On Buffer Centering for Bittide Synchronization

Sanjay Lall[1] Călin Caşcaval[2] Martin Izzard[2] Tammo Spalink[2]

Abstract

We discuss distributed reframing control of bittide systems. In a bittide system, multiple processors synchronize by monitoring communication over the network. The processors remain in logical synchrony by controlling the timing of frame transmissions. The protocol for doing this relies upon an underlying dynamic control system, where each node makes only local observations and performs no direct coordination with other nodes. In this paper we develop a control algorithm based on the idea of reset control, which allows all nodes to maintain small buffer offsets while also requiring very little state information at each node. We demonstrate that with reframing, we can achieve separate control of frequency and phase, allowing both the frequencies to be syntonized and the buffers to be moved the desired points, rather than combining their control via a proportional-integral controller. This offers the potential for simplified boot processes and failure handling.

1 Introduction

The Google bittide system is designed to enable synchronous execution at large scale without the need for a global clock. Synchronous communication and processing offers significant benefits for determinism, performance and utilization, and through simplification, robustness. Synchronous execution is used successfully in real time systems [3].

In bittide, synchronization is decentralized, as every node in the system adjusts its frequency based on the observed communication exchanges with its neighbors. This mechanism is a distributed dynamic feedback control system. The bittide system was first proposed in [11]. It defines a synchronous logical clock that is resilient to variations in physical clock frequencies. In [7, 8] we discuss a model for the dynamics, the *Abstract Frame Model* (AFM) and its implication for controlling node frequencies. This work makes use of the model and ideas developed in that work. In this paper we focus on one critical aspect of the design, that is control of the buffers that bittide links use to smooth out frequency variations.

[1]S. Lall is with the Department of Electrical Engineering at Stanford University, Stanford, CA 94305, USA, and is a Visiting Researcher at Google. `lall@stanford.edu`

[2]Călin Caşcaval, Martin Izzard, and Tammo Spalink are with Google.

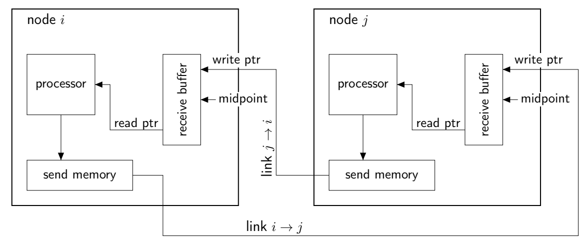

Figure 1: Two nodes showing the receive buffers as part of the links.

The bittide system relies on buffering frames on the receiving side of a link to absorb frequency oscillations in both directions. These buffers have two primary desirable properties. First, the control system must avoid overflow or underflow. Second, buffer size should be minimized, since buffer memory is system overhead. An initial transient period ensures read and write pointers are at buffer midpoints, after which their maximum needed size is determined by how much the control system can control variation without overflow or underflow.

Processes on bittide coordinate using ahead-of-time scheduling, and so frame retransmissions are not allowed. Therefore, the buffers need to ensure that no frames are lost. Note that this does not imply that links in the system cannot fail, simply that failures are dealt with at a different level. For the purpose of frame delivery, buffers should be considered reliable.

To satisfy these properties, we design a dynamic control system that switches modes after the initial transitory period. We trade-off convergence speed by using a proportional controller that allows buffer occupancy to stabilize. Buffers use virtual pointers (*i.e.*, frame counters) to provide the correct input to the controller. After the controller converges, the proportional controller is replaced with a proportional-plus-offset controller to drive the buffer occupancy to the midpoint, and retain the same converged frequency.

2 Prior work

Distributed systems have long striven to achieve synchrony at some level. This has been mostly effected by mechanisms that align local clocks to a global master such as a UTC server. The most common are the NTP [9] and PTP [6] standards. Telecom systems like SONET [12] and some packet networking equipment using SyncE [13] apply a further level of effort to aid synchrony, specifically, they work to syntonize local time-reference oscillators in hierarchical manner. Syntony here is used to mean that all reference oscillators in the distributed system maintain the same frequency on average over time. There are well-known methods to achieve this, mostly by clock extraction from upstream communication links. Having syntonized oscillators eases the task of aligning local clocks to a master.

The bittide system also uses syntonization, although there is no inherent need for hierarchy, and instead bittide adds precise phase control at the lowest level. This phase control is not quite phase alignment across the distributed system because there will be temporal *wobble*. The bittide system uses small elastic buffers to absorb this wobble. This opens the possibility of very precise coordination across a bittide distributed system, coordination equivalent to what is possible in a traditional synchronous system.

A model for the dynamics of a bittide system was developed in [7, 8]. In this paper we focus on a simplified linear version of this model. Many similar linear consensus models have been extensively studied; see [5] for a survey focused on synchronization, and see [10] for a discussion of stability analysis.

This paper develops an approach to control of bittide systems we call *reframing* of a bittide system. This is related to the idea of *reset* which has a long history in control. Before the development of integral control, in order to ensure that a system achieved a small steady-state error, the offset (or *reset*) parameter in a proportional controller was adjusted manually. Integral control was developed as a way of automatically performing such resets [1]. Here, we use a variant of the reset idea for a *distributed* system, to simultaneously control *two* quantities per node, frequency and buffer occupancy. Because the system being controlled is a computer network, it has sufficiently ideal properties (such as conservation of frames [7]) that we can perform a reset exactly once, at bootup. The idea of reset is attributed [2] to Mason in the 1930s.

3 Notation and preliminaries

We represent the bittide topology as a directed graph with n nodes and m edges. Define the source incidence matrix $S \in \mathbb{R}^{n\times m}$ by

$$S_{ie} = \begin{cases} 1 & \text{if node } i \text{ is the source of edge } e \\ 0 & \text{otherwise} \end{cases}$$

and the destination incidence matrix $D \in \mathbb{R}^{n\times m}$ by

$$D_{ie} = \begin{cases} 1 & \text{if node } i \text{ is the destination of edge } e \\ 0 & \text{otherwise} \end{cases}$$

The usual incidence matrix of the graph is then $B = S - D$. Let $\mathbf{1}$ be the vector of all ones, then $B^{\mathsf{T}}\mathbf{1} = 0$.

A directed graph is called ***strongly connected*** or ***irreducible*** if for every i, j there exists directed paths $i \to j$ and $j \to i$. Suppose $A \in \mathbb{R}^{n\times n}$ is a nonnegative matrix such that $A_{ij} > 0$ if there is an edge $i \to j$ and $A_{ij} = 0$ otherwise. The matrix A is called irreducible if the corresponding graph is irreducible. Note that this does not depend on the diagonal elements of A.

A matrix $Q \in \mathbb{R}^{n\times n}$ is called ***Metzler*** if $Q_{ij} \geq 0$ for all $i \neq j$. A Metzler matrix Q is called a ***rate matrix*** if its rows sum to zero. If Q is Metzler and irreducible, then there is a real eigenvalue λ_{metzler}, with positive left and right eigenvectors. All other eigenvalues λ satisfy $\Re(\lambda) < \lambda_{\text{metzler}}$.

A Metzler matrix Q has a nonnegative matrix exponential. To see this, let $s > 0$ be such that $sI + Q \geq 0$. Then $e^Q = e^{-sI}e^{sI+Q}$ and both terms on the RHS are elementwise nonnegative. For the special case of a rate matrix Q, since $Q\mathbf{1} = 0$ we have directly $e^Q\mathbf{1} = \mathbf{1}$ and so e^Q is a stochastic matrix.

4 The bittide control system

A detailed model of the bittide system, called the *abstract frame model* is developed in [7]. A simplified finite-dimensional linear time-invariant model was presented in [8]. In both papers, we considered the special case of a system where all links are bidirectional. We now update the model to include unidirectional links. We focus on the case where the controller is continuous-time, latencies are small, and measurements are unquantized. The effectiveness of this approximation was investigated in [7], so we do not dwell here on this issue.

The simplified fundamental dynamics of the bittide system are as follows.

$$\dot{\theta}_i(t) = \omega_i(t) \tag{1}$$

$$\beta_{j\to i}(t) = \theta_j(t) - \theta_i(t) + \lambda_{j\to i} \tag{2}$$

$$\omega_i(t) = \omega_i^{\text{u}} + c_i(t) \tag{3}$$

Here $i, j \in 1, \ldots, n$ index nodes in the graph, and $j \to i$ refers to an edge from j to i. The variable θ_i is the *clock phase* at node i, whose time-derivative $\omega_i(t)$ is the clock frequency. The clock frequency is the sum of two terms, the first is the constant ω_i^{u}, which is the *uncontrolled frequency* of the clock. It is unknown, and not available to the bittide control system. The second term is c_i, the *frequency correction*, which is the control input; it is chosen by the controller at node i. Equation (2) gives $\beta_{j\to i}$, the occupancy of the elastic buffer at node i associated with the edge $j \to i$. The quantity $\lambda_{j\to i}$ is a constant associated with the link.

Control for the bittide system is inherently distributed and as such, does not have access to global information.

At each node i, the controller measures all of the occupancies $\beta_{j \to i}$ for all of the incoming links. It cannot observe the occupancies at other nodes, nor can it observe t or $\dot{\theta}_i$. Using this limited information, it chooses the frequency correction c_i. Dynamic controllers cannot be implemented exactly, since the controller does not know the time t, and so, for example, the controller cannot exactly perform any integration or differentiation. The only clock information available at node i is the clock phase θ_i, and in general this varies from one node to the next. This means that we are desirous of implementing a purely static controller, such as a proportional (plus-offset) controller. One form, which has been studied in [8], is

$$c_i(t) = k \sum_{j \mid j \to i} (\beta_{j \to i} - \beta_i^{\text{off}}) + q_i(t) \tag{4}$$

The buffer occupancy $\beta_{j \to i}$ is measured relative to an offset β_i^{off}, corresponding to the desired equilibrium buffer occupancy. The difference $\beta_{j \to i} - \beta_i^{\text{off}}$ is called the *relative* buffer occupancy. The controller chooses the correction to be proportional to the sum of the relative buffer occupancies at the node, plus a constant frequency offset q_i. The controller parameter k could in principle depend on the node i, but for simplicity and scalability we do not consider that case. As we discuss in this paper, the frequency offset $q_i(t)$ may vary with both time and node.

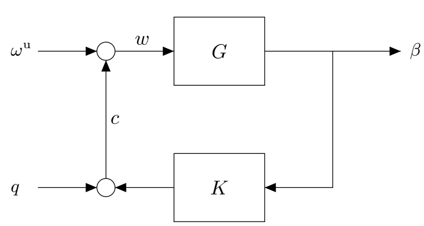

Figure 2: Feedback block diagram

The controller interconnection is illustrated in Figure 2. The system model G maps frequency ω to buffer occupancy β, and the controller K maps β to correction c minus the offset q.

4.1 Model

The model for the closed-loop system is described in vector form as follows:

$$\begin{aligned} \dot{\theta} &= \omega^{\text{u}} + c \\ \beta &= B^{\mathsf{T}} \theta + \lambda \\ c &= kD(\beta - \beta^{\text{off}}) + q \end{aligned} \tag{5}$$

Here $\beta, \lambda \in \mathbb{R}^m$ and $\theta, c, q \in \mathbb{R}^n$. It's convenient to write this as

$$\begin{aligned} \dot{\theta} &= A\theta + \omega^{\text{u}} + q + r \\ \beta &= B^{\mathsf{T}} \theta + \lambda \\ c &= A\theta + q + r \end{aligned} \tag{6}$$

where

$$A = kDB^{\mathsf{T}} \qquad r = kD(\lambda - \beta^{\text{off}}) \tag{7}$$

Note that the matrix A is not Hurwitz, and so θ does not converge.

When the system is booted up, the offsets are chosen to be *feasible*, that is we set $\beta^{\text{off}} = \beta(t^0)$ at some time t^0. This has the following consequence.

Lemma 1. *Suppose β^{off} is feasible, that is, there exists t^0 such that*

$$\beta^{off}(t^0) = B^{\mathsf{T}} \theta(t^0) + \lambda$$

Let r be given by equation (7). Then $r \in \operatorname{range}(A)$.

Proof. This holds because $r = kD(\lambda - \beta^{\text{off}})$ and so $r = -kDB^{\mathsf{T}}\theta(t^0) = -A\theta(t^0)$. ■

We assume the graph is irreducible. Then the matrix A is a scaled directed Laplacian matrix for the graph. It is an irreducible rate matrix. The following result is standard.

Lemma 2. *Suppose A is an irreducible rate matrix, and let $z > 0$ be it's Metzler eigenvector, normalized so that $\mathbf{1}^{\mathsf{T}} z = 1$. Then*

$$\lim_{t \to \infty} e^{At} = \mathbf{1} z^{\mathsf{T}}$$

Proof. Since A is irreducible, the Metzler eigenvalue, which is zero, has multiplicity one. Let the eigendecomposition of A be $AT = TD$. Then we have

$$D = \begin{bmatrix} 0 & 0 \\ 0 & \Lambda \end{bmatrix} \quad T = \begin{bmatrix} \mathbf{1} & T_2 \end{bmatrix} \quad T^{-1} = \begin{bmatrix} z^{\mathsf{T}} \\ V_2^{\mathsf{T}} \end{bmatrix}$$

for appropriate matrices T_2, V_2, and Λ. The other eigenvalues of A all have negative real part. Since $T^{-1}T = I$ we have $\mathbf{1}^{\mathsf{T}} z = 1$. Then

$$e^{At} = T \begin{bmatrix} 1 & 0 \\ 0 & e^{\Lambda t} \end{bmatrix} T^{-1}$$

Taking the limit gives the result. ■

We denote by $z > 0$ the Metzler left-eigenvector of A, normalized so that $\mathbf{1}^{\mathsf{T}} z = 1$, and let $W = \mathbf{1} z^{\mathsf{T}}$. The matrix W is called the spectral projector of the Metzler eigenvalue. It satisfies $W^2 = W$ and in addition $WA = AW = 0$.

5 Approach

There are many important practical requirements that the controller must meet, which are discussed in [7]. In this paper, we focus on two critical requirements. The first is that the controller must ensure that the frequency of all nodes converges to the same value, that is, for some $\bar{\omega} > 0$, we have

$$\lim_{t \to \infty} \dot{\theta}_i(t) = \bar{\omega} \quad \text{for all } i$$

The second requirement is that, after some initial startup time T, the buffer occupancies remain close to the offset, that is, $|\beta_i(t) - \beta_i^{\text{off}}|$ should be small for all i and all $t > T$. Both of these requirements are specifications of allowed steady-state behavior.

One approach to control this system is to use an approximate proportional-integral (PI) control, as discussed in [7, 8]. Even in situations where the effects of the approximation are small, there are potential disadvantages to the PI controller. One is that the PI controller contains the integral state, which the controller may need to set carefully when a node starts up and when neighboring nodes fail. In the distributed setting of bittide, one of the design tenets is to avoid in-band signaling for controlling synchronization, and thus we have no mechanism to exchange such information. Appropriate choice of the integral gain may be affected by the underlying network topology, link rates, and link latencies.

Another alternative method to proportional-integral control is simply using a very large gain k. Very large gains have negative consequences for feedback system behavior in several well-known ways. In particular for bittide, this would adversely affect delay robustness and response to quantization noise, both of which are important in this setting. We therefore develop an alternative approach in this paper.

Reframing control. We give here for convenience a brief summary of the technical approach, which is detailed precisely in Section 6. Our approach is to make use of the offset term q in the proportional-plus-offset controller. At each node, the local controller sets $q = 0$ initially. The system frequency ω will then converge, so that all nodes have the same frequency, a weighted average of ω^{u}. This is shown in Lemma 4.

The buffer occupancy also converges, as shown in Lemma 5. Typically the buffer occupancy will not converge to the buffer offset, because a nonzero correction c given by equation (8) is necessary to maintain frequency equilibrium, and with $q = 0$ the controller (4) can only achieve this with a nonzero relative occupancy $\beta - \beta^{\text{off}}$.

After this initial period of convergence, the correction has converged to a steady-state value c^{ss}. The controller now performs a *reframing*; it sets the offset q in the controller to c^{ss}. The control signal emitted by the controller is now not an equilibrium solution, and so the system will need to reconverge to a new equilibrium. The key point here is that the new equilibrium is at the same *frequency*, but a different *buffer occupancy*. We show in Lemma 7 that, after the reframing, the frequency converges to the same weighted equilibrium value as before. Furthermore, Lemma 8 shows that after this second phase, the buffer occupancy $\beta(t)$ converges to the midpoint β^{off}.

By using this two-phase approach, we can therefore achieve *both* controller requirements. The steady-state frequency is exactly the same as that achieved by the proportional controller, and the final buffer occupancy is at the desired offset point.

6 Main results

We first analyze simple convergence. The clock phase θ does not converge, so there is no steady-state value for it. However, the frequency $\omega(t) = \dot{\theta}(t)$ does converge. We will make use of the following simple property.

Lemma 3. *Let A be an irreducible rate matrix, and $\dot{\theta}(t) = A\theta(t) + v$. Let W be the spectral projector corresponding to the Metzler eigenvalue. Then*

$$\lim_{t\to\infty} A\theta(t) = (W - I)v$$

for any initial conditions $\theta(0)$.

Proof. We have

$$\theta(t) = \int_0^t e^{A(t-s)} v \, ds + e^{At}\theta(0)$$

and hence

$$A\theta(t) = (e^{At} - I)v + Ae^{At}\theta(0)$$

Taking the limit gives the desired result. ∎

Define for convenience the function $F : \mathbb{R}^n \to \mathbb{R}^n$ by

$$F(q) = (W - I)\omega^{\text{u}} + W(q + r)$$

This function maps the controller offset q to the steady-state correction, as follows.

Lemma 4. *Consider the dynamics of (6). For any initial $\theta(0)$ and any $q \in \mathbb{R}^n$ we have*

$$\lim_{t\to\infty} c(t) = F(q) \tag{8}$$

Proof. This is a direct consequence of Lemma 3. ∎

The frequency of the system is $\omega(t) = c(t) + \omega^{\text{u}}$ which gives steady-state frequency

$$\omega^{\text{ss}} = \lim_{t\to\infty} \omega(t) = W(q + r + \omega^{\text{u}})$$

Hence, before reframing, the frequency converges. The frequency that the system converges to meets the first performance requirement of bittide, since we have $W = \mathbf{1}z^{\mathsf{T}}$ and therefore we must have all components of ω^{ss} equal. Since $\mathbf{1}^{\mathsf{T}}z = 1$ the steady-state frequency is a convex combination of the entries of $q + r + \omega^{\text{u}}$. From Lemma 1, we have $r \in \text{range}(A)$. Then

$$\omega^{\text{ss}} = W(q + \omega^{\text{u}})$$

since $WA = 0$. Then we can see that a pure proportional controller (*i.e.*, $q = 0$) results in all nodes of the bittide system converging to a steady-state frequency equal to the weighted average $z^{\mathsf{T}}\omega^{\mathrm{u}}$, that is

$$\omega^{\mathrm{ss}} = W\omega^{\mathrm{u}} = \mathbf{1}z^{\mathsf{T}}\omega^{\mathrm{u}} \tag{9}$$

The steady-state frequency is affected by q, and so even though this frequency meets the bittide requirement, after reframing we will adjust q, and so it appears in principle possible that the frequency of the system could change. We will see below that, although at the time of reframing the frequency of the system does change, it nonetheless returns and converges to the same frequency. An example of this behavior is given in Figure 4. The first bittide requirement is satisfied; for any positive gain k the nodes all converge to the same steady-state frequency. We now turn attention to the second requirement, ensuring that eventually β stays close to the offset β^{off}. We first show that, before reframing, the buffer occupancy converges.

Lemma 5. *Consider the dynamics of* (6). *For any initial* $\theta(0)$ *and any* $q \in \mathbb{R}^n$, *the buffer occupancy* $\beta(t)$ *converges as* $t \to \infty$.

Proof. Using the same notation as in the proof of Lemma 2, we have

$$\int_0^t e^{A(t-s)}\,ds = T\begin{bmatrix} t & 0 \\ 0 & \Lambda^{-1}(e^{\Lambda t} - I)\end{bmatrix}T^{-1}$$

Then

$$\begin{aligned}\beta(t) &= \lambda + B^{\mathsf{T}}\theta(t) \\ &= \lambda + B^{\mathsf{T}}T\begin{bmatrix} 0 & 0 \\ 0 & \Lambda^{-1}(e^{\Lambda t} - I)\end{bmatrix}T^{-1}(\omega^{\mathrm{u}} + q + r) \\ &\qquad + B^{\mathsf{T}}T\begin{bmatrix} 0 & 0 \\ 0 & e^{\Lambda t}\end{bmatrix}T^{-1}\theta(t)\end{aligned}$$

where we have used the fact that the first column of T is $\mathbf{1}$ and $B^{\mathsf{T}}\mathbf{1} = 0$. Then

$$\lim_{t\to\infty}\beta(t) = \lambda - T_2\Lambda^{-1}V_2^{\mathsf{T}}(\omega^{\mathrm{u}} + q + r) \qquad \blacksquare$$

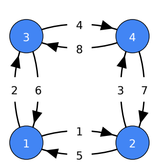

Figure 3: Graph used to generate the simulations of Figures 4 and 5.

We now turn to the reframing. The controller runs a proportional controller for some amount of time T_1, long enough to ensure that, in practice, the frequency and the buffer offsets have converged. After that time, the controller changes to using a non-zero offset, which is simply equal to the converged value of the correction. This controller is stated formally below.

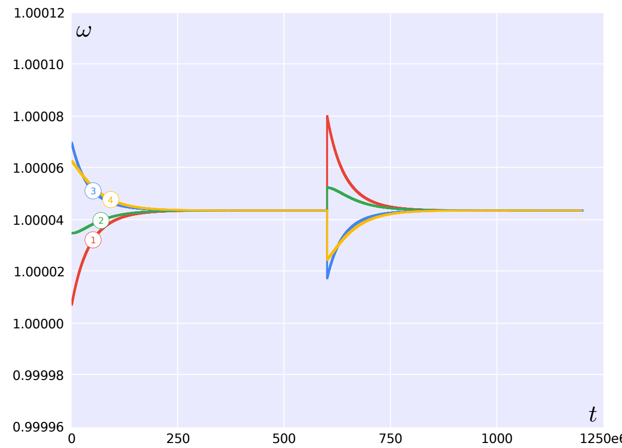

Figure 4: Frequency behavior as a function of time. The reframing occurs at about $t = 600$ns. Simulations were performed using the `Callisto` [4] simulator with a detailed frame-accurate model of bittide. Traces on the graph are labeled with the corresponding node number.

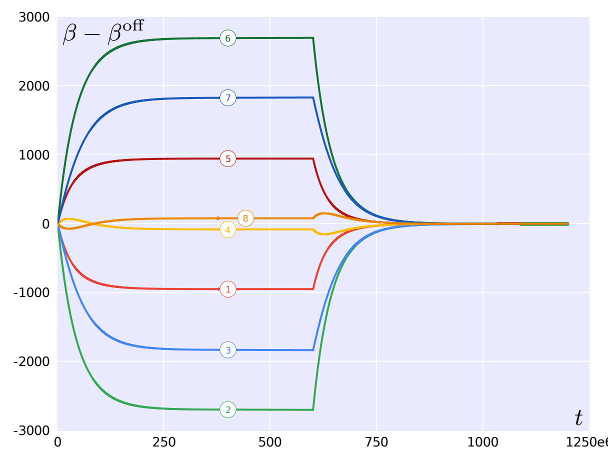

Figure 5: Relative buffer occupancies $\beta - \beta^{\mathrm{off}}$ for the same simulation as in Figure 4. Traces on the graph are labeled with the corresponding edge number.

Definition 6. *We define the* ***reframing controller*** *as follows. For some* $T_1 > 0$, *let the correction be*

$$c(t) = \begin{cases} kD(\beta(t) - \beta^{\mathit{off}}) & \text{for } t \le T_1 \\ kD(\beta(t) - \beta^{\mathit{off}}) + kD(\beta(T_1) - \beta^{\mathit{off}}) & \text{otherwise}\end{cases}$$

Now we show the desired frequency convergence property. After reframing, the controller converges to the same frequency as that before reframing, in equation (9).

Lemma 7. *Suppose* β^{off} *is feasible. Using the reframing controller, as* $T_1 \to \infty$ *and* $t \to \infty$, *the frequency converges*

$$\omega(t) \to W\omega^u$$

Proof. Since we are considering both T_1 and t large, we can evaluate convergence in two phases. In the first phase we have $q = 0$ and so according to Lemma 4 we have $c(T_1) \to F(0)$. The reframing controller is

$$c(t) = kD(\beta(t) - \beta^{\text{off}}) + c(T_1) \qquad \text{for } t > T_1$$

We can therefore use Lemma 4 again, with $q = c(T_1)$, to give

$$\begin{aligned}\lim_{t\to\infty} c(t) &= F(F(0)) \\ &= F\big((W - I)\omega^{\text{u}} + Wr\big) \\ &= (W - I)\omega^{\text{u}} + W(r + (W - I)\omega^{\text{u}} + Wr) \\ &= (W - I)\omega^{\text{u}} + 2Wr \\ &= (W - I)\omega^{\text{u}}\end{aligned}$$

where the last line holds since β^{off} is feasible. Then since $\omega(t) = c(t) + \omega^{\text{u}}$ we have

$$\lim_{t\to\infty} \omega(t) = W\omega^{\text{u}}$$

as desired. ■

Finally, we turn to the critical requirement for bittide, that the buffer occupancies be kept close to the offset. The following result shows that, after the reframing, buffer occupancies return to the midpoint. This is illustrated by the simulation in Figure 5.

Lemma 8. *Suppose β^{off} is feasible. Using the reframing controller, as $T_1 \to \infty$ and $t \to \infty$, the buffer occupancy converges*

$$\beta(t) \to \beta^{off}$$

Proof. Denote $\beta^{\text{ss}} = \lim_{t\to\infty} \beta(t)$. With the reframing controller, we have

$$\lim_{t\to\infty} c(t) = kD(\beta^{\text{ss}} - \beta^{\text{off}}) + c(T_1)$$

The proof of Lemma 7 shows that $\lim_{t\to\infty} c(t) = c(T_1)$ and therefore

$$D(\beta^{\text{ss}} - \beta^{\text{off}}) = 0 \tag{10}$$

Now, since $r = 0$, we have $\beta^{\text{off}} = B^{\mathsf{T}}\theta(t^0) + \lambda$, and so

$$\beta(t) - \beta^{\text{off}} = B^{\mathsf{T}}(\theta(t) - \theta(t^0))$$

hence for all t we have $\beta(t) - \beta^{\text{off}} \in \text{range}(B^{\mathsf{T}})$. Hence $\beta^{\text{ss}} - \beta^{\text{off}} = B^{\mathsf{T}}x$ for some x, and so (10) means that $DB^{\mathsf{T}}x = 0$. Since DB^{T} is an irreducible rate matrix, this means $x = \gamma\mathbf{1}$ for some γ, and hence $B^{\mathsf{T}}x = 0$, from which we have $\beta^{\text{ss}} - \beta^{\text{off}} = 0$ as desired. ■

Lemmas 7 and 8 show that the reframing methodology achieves the requirements. In practice, this works because, before reframing, the buffers can overflow. This occurs during system startup, when the network frames do not contain any data, and so the system does not need to store the actual frames, and instead can use counters or pointers to keep track of how many frames *would* be in the buffer. After the reframing, and subsequent convergence, the bittide system can begin executing code, at which time it is essential that frames not be dropped. At this point, the buffer occupancies have returned to a stable equilibrium at the midpoint.

7 Conclusions

We proved that we can satisfy the requirements for controlling a bittide system and satisfy the desirable properties for buffer occupancy by developing a dynamic control system that resets after convergence. The initial proportional controller drives frequency convergence, and the proportional plus offset controller ensures that buffer occupancies are driven toward the desirable midpoint.

8 Acknowledgments

We thank Robert O'Callahan, Pouya Dormiani, Chase Hensel, and Chris Pearce for all of their work on this project, and for much collaboration and helpful discussion.

References

[1] K. J. Åström and R. M. Murray. *Feedback systems: an introduction for scientists and engineers.* Princeton university press, 2021.

[2] D. M. Auslander. Evolutions in automatic control. *Transactions of the ASME*, pages 4–9, March 1971.

[3] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, Jan. 2003.

[4] The Callisto simulator for bittide networks. https://github.com/bittide/Callisto.jl.

[5] F. Dorfler and F. Bullo. Synchronization in complex networks of phase oscillators: A survey. *Automatica*, 50(6):1539–1564, 2014.

[6] Precision clock synchronization protocol for networked measurement and control systems. IEEE standard 2021.9456762.

[7] S. Lall, C. Caşcaval, M. Izzard, and T. Spalink. Modeling and control of bittide synchronization. In *Proceedings of the American Control Conference*, pages 5185–5192, 2022.

[8] S. Lall, C. Caşcaval, M. Izzard, and T. Spalink. Resistance distance and control performance for bittide synchronization. In *Proceedings of the European Control Conference*, pages 1850–1857, 2022.

[9] D. L. Mills. Internet time synchronization: The network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, 1991.

[10] L. Moreau. Stability of continuous-time distributed consensus algorithms. In *Proceedings of the IEEE Conference on Decision and Control*, volume 4, pages 3998–4003, 2004.

[11] T. Spalink. *Deterministic sharing of distributed resources.* Princeton University, 2006.

[12] Synchronous optical network (SONET) transport systems: Common generic criteria. Telcordia GR-253.

[13] Timing and synchronization aspects in packet networks. ITU-T G.8261/Y.1361.