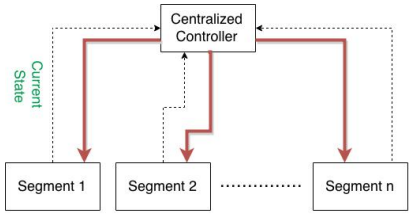

Fig. 5. Centralized controller model design

$S4$ that connect with input bins are required to executed the look ahead designated steps, while the other belts just need to predict future one step according to the sum of their input belt speeds.

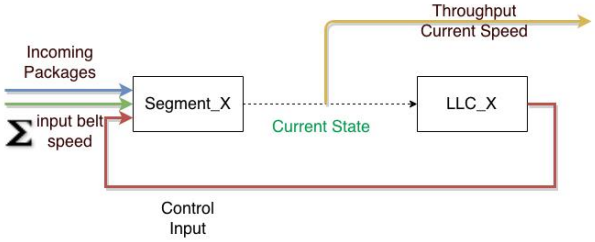

Fig. 6. Decentralized controllers model design

5) Hierarchical Controllers: This control design is based on the distributed controllers design, however, with two levels as shown in Figure 7. At the first level, the individual controllers make short-term predictions of workloads. These controllers are in turn managed by a second level global controller that makes longer-term forecasts of expected workloads and fine tunes the performance of the system. The global controller tunes the range of control input every 60 time units. Within 60 time units, each belt keeps sending its real-time state variable (current speed, incoming packages, queue level, throughput and energy consumption) to the global controller, so the communication of the global controller is very high, which is proportional to the total number of belts in the system.

The global controller can compute the average values for

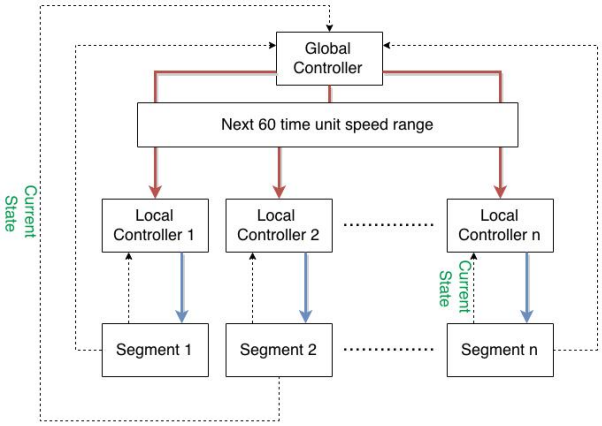

Fig. 7. Hierarchical global controllers model design

all state variables over a number of time units as shown below for 60 time units.

$$X(K) = [U(K), \Lambda(K), Q(K), Y(K), E(K)]$$

where

$$U(K) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{60} u_i(j)}{60 \cdot n}$$

where $X[K]$ is the global state and $U[K]$ is the average of all belt speeds in this case for the given time range. Other state variables in $X(K)$ are defined similarly. The long term prediction generates next 60 time unit global states by traversing all possible speed levels $\hat{H}(k+1)$ to calculate global utility function J_G.

$$\hat{X}(K+1) = [\hat{Q}(K+1), \hat{Y}(K+1), \hat{E}(K+1)]$$

$$\hat{J}_G(K+1) = max[-w1 \cdot \hat{E}^2(K+1) + w2 \cdot Y^2(K+1) - w3 \cdot \Delta U]$$

The benefit of hierarchical controllers is that the global controller can reduce the local controllers having to traverse all the steps in decision-making computation and should save significant CPU time, i.e., if the lookahead steps is 3, once the global controller decides the speed range as medium, then the local controllers do not have to traverse all 27 paths as decentralized controllers'local controllers did. The local controllers in hierarchical controllers can traverse only one path rather than 27 paths.

6) Time Abstract Centralized Controller: In our final approach, one centralized controller is used in the system. The controller synthesizes the properties of the global controller and distributed local controllers in the hierarchical controllers design. The centralized controller can not only tune the control input for all belts, but also self-tune the speed range every pre-set time unit to reduce the decision making time. So the parameters of time abstract centralized controllers design on throughput, energy consumption, belt speed switching cost should be similar to the hierarchical controllers. The communication cost of the centralized controller is as high as the global controller in hierarchical controllers design. However, unlike hierarchical controllers, in time abstract centralized controller there are no segments communicates with a local controller, so the CPU time of time of abstract centralized controller should less than hierarchical controllers.

V. Evaluating the Controller Designs

We evaluate the performance management properties of our five controllers via simulations. We use Matlab to implement the controller design in the context of a sample topology shown in Figure 8. Two workload flows are used according to the arrow direction: $In_1 \rightarrow S1 \rightarrow T1 \rightarrow S2 \rightarrow T2 \rightarrow S8 \rightarrow T4 \rightarrow S6 \rightarrow Out_2$ $In_2 \rightarrow S4 \rightarrow T3 \rightarrow S5 \rightarrow T4 \rightarrow S6 \rightarrow Out_2$. The data collection is from a network flow scenario, which is a trend model that provides all the requirements in our experiments. All data is collected over 9,000 simulation time units. A time unit is the smallest time range to simulate discrete event, i.e. within one time unit, we need to transmit 500 packages from one belt to its succeeding belt.

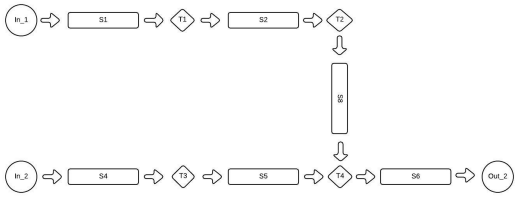

Fig. 8. Pre-fixed Routing

A. Prediction Evaluation

We evaluate local controller's prediction and self-optimized performance by applying look ahead 1 step. The actual work loads is presented in Figure 9. Initially, the local controller does nothing, simply keeps waiting and collecting data about incoming packages via an embedded sensor, and it controls the belt using high velocity. After collecting enough data for control purposes, the local controller starts using ARIMA prediction model. It sends the control input to the belt. In Figure 10 we can see the trend for belt speed for each time unit as it matches the fluctuation of actual incoming package arrival rate with a sample 240-unit time period. As illustrated in Figure 11, 100% denotes the accuracy of workload prediction by the local controller. The prediction consistency line is gradually growing after first 100 time units, and remains high. As a result, the mean of prediction correctness for 9,000 time unit is 96.06%, with the standard deviation of 0.1058. In conclusion, when the trend of incoming packages arrival rate is close to a trend model, it is predictable and well-defined.

B. Controllers Design Evaluation

In this section, we evaluate the different 5 control designs with 6 different parameters discussed in Section III. For each design, we apply 3 different look ahead steps to evaluate the simulation performance.

1) CPU Time: The simulation unit of decision making time unit is millisecond shown in Figure 12. For independent distributed and decentralized controllers, we can easily observe that the time increases exponentially. Hierarchical controllers greatly reduce the traverse time each round, so it spends

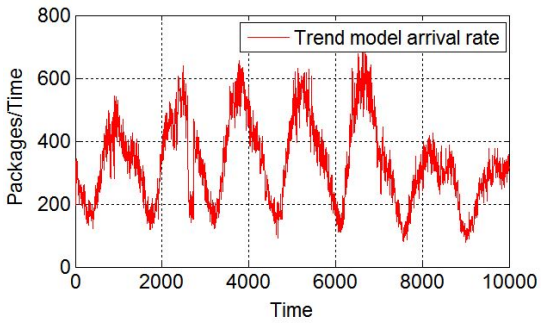

Fig. 9. Trend-mode Actual Arrival Rate

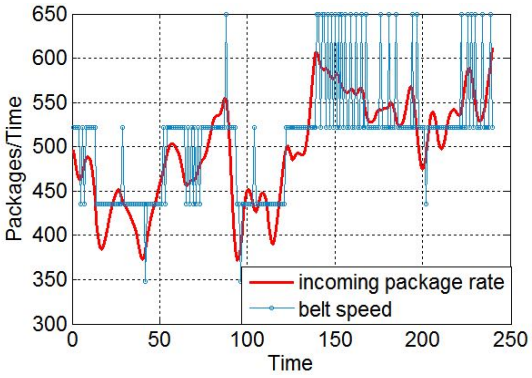

Fig. 10. Belt Speed Matches Trend of Arriving Workload

much less time compared to independent controllers as look ahead steps increase. When look ahead steps is larger than 1, not all belts in decentralized controllers with dependency need to forecast that many steps if they have input belts, so this design's CPU time is better than completely independent controllers. In the independent controllers, the decision making is simultaneous, and in centralized controllers, the strategies are made sequentially, but we fail to see any big difference with their CPU time performance. Both hierarchical controllers and time abstract controllers have long-term and short-term decision making, while the former one has two levels, so its CPU time is larger than time abstract one-level controller.

2) Memory Usage: With the look ahead step growing, the average (Figure 13) and maximum memory usage (Figure 14)

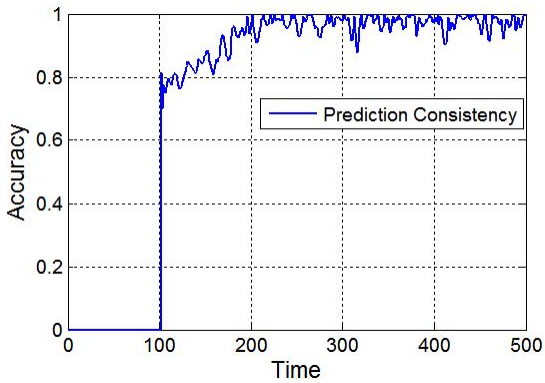

Fig. 11. Prediction Accuracy Sample Example

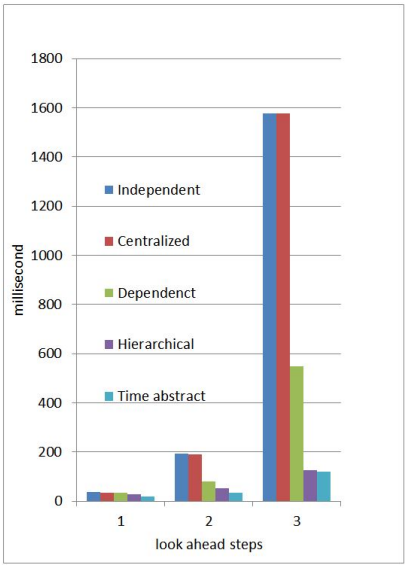

Fig. 12. CPU Time Summary

of each approach also increases. Decentralized controllers with dependency consumes large memory because of the extra communication within belts and their preceding belts. Besides local communication, the hierarchical controller design also needs layer-to-layer communication between first level and second level, so it performs worse than one level control design, e.g., independent distributed and decentralized controllers.

The centralized design's memory usage is less than distributed design's although there is much more interaction in centralized controller. Hierarchical controllers needs more local communication than time abstract centralized controllers and hence needs more memory usage. According to the result between independent controllers and centralized controllers, we can conclude that local distributed communication memory cost is more significant than centralized communication.

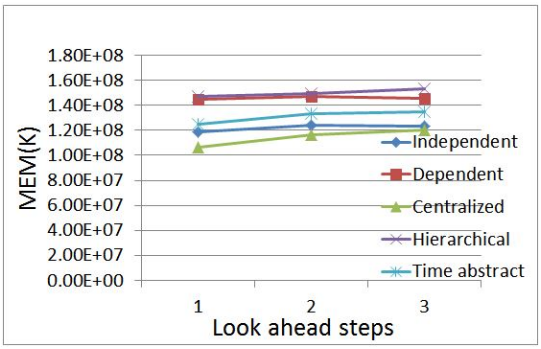

Fig. 13. Average Memory Usage Summary

3) Throughput: We assume the capacity of each belt is large enough, so all 5 approaches can be self-optimized and achieve maximum throughput at the end of the simulation. For this, the max speed must be set sufficiently high for large arrival rate. However in real industry, package piling up is also a key question. If we have a large arrival rate that max belt velocity cannot handle, the number of packages on a belt will substantially increase, and it will need more time to reduce the

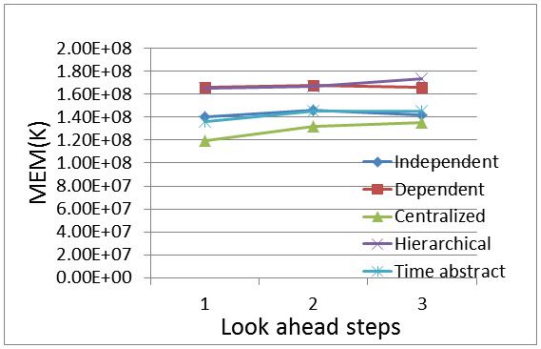

Fig. 14. Maximum Memory Usage Summary

queue of a belt to reach a normal load. In this case, we need to use extra warehouse or space to store the overflow packages. Figure 15 shows the relationship with between large incoming arrival rate (first 200 time unit) and belt capacity.

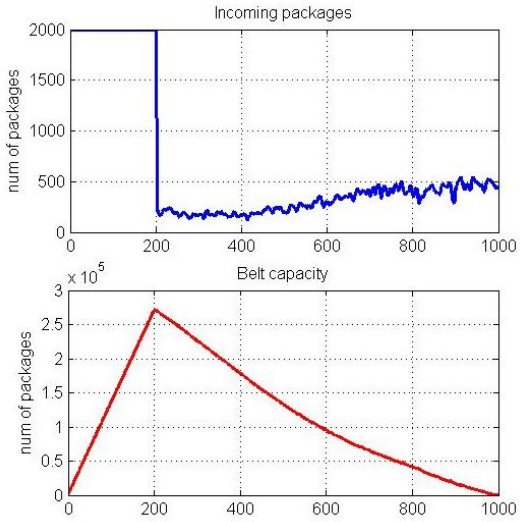

Fig. 15. Package Piling Up

This exceeding package accumulation can happen on any belts that connect to the input bins. In both the hierarchical controllers and time abstract controllers, packages piling up may also happen when large workload fluctuations occur between two long term predictions. Because the speed level range of the local controller can be used to traverse within a long-term unit, the superfluous packages can be detected and handled in next long-term forecasting.

4) Energy Consumption and Speed Switching Cost: We fixed the throughput parameter to be maximum value so that we can analyze belt energy (Figure 16 and speed switching cost (Figure 17 at the same time. With the look ahead steps increasing, the energy consumption has the same trend, but the speed switching cost has opposite trend. This implies if we reduce the time and range of switching belt speed, the total energy cost will increase. It also illustrates that the system aims to smooth the belt switching level when the system look ahead has more steps. Decentralized local controllers that connect a belt which consider the preceding belt's speed as estimated arrival rate can only foresee 1 look ahead, so the decentralized controllers with dependency performs worst in this scenario. On the other hand, independent distributed controllers make similar belt switching decision as hierarchical controllers, however, independent controllers conserve less energy.

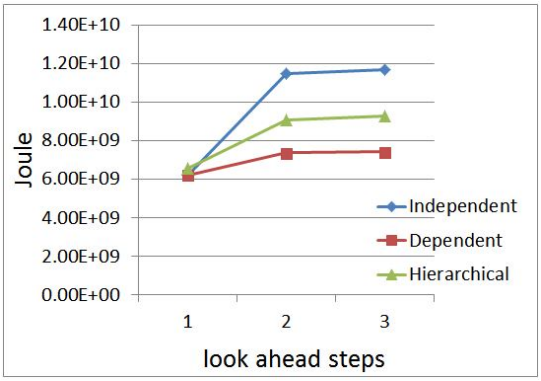

Fig. 16. Energy Consumption Summary

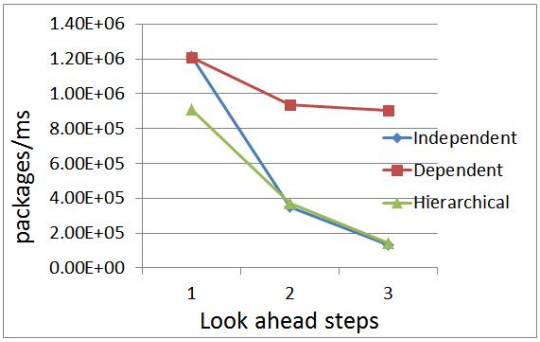

Fig. 17. Belt Speed Switching Cost Summary

5) System Utility: The system utility decreases with increasing look ahead steps as shown in Figure 18. In consideration of energy consumption and belt speed switching cost, we can conclude that if we further increase the look ahead step, the performance cannot have prominent change, so the system utility of the 5 control designs tends to be stable. So they are greatly superior to a static configuration system (see Table I). The reason for the utility trend for 5 control designs is that since we drop the belt switch cost so that the energy consumption gets large, it is decided by utility function's weight value. For static configuration, high speed can result in max throughput but needs incredible energy consumption; low or medium speed may not handle high package arrival rate so that the system throughput gets smaller, which greatly affects the system utility.

To decide which strategy and how many look ahead steps we need to overall consider all 6 parameters. Decentralized controllers with dependency performs better than the other 4 control designs, but it needs more memory usage and its CPU time is also exceptionally high when look ahead step is large. The system utilities of independent distributed controllers and centralized controllers are almost same but they cannot do better when look ahead steps is larger. Their CPU time is also short but they can conserve more energy. For hierarchical controllers and time abstraction controllers, they

can use less CPU time to make a decision compared to the other 3 approaches with large memory usage. But they can reduce the belt speed switching cost which can also compete with other control designs.

In summary, to decide which control design to use is a trade-off and we need to consider more on what is going on in real industry environment.

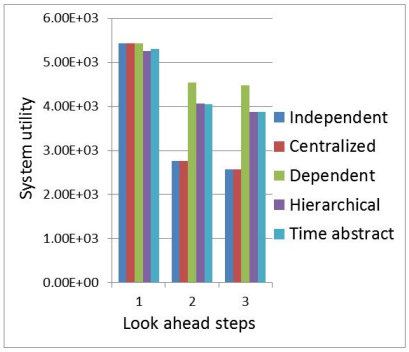

Fig. 18. System Utility Summary

TABLE I. SYSTEM UTILITY FOR STATIC CONFIGURATION

Static Configuration	Low	Medium	High
System Utility	-2.49E+11	-1.91E+15	-6.04601E+13

VI. CONCLUSIONS

This paper presented five controller designs that are based on model-predictive concepts with Limited Lookahead Control (LLC). The five controller designs comprise a single-level, completely decentralized controller with two alternatives on how belt speed is estimated; a centralized controller; a two-level controller; and a time-abstracted centralized controller.

We evaluate the prediction accuracy on incoming package arrival rate, and demonstrate how conveyor belts are self-optimized to match the fluctuation in workloads under the actuation commands from the controllers. The controllers can tune the belts to select the best speed for the next time unit according to a self-defined utility function. We also compare the CPU time and memory usage of 5 designs, which are also the metric to evaluate the controllers. Our result shows that it exists some dependence relationship among all 6 parameters, we need to balance them in order to know which control designs we should apply and get good QoS result.

Our future work in this area will explore analysis of failures in the system. We plan to target both the physical failures, such as motor failing, and cyber failures, such as the microcontroller logic failing. Our goal is to identify the impact on the system throughput due to failures, and also to understand how runtime adaptation by rerouting goods will help to maintain acceptable levels of performance in system operation. We also plan to prioritize according to package types and packet dimensions.

The controller designs in the form of Matlab models are available at https://github.com/onealbao/Composable_Predictable_Controller/.

ACKNOWLEDGMENTS

This work is supported in part by NSF CAREER CNS 0845789. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

REFERENCES

[1] B. Archer, S. Sastry, A. Rowe, and R. Rajkumar, "Profiling primitives of networked embedded automation," in *IEEE Conference on Automation Science and Engineering*, 2009.

[2] AMP 2.0 Steering Committee, "Accelerating U.S. Advanced Manufacturing," President's Council of Advisors on Science and Technology (PCAST), Tech. Rep., Oct. 2014.

[3] K. An, A. Trewyn, A. Gokhale, and S. Sastry, "Model-driven Performance Analysis of Reconfigurable Conveyor Systems used in Material Handling Applications," in *Second IEEE/ACM International Conference on Cyber Physical Systems (ICCPS 2011)*. Chicago, IL, USA: IEEE, Apr. 2011, pp. 141–150.

[4] S. Sastry and A. Gokhale, "Resolving Priority Inversions in Composable Conveyor Systems," *Elsevier Journal of Systems Architecture (JSA)*, vol. 60, no. 6, pp. 509–518, Jun. 2014.

[5] S. Abdelwahed, J. Bai, R. Su, and N. Kandasamy, "On the Application of Predictive Control Techniques for Adaptive Performance Management of Computing Systems," *Network and Service Management, IEEE Transactions on*, vol. 6, no. 4, pp. 212–225, 2010.

[6] J. Luo, W. Huang, and S. Zhang, "Energy cost optimal operation of belt conveyors using model predictive control methodology," *Journal of Cleaner Production*, no. 0, pp. –, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959652614010130

[7] S. Zhang and Y. Tang, "Optimal scheduling of belt conveyor systems for energy efficiency with application in a coal-fired power plant," in *Control and Decision Conference (CCDC), 2011 Chinese*, May 2011, pp. 1434–1439.

[8] L. Ristic and B. Jeftenic, "Implementation of Fuzzy Control to Improve Energy Efficiency of Variable Speed Bulk Material Transportation," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 7, pp. 2959–2969, July 2012.

[9] Q. Lu, X. Wang, and L. Zhuang, "Research and design of monitoring system for belt conveyor," in *Computer Science & Service System (CSSS), 2012 International Conference on*. IEEE, 2012, pp. 1943–1945.

[10] G. Black and V. Vyatkin, "Intelligent component-based automation of baggage handling systems with iec 61499," *Automation Science and Engineering, IEEE Transactions on*, vol. 7, no. 2, pp. 337–351, 2010.

[11] Y. Maeda, H. Kikuchi, H. Izawa, H. Ogawa, M. Sugi, and T. Arai, "An easily reconfigurable robotic assembly system," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 2586–2591.

[12] T. Kruhn, S. Falkenberg, and L. Overmeyer, "Decentralized control for small-scaled conveyor modules with cellular automata," in *Automation and Logistics (ICAL), 2010 IEEE International Conference on*. IEEE, 2010, pp. 237–242.

[13] D. Menascé, V. Almeida, R. Riedi, F. Ribeiro, R. Fonseca, and W. Meira Jr, "In search of invariants for e-business workloads," in *Proceedings of the 2nd ACM conference on Electronic commerce*. ACM, 2000, pp. 56–65.

[14] M. F. Arlitt and C. L. Williamson, "Web server workload characterization: The search for invariants," in *Proc. ACM SIGMETRICS Conf.*, 1996, pp. 126–137.

[15] M. Arlitt and T. Jin, "Workload characterization of the 1998 world cup web site," Hewlett-Packard Labs., Technical Report HPL-99-35R1, September 1999.

[16] S. A. DeLurgio, *Forecasting Principles and Applications*. McGraw-Hill, 1998.

[17] K. Brammer and G. Siffling, *Kalman-Bucy Filters*. Norwood MA: Artec House, 1989.