Distributed resource management using iterative gradient update synthesis

Mårtensson, Karl; Vladimerou, Vladimeros

2011

2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Distributed resource management using iterative gradient update synthesis

Karl Mårtensson and Vladimeros Vladimerou

Abstract— **We consider load balancing on a network. Servers of limited bandwidth move a single commodity through a network of buffers (or queues) while external random processes generate and consume this commodity. Our contribution is a distributed algorithm for regulating the backlogs of these queues to a given reference while balancing the mean flow in the network.**

We formulate this as a fluid buffer regulation problem and use distributed gradient descent to update the feedback gains for an LQG controller. Our proposed distributed algorithm both implicitly and explicitly estimates the statistics of the external process flows using only local information on fixed time intervals and updates the feedback matrix for the regulator accordingly.

We demonstrate our method on a simulation of an industrial floor where autonomous vehicles remove palettes from production line buffers.

I. Introduction

Flow optimization and load balancing problems presented on computer queuing networks, can be examined from a discrete or continuous point-of-view, using algorithms on graph structures. Given a graph, the interconnection of sources and destinations, linear programming [6], [5] and fast approximate solutions can be applied for solving single- or multi-commodity flow maximization problems [7].

Also related to flow maximization is the problem of load balancing, where many servers partake in servicing incoming tasks and the goal is to maintain a load balance so as to not overload some of the queues while allowing others to idle. In [1], the authors used Discrete Event Systems models on a metric state space and Lyapunov theory for the analysis of load balancing systems. In [9] the authors also deal with load sharing in terms of buffered systems, in the same DES framework, with optimal solutions to the allocation of bandwidth and buffer size.

In Fig. 1 we see the configuration where a network of three servers are connected with 7 queues. In the work by [13] the models allow the servers to route jobs from each queue to another.

From some queues, the servers can consume the jobs, getting them outside of the system. Traffic with specified destination gets generated at the queues with some random rates (usually with bounded burst rates), and the goal is to balance/stabilize the queues and route traffic to its destination where it gets consumed, or converted to another type of traffic.

This work is supported by EU/IST project CHAT, and the LCCC

Karl Mårtensson is with the Dept. of Automatic Control, LTH, Lund University karl.martensson@control.lth.se

Vladimeros Vladimerou is with the LCCC Linneaus Center in LTH, Lund University vladimeros.vladimerou@control.lth.se

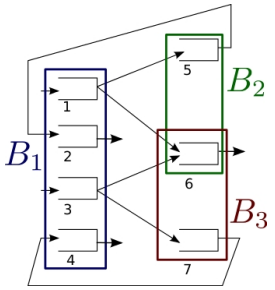

Fig. 1. Servers B_1, B_2, and B_3 serve queues $1,2,3,4,5,6$, and 7 as shown. Incoming traffic can be routed to other queues (for example from queue 1 to queues 5 or 6) or out of the system (for example, from queue 4 or 6).

Other work considers discretely enabled links in order to avoid interference [3]. Distributed weighted graph matching algorithms are the main method used in that and similar work. Longest queue first (LQF) schemes also fall in the the category of discrete service activation. Local pooling [4] properties have been long used for providing guarantees for stability of such schemes. Stability proofs follow from Lyapunov theory as the longest queue becomes then the Lyapunov function for the system.

Quite similarly, the backpressure algorithm [13], in various forms, uses an intuitive method of feedback that gives stability of the queues. For example in Fig. 2 we see that server B_1 is allocated to one of queues $1,2,3$. The filled areas in the figure represent quantitatively the backlogs of the queues. Jobs are routed according to the differential "pressure" induced by the size of the queue backlogs at the servers' sources and destinations.

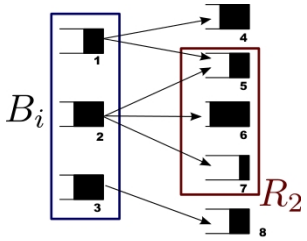

Fig. 2. Backpressure algorithm intuition: Server i serves queues $B_i = \{1,2,3\}$. Traffic from queue 2 can be routed by server i to any queue in the set $R_2 = \{5,6,7\}$. In this example, with the queue backlog as the filled area in each queue, server i will choose queue 2 to service and will direct its traffic to buffer 7.

We present a combined flow optimization and load bal-

ancing algorithm. Our work considers the allocation of server resources in routing identical items from their sources to possible destinations through a network of buffers (or queues) so as to balance the queue backlogs and avoid overflows. Our model is very similar to the classic one in [13] with one of the main differences being that it is formulated on a continuous state space to begin with. Unlike the above references which use, in the heart of their algorithms, differential queue comparison, we use a distributed LQG regulator, which allows for very fast localized computation. At the same time, in longer intervals, we update the statistics and improve the performance of our feedback rule using a distributed gradient descent algorithm introduced in [10], [11] and a linear program. Our method has fast computation times so it can be applied to both vehicular traffic as well as digital computer networks.

The rest of this article is structured in sections as follows:

(II) after this introduction we give a preliminary overview of our distributed iterative learning control (ILC) scheme

(III) we formulate the problem on a networked buffer model and give our complete algorithm

(IV) we demonstrate the effectiveness of our algorithm in an example application: an industrial multi-vehicle system for handling product palettes

(V) we conclude with some remarks and future plans

II. Distributed iterative learning control

In this section we will summarize the theory behind the method for distributed ILC, found in [10], [11]. The systems considered have an associated graph, whose set of vertices is a partition of the system states in such a way that each vertex represents a subsystem of the complete system. An edge between two vertices indicates that two subsystems directly affect each other. The edges also specify with which subsystems a certain vertex can communicate. Consider a discrete time system in state space form

$$x[n+1] = Ax[n] + Bu[n] + w[n] \tag{1}$$

where w is independent, zero-mean, Gaussian noise with variance σ_w^2. Let the associated graph be $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ the set of edges (i.e. ordered pairs of vertices). Then there will be a partition of the states over the vertices, $x = \begin{bmatrix} x_1^T & \dots x_{|\mathcal{V}|}^T \end{bmatrix}^T$ (we write x_i for the states associated to vertex v_i). According to the edges of the graph, we restrict the blocks of the dynamics matrix A by

$$A_{ki} \neq 0 \implies (v_i, v_k) \in \mathcal{E} \text{ or } (v_k, v_i) \in \mathcal{E}$$

We also assume that each subsystem has a distinct set of control signals, implying that the B matrix will be block diagonal.

The control scheme will be state feedback, i.e. $u[n] = -Lx[n]$. In a similar manner as with the dynamics matrix, we impose a restriction on the allowed feedback matrices L

$$L_{ki} \neq 0 \implies (v_i, v_k) \in \mathcal{E} \text{ or } (v_k, v_i) \in \mathcal{E}$$

The structure of the feedback matrix implies that for a subsystem to calculate its control signal, it is allowed to use measurements from other subsystems only if those are its neighbors, i.e. there is a edge between them in the graph.

We introduce the performance

$$J(L) = \mathbf{E}\left(|x|_Q^2 + |u|_R^2\right)$$

where x satisfies (1) in stationarity with $u = -Lx$. The weight matrix Q is assumed to be *block-diagonal*, positive semi-definite and R is assumed to be *block-diagonal* positive definite. Note that the performance J is only defined for stabilizing feedback matrices L.

The objective of the distributed ILC scheme is to use measurements of x and u from neighboring agents to update the feedback matrix to improve the performance J. This will be done by determining a descent direction G of J with respect to L and change L according to

$$L^{\text{new}} = L + \gamma G$$

where γ is sufficiently small. All calculations are to be performed in a localized manner, i.e. to update the feedback in agent x_i, only local measurements and local model information may be used.

In order to find a descent direction of $J(L)$, we determine the gradient of $J(L)$.

Proposition 1: Given a stabilizing state feedback $u[n] = -Lx[n]$ to the system (1), the gradient of J with respect to L is

$$\nabla_L J = 2\left[RL - B^T P(A - BL)\right] X \tag{2}$$

where X and P satisfy the Lyapunov equations

$$X = (A - BL)X(A - BL)^T + \sigma_w^2 \tag{3}$$

$$P = (A - BL)^T P(A - BL) + Q + L^T RL \tag{4}$$

Proof. We use the fact that $J = \operatorname{tr}(P\sigma_w^2)$. Denote

$$A_L = A - BL$$
$$M = dL^T(RL - B^T PA_L)$$

By differentiating (4) w.r.t. L we see that dP satisfies the Lyapunov equation

$$dP = A_L^T dP A_L + M + M^T$$

Hence,

$$dP = \sum_{k=0}^{\infty} (A_L^T)^k (M + M^T) A_L^k$$

and

$$dJ = \operatorname{tr}\left(dP \cdot \sigma_w^2\right) = 2\operatorname{tr}\left(M \sum_{k=0}^{\infty} A_L^k \sigma_w^2 (A_L^T)^k\right) = \\ = 2\operatorname{tr}\left(dL^T(RL - B^T PA_L)X\right)$$

We conclude the proof by using the fact that $dZ = \operatorname{tr}(dX^T Y) \Rightarrow \nabla_X Z = Y$ for matrices $X^T, Y \in \mathbb{R}^{n \times p}$. □

By introducing adjoint (or dual) state variables, (2) can be rewritten so that the gradient in can be computed in a distributed way.

Proposition 2: Under the assumptions of Proposition 1, let the stationary process λ defined by the backwards iteration

$$\lambda(t-1) = (A-BL)^T\lambda[n] - (Q+L^TRL)x[n] \tag{5}$$

where $x[n]$ is the state of the original system (λ is called the adjoint or dual state). Then

$$\nabla_L J = 2(RL\mathbf{E}xx^T + B^T\mathbf{E}\lambda x^T) \tag{6}$$

Proof. Denote $Q_L = Q + L^TRL$. The dynamics equation of λ gives

$$\begin{aligned}\lambda[n] &= -\sum_{j=n+1}^{\infty}(A_L^T)^{j-n-1}Q_Lx[j] = \\ &= -\sum_{j=0}^{\infty}(A_L^T)^jQ_LA_L^{j+1}x[n] + \\ &+ \Psi\{(w[n], w[n+1], \ldots)\}\end{aligned}$$

where the operator Ψ is the appropriate linear operator of how the noise $w[k]$ affects $\lambda[n]$ for $k \geq n$. Hence

$$\begin{aligned}\mathbf{E}\lambda[n]x[n]^T &= -\mathbf{E}\left(\sum_{j=0}^{\infty}(A_L^T)^jQ_LA_L^{j+1}x[n]x[n]^T\right) = \\ &= -PA_LX\end{aligned}$$

since $w[k]$ and $x[n]$ are independent for $k \geq n$. □

By running the system for a number of steps and simulating the adjoint equations backwards in time, each subsystem can collect measurements of local states x and local adjoint states λ. These can then be used to determine estimates of the variances and cross-covariances in (6) and hence determine an estimate of a part of the gradient of J used to update the part of the feedback matrix relevant to that subsystem. In subsystem v_i the parts of the gradient relevant to update L_i is $[\nabla_L J]_{ik}$ where $(v_i, v_k) \in \mathcal{E}$ or $(v_k, v_i) \in \mathcal{E}$. Further analysis on how this procedure actually is distributed, i.e. all computations can be performed while using the allowed communication links, can be found in [10].

III. Model and design description

Our system is described by a directed bigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V} = \mathcal{S} \cup \mathcal{Q}$, where $\mathcal{S}$ is a set of servers or routers, and $\mathcal{Q}$ is a set of queues or buffers. We use a continuum of fluid material to represent items routed by the servers from queue to queue. At each queue $i \in \mathcal{Q}$, based on a stationary discrete-time random process $w_i[n] \in \mathbb{R}$, a continuum of fluid is produced ($E(w_i) \geq 0$), or consumed ($E(w_i) < 0$) every time instant n. Each server $j \in \mathcal{S}$ has a bandwidth capacity C_j which allows it to redirect fluid among different queues. More specifically, at each time instant, it can move some fluid from any queue in its incoming set $\mathcal{Q}_j^{\text{in}} = \{i|(i,j) \in \mathcal{E}\} \subset \mathcal{Q}$ to any queue in its outgoing set $\mathcal{Q}_j^{\text{out}} = \{k|(j,k) \in \mathcal{E}\} \subset \mathcal{Q}$ for up to a total of C_j units of fluid. Similarly, each queue is characterized by an incoming set $\mathcal{S}_i^{\text{in}} = \{j|(j,i) \in \mathcal{E}\} \subset \mathcal{S}$ and an outgoing set $\mathcal{S}_i^{\text{out}} = \{k|(i,k) \in \mathcal{E}\} \subset \mathcal{S}$ of servers, which can reroute its fluid backlog. For shorthand we define $Q_i = Q_i^{\text{out}} \cup Q_i^{\text{in}}$ and $\mathcal{S}_i = \mathcal{S}^{\text{out}}{}_i \cup \mathcal{S}^{\text{in}}{}_i$. We also define: $\mathcal{P}_k = \{i|\exists j \;.\; j \in \mathcal{S}_k \wedge j \in \mathcal{S}_i\}$, the set of peers of queue k (other queues that share servers).

When $E(w_i) \geq 0$ ($E(w_i) < 0$) then we say that queue i is an input (output) node. This is shown in Fig. 3.

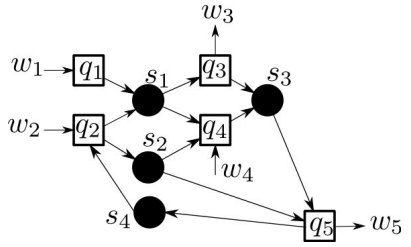

Fig. 3. An example of a network subsumed by our formulation. Here, for example, $\mathcal{S}_{q_2}^{\text{out}} = \{s_1, s_2\}$, $\mathcal{S}_{q_2}^{\text{in}} = \{s_4\}$, $\mathcal{S}_{q_4}^{\text{out}} = \{s_3\}$, $\mathcal{S}_{q_4}^{\text{in}} = \{s_1, s_2\}$, $\mathcal{Q}_{s_1}^{\text{out}} = \{q_3, q_4\}$, $\mathcal{Q}_{s_1}^{in} = \{q_1, q_2\}$. Queues q_1, q_2, and q_4 are input nodes and queues q_3, q_5 are output nodes.

The system equations are thus:

$$\begin{aligned}q_i[n+1] = \quad & q_i[n] + w_i[n] \\ & - \sum_{j \in \mathcal{S}_i^{\text{out}}} \sum_{k \in \mathcal{Q}_j^{\text{out}}} u_{jik}[n] \\ & + \sum_{j \in \mathcal{S}_i^{\text{in}}} \sum_{l \in \mathcal{Q}_j^{\text{in}}} u_{jli}[n]\end{aligned} \tag{7}$$

where $u_{jik}[n]$ is the amount of fluid routed at time n by server j from queue i to queue k, and $w_i[n]$ is Gaussian $\sim N(\mu_i, \sigma_i^2)$ for every i, n, indicating an uncertain production/consumption rate. One can consider infinite capacity queues, or queues with an upper backlog limit $0 \leq q_i[n] \leq Q_i$, $\forall i, n$. In addition, one can either assume a closed system where fluid only enters and leaves the system via the random processes w_i, or an open system where some queue is allowed to have an infinite backlog (positive or negative). Using the latter one can model situations where a external resource can consume or produce fluid in the network at rates much higher than the processes in the queues.

Given that we have finite routing capacity for each server, i.e. that $\sum_{(i,k) \in \mathcal{Q}_j^{in} \times \mathcal{Q}_j^{out}} u_{jik} \leq C_j$, for some $C_j > 0$ for each $j \in \mathcal{S}$, the goal is to design the process $u_{ijk}[n]$ so that the variance of $q - q^{\text{nom}}$, for some given reference q^{nom} is minimal. In other words, we would like to control the backlogs of the queues to be as close to some fixed value as possible.

A. Nominal Operating Points

The first step in designing a controller for our system is to remove the nonzero means from the random production/consumption processes $w_i[n]$, so that we bring the problem into an LQG formulation. One has to assume that the nominal consumption/production rates are sustainable given the capacities of the servers. This can be verified via max-flow or feasibility analysis [6], [5] using a simple linear program.

After part of the server capacity is used to balance out the nonzero-mean noise, the rest can be allocated to regulate fluctuations in the system fluid. Without feedback for variance regulation, the queues will have a zero mean but unfortunately, the open-loop variance of their backlogs grows to infinity linearly with n, $E|(q[n]-q^{\text{nom}})^2| = \sigma^2 n$.

An intuitive way to distribute server capacity towards variance regulation is to partition a static bandwidth u_{jik} of each server j allocated to each link $i \to j \to k$ into: $u_{jik} = u_{jik}^{\text{nom}} + u_{jik}^{\text{slack}}$ where u_{jik}^{nom} is used to remove the nominal mean of the exogenous inputs w_i and the slack u_{jik}^{slack} is used for variance regulation. This we can do using the following linear program:

$$\max_{u_{jik}^{\text{nom}}} \min_i \quad \Big(\sum_{j \in \mathcal{S}_i^{\text{out}}} \sum_{k \in Q_j^{\text{out}}} u_{jik}^{\text{slack}} + \sum_{j \in \mathcal{S}_i^{\text{in}}} \sum_{l \in Q_j^{\text{in}}} u_{jli}^{\text{slack}} - \sigma_i \Big) \quad (8)$$

$$u_{jik} = u_{jik}^{\text{nom}} + u_{jik}^{\text{slack}} \qquad \forall j \in \mathcal{S},\ i,k \in \mathcal{Q} \quad (9)$$

$$\mu_i - \sum_{j \in \mathcal{S}_i^{\text{out}}} \sum_{k \in Q_j^{\text{out}}} u_{jik}^{\text{nom}} + \sum_{j \in \mathcal{S}_i^{\text{in}}} \sum_{l \in Q_j^{\text{in}}} u_{jli}^{\text{nom}} = 0 \quad \forall i \in \mathcal{Q} \quad (10)$$

$$\sum_{\substack{i \in Q_j^{\text{in}} \\ k \in Q_j^{\text{out}}}} u_{jik} \le C_j \qquad \forall j \in \mathcal{S} \quad (11)$$

$$u_{jik}, u_{jik}^{\text{nom}}, u_{jik}^{\text{slack}} \ge 0 \quad \text{if } (i,j),(j,k) \in \mathcal{E}, \quad \text{zero else} \quad (12)$$

The cost function to be maximized in (8) is the minimum difference between the sum of the incoming and outgoing slack component of the server bandwidth over all queues. The constraint (10) allocates the nominal components to balance the mean flow in the system and together q^{nom} gives the nominal operating point for the system. This allows us to eliminate the effect of the non-zero means $\mu_i = E[w_i]$ of the external flows in/out of the system, while, at the same time, allocating enough bandwidth to handle fluctuations where the noise has large variance σ_i. Constraint (11) gives the capacity bound for each server, and (12) gives the graph structure.

This program is used to calculate a "DC" component of the server bandwidth as shown in Fig. 4. It may be run periodically as the statistics of the production/consumption processes change. Sparsity of the system graph implies that one can use distributed constraint consensus algorithms such as the one in [12].

The quantization block in Fig. 4 is used to indicate a countable nature of the commodity - discrete items instead of continuous fluid. The saturation block is used to enforce the capacity limitation of the servers. In practice, saturation is avoided by the maximization of the minimum slack component of our servers together with the appropriate choice of Q and R matrices in the following LQG formulation. When saturation (seldomly) happens, we distribute the entire capacity of server j while maintaining the ratios belonging to each component u_{jik}, for example if $u_1 = u_{112} + u_{113} + u_{114}$ where $u_{112} = 1, u_{113} = 3, u_{114} = 3$ and $C_1 = 5$ the actual used capacities will be $u_{112}^{\text{actual}} = (\frac{5}{6})(1), u_{113}^{\text{actual}} = (\frac{5}{6})(3), u_{114}^{\text{actual}} = (\frac{5}{6})(3)$, summing to 5 units.

B. Distributed ILC Scheme

The theory described in the section II will now be used to minimize the variance around the nominal operating point found in III-A. Let $\tilde{q} = q - q^{\text{nom}}$, $\tilde{u} = u - u^{\text{nom}}$ and let w be independent, zero-mean, Gaussian noise. Translating (7) around the nominal operating point q^{nom} we get:

$$\tilde{q}[n+1] = \tilde{q}[n] + B\tilde{u}[n] + w[n] \quad (13)$$

The matrix B will be on the form $\begin{bmatrix} B_1 & \dots & B_{|\mathcal{S}|} \end{bmatrix}$ where each B_j correspond to each server in $\mathcal{S}$. The column of each B_j consists of all column vectors $[B_j]_{i_{\text{in}}, i_{\text{out}}}$, for $(i_{\text{in}}, i_{\text{out}}) \in Q_j^{\text{in}} \times Q_j^{\text{out}}$, with zeros in all entries except for a 1 in position i_{in} and -1 in i_{out}.

The servers are not allowed to use measurements of all queues to decide which of the queues to pick up from and deliver to. Each server s_j may only receive measurements from the queues in Q_j, i.e. only from the queues it serves. This imposes the restriction on the feedback matrix $L = \begin{bmatrix} L_1^T & \dots & L_{|\mathcal{S}|}^T \end{bmatrix}^T$ where L_j is associated with server s_j. The columns in each L_j must satisfy

$$[L_j]_i = 0 \text{ when } i \notin Q_j$$

With the structure imposed on L this means that $BL = \sum_{j \in \mathcal{S}} B_j L_j$ where the elements of each $B_j L_j$ are

$$[B_j L_j]_{ik} = \sum_{m=1}^{|Q|} b_{im} \ell_{mk} = \sum_{(i,m) \in Q_j} \ell_{mk}$$

Hence $[B_j L_j]_{ik} = 0$ unless $k \in Q_j$.

Now, we will use the theory presented in II for an algorithm to control the system of queues. The distributed algorithm becomes

Algorithm 1: Between times n_k and $n_{k+1} - 1$, let the state feedback law be $\tilde{u}[n] = -L^{(k)}\tilde{q}[n]$ with the constant feedback matrix $L^{(k)}$. To update the feedback matrix $L_j^{(k)}$ in each server s_j:

1) Let the system of queues run for times between n_k and $n_{k+1} - 1$.
2) Each queue i determines its corresponding $\lambda_i[n]$ by communicating with neighboring queues according to (5).
3) Each server s_j receives measurements of all $q_i[n]$ and $\lambda_i[n]$ for $i \in Q_j$.
4) Each server determines estimates of all $\mathbf{E}\tilde{u}_j\tilde{q}_i$ for $i \in Q_j$ and $\mathbf{E}\lambda_{i_1}\tilde{q}_{i_2}$ for $i_1, i_2 \in Q_j$.
5) Each server determines a descent direction G_j in which to update its feedback matrix $L_j^{(k)}$ by
$$G_{ji} = -2\left[R_j(\mathbf{E}\tilde{u}_j\tilde{q}_i)_{\text{est}} + B_j^T(\mathbf{E}\lambda\tilde{q}_i)_{\text{est}}\right]$$
6) In each server, for all $i \in Q_j$, let
$$L_j^{(k+1)} = L_j^{(k)} + \gamma G_{ji}$$
for some step length γ.
7) Increase k by one and go to 1).

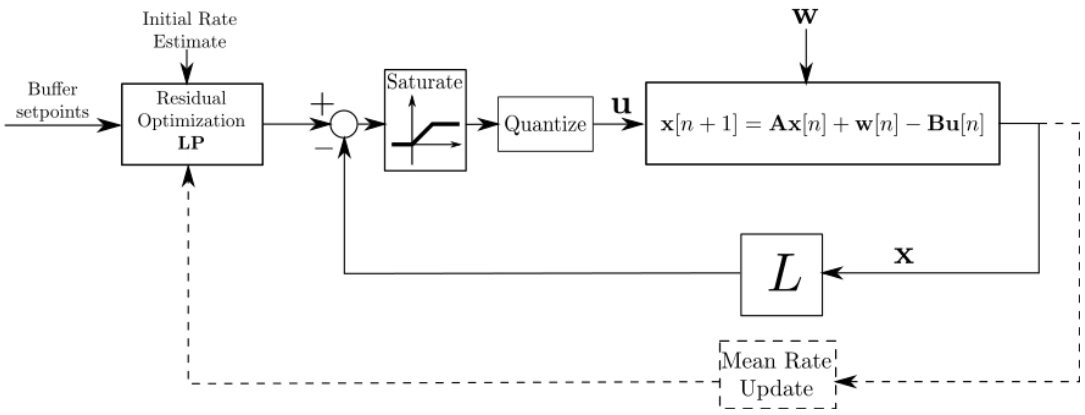

Fig. 4. Overview of our queue backlog management design. The required backlog level enters as a reference to a feedback system with bounded-sum, quantized inputs and additive non-zero mean Gaussian noise. We note that although in our case the mean of incoming noise vector $\mathbf{w}$ is known in advance, it can be also estimated by the controller as shown in the outermost loop.

Remark 1. In step 5 in Algorithm 1 one might presume that the second term $B_j^T(\mathbf{E}\lambda\tilde{q}_i)_{\text{est}}$ requires all λ, but by the sparse structure of B_j it is easily seen that only λ_i for $i \in Q_j$ is needed to form the product $B_j^T\lambda$.

IV. Case Study

A. Model overview

We now consider a scenario where autonomous vehicles pick up ready-product palettes from queues in front of production lines and deliver them to stretch-wrapping stations for further processing. These stations lie across the queues as shown in Fig. 5 and process palettes at a very fast rate. Vehicles, are only limited to operating on a subset of the queues.

We assume that there exists a vehicle routing algorithm with time windows [2] which guarantees that vehicle j can pick up at least C_j palettes per unit time from any of its assigned queues and deliver them to its respective station.

A nonzero-mean random number $w_i[n]$ of palettes is generated at each queue i at time period n, and each vehicle j picks up $u_{jk}[n]$ palettes from queue k at the same time (the third subscript index is not needed here).

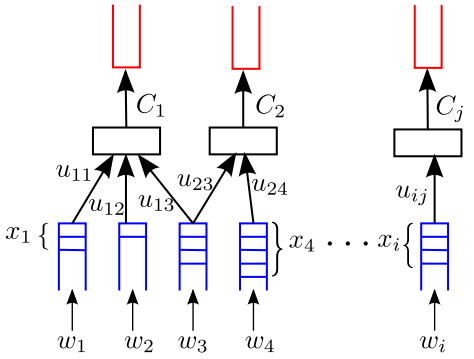

Fig. 5. Production queues (in blue) with their servers delivering palettes to stretch-wrapping lines (in red).

The goal is to use our scheme to keep the number of palettes in each queue from varying above their capacity, as this requires halting of the production line and manual emptying of the queues, a very costly process.

B. Numerical Example

For a numerical example, we consider 4 production queues, each holding a maximum of 20 items. If a queue reaches this maximum, the production to that queue is stopped and restarted after the queues are manually emptied, something which is quite costly to the process. Note that this procedure is not shown in the simulation plots where we allow the queues to overflow for demonstration purposes.

Each queue is driven by the production noise $w \in \mathcal{N}(6,2)$. The queues are served by 6 servers, where each server has a capacity of 5 units per time sample.

The servers are allocated to the queues according to: Queue i is served by server i, $i+1$ and $i+2$. With this scheme the sets $\mathcal{S}_i$ become

$$\mathcal{S}_i = \{i, i+1, i+2\} \quad 1 \le i \le 4$$

and sets Q_j are

$$\begin{array}{lll} Q_1 = \{1\} & Q_2 = \{1,2\} & Q_3 = \{1,2,3\} \\ Q_4 = \{2,3,4\} & Q_5 = \{3,4\} & Q_6 = \{4\} \end{array}$$

With this allocation setup and the knowledge of the noise statistic of the production rate, the nominal operating point for the servers can be determined using the method described in section III-A. The nominal operating points were only calculated once and are

$$\begin{array}{ll} \sigma_1 = \{3.5\} & \sigma_2 = \{1.60, 2.46\} \\ \sigma_3 = \{0.90, 1.77, 1.77\} & \sigma_4 = \{1.77, 1.77, 0.90\} \\ \sigma_5 = \{2.46, 1.60\} & \sigma_6 = \{3.5\} \end{array}$$

When the nominal operating point is determined, the online iterative gradient descent method for ILC can be used to minimize the variance of the number of units on the queues.

The cost that will be considered is

$$\mathbf{E}\left(\sum_{1\le i\le N_q}|\tilde{q}_i(t)|^2+\sum_{1\le j\le N_v}\left[\left|\sum_{i\in Q_j}\tilde{u}_{ij}(t)\right|^2+\varepsilon\left|\tilde{u}_{jj}(t)\right|^2\right]\right)$$

where $\varepsilon = 10^{-4}$ is to insure that the resulting weight matrix for the control signals, R, is positive definite. The reason of summing the effort of each server over the queues it serves, is that the variance of the servers total effort is to be minimized.

Following the method in section III-B, the length of the time interval between updates of the feedback matrix is set to be 5 time samples ($\Delta n_k = n_{k+1} - n_k = 5$ for all k). The initial feedback matrix is set to be close zero, meaning that there is in principal no feedback in the beginning. The result of simulating the system for 300 time sample is shown in Fig. 6 together with the result of the same system simulated without feedback, i.e. only using the nominal control.

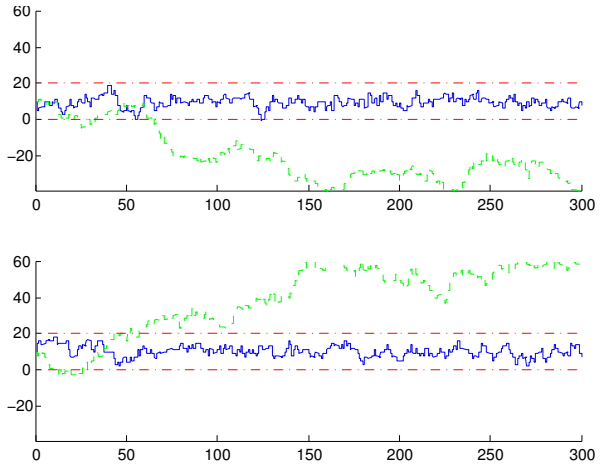

Fig. 6. The number of items on each queue when controlling the system with feedback is shown in blue and without feedback is shown in green for the first two queues of the system. Notice the costly overflows that occur throughout the feedback-less case.

We see large improvement when introducing feedback. Worth noting is that the simulation with feedback actually starts without feedback, but before it can be seen in the plots, the gradient method is able to determine a feedback matrix which improves the performance. A large cost occurs when the non-feedback regulation goes above the queue capacity (here 20 items), where the production has to be stopped and the queues emptied.

We also compared our approach to a one where we used a longest-queue-first algorithm, i.e.. the servers allocate their capacity so that the current largest residual from the reference is minimized. Although the results where quite similar, the computational time required by the longest-queue-first method was approximately 100 times larger than our method since it required the solution of a linear program at each time step.

V. Conclusion

We have outlined a comprehensive method for load balancing on queue networks, which using only local information regulates the variance of the networked buffer contents. It is a novel approach which uses a classical method from control engineering, namely LQG regulation, it is very fast computationally and distributed. With unbounded buffer and server capacity our method guarantees suboptimality bounds [11]. When the bounds are in place, our proposed quasi-static re-allocation of server bandwidths helps in reducing server overflow during variance regulation.

In future work we will look at the stability analysis of the quantized version of the scheme as well as stochastic analysis of the expected queue sizes. Currently the Q and R matrices are tuned in order to minimize overflow of the server capacities. For handling the server capacity constraints more rigorously we will also look at receding horizon approach, adapting work from [8].

References

[1] Kevin L. Burgess and Kevin M. Passino. Stability analysis of load balancing systems. In *American Control Conference, 1993*, pages 2415 –2419, 2-4 1993.

[2] Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, and Franois Soumis. Chapter 2 time constrained routing and scheduling. In C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35 – 139. Elsevier, 1995.

[3] S. Dhakal, B.S. Paskaleva, M.M. Hayat, E. Schamiloglu, and C.T. Abdallah. Dynamical discrete-time load balancing in distributed systems in the presence of time delays. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 5, pages 5128 – 5134 Vol.5, 9-12 2003.

[4] Antonis Dimakis and Jean Walrand. Sufficient conditions for stability of longest-queue-first scheduling: second-order properties using fluid limits. *Adv. in Appl. Probab*, 38(2):505–521, 2006.

[5] P. Elias, A. Feinsteini, and C. E. Shannon. A note on the maximum flow through a network. *IRE Transactions on Information Theory IT 2*, pages 117–119, 1956.

[6] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[7] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 698–707, New York, NY, USA, 1993. ACM.

[8] Pontus Giselsson and Anders Rantzer. Distributed model predictive control with suboptimality and stability guarantees. In *Proceedings of the 49:th IEEE Conference on Decision and Control 2010*, Atlanta, GA, USA, December 2010. Accepted for publication.

[9] K. Kumaran and D. Mitra. Performance and fluid simulations of a novel shared buffer management system. In *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1449 –1461 vol.3, 29 1998.

[10] Karl Mårtensson and Anders Rantzer. Gradient methods for iterative distributed control synthesis. In *Proceedings of the 48th IEEE Conference on Decision and Control*, Shanghai, China, December 2009.

[11] Karl Mårtensson and Anders Rantzer. Sub-optimality bound on a gradient method for iterative distributed control synthesis. In *Proc. 19th International Symposium on Mathematical Theory of Networks and Systems*, Budapest, Hungary, July 2010.

[12] G. Notarstefano and F. Bullo. Network abstract linear programming with application to cooperative target localization. In A. Chiuso, L. Fortuna, M. Frasca, L. Schenato, and S. Zampieri, editors, *Modelling, Estimation and Control of Networked Complex Systems*, Understanding Complex Systems, pages 177–190. 2009.

[13] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *Automatic Control, IEEE Transactions on*, 37(12):1936 –1948, dec 1992.