Google Research

# On Buffer Centering for Bittide Synchronization

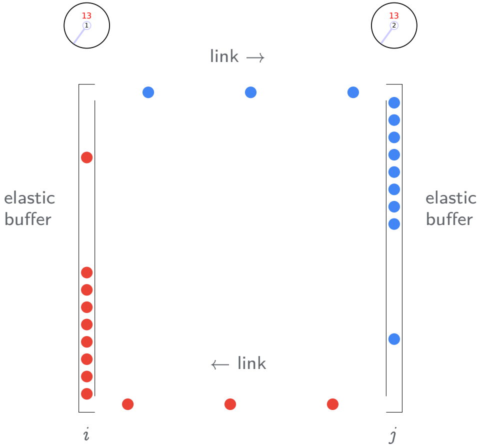
Sanjay Lall, Google Research and Stanford University
Călin Caşcaval, Martin Izzard, and Tammo Spalink, Google Research

# Overview

- *Google research project*
  - originated at Princeton: Spalink 2006.
  - broad scope: applications, scheduling, simulation, hardware, theory
- *paper*
  - reframing control
- *outline:* the mechanism, logical synchrony, controlling frequency

# Overview

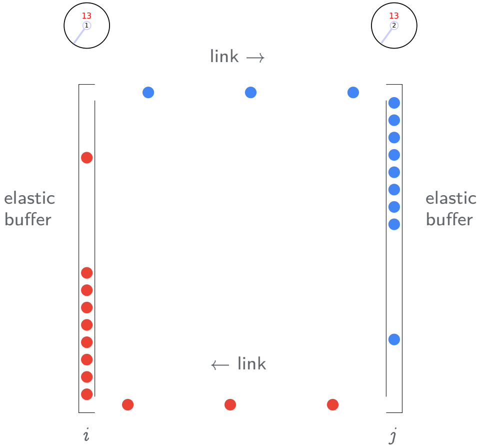
- at each node $i$
  - clock
  - each incoming link has a queue called the *elastic buffer*
- at each node, with each clock tick
  - a frame is removed from all elastic buffers
  - a frame is sent on all outgoing links



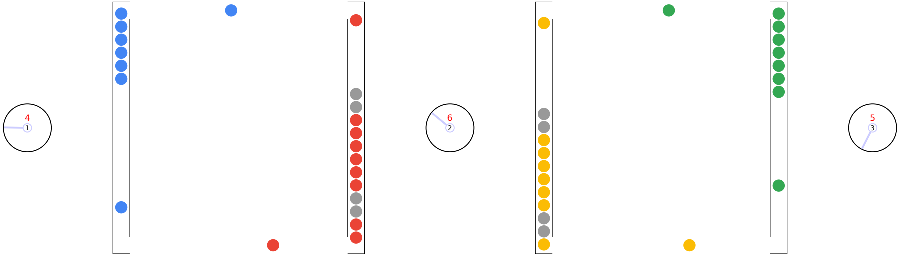

Google

# Mechanism

- if oscillator at node $j$ is faster than that at $i$
  - $j$'s elastic buffer will drain
  - $i$'s elastic buffer will fill
- nodes observe elastic buffers, adjust frequency



# Logical Synchrony




- marked frames from nodes $1$ and $3$ always arrive simultaneously at node $2$
- an example of *logical synchrony*
- does not require clocks to be synchronized

# Abstract frame model

# Modeling frames and phase

$$\frac{d\theta_i}{dt} = \omega_i$$

$$\beta_{i \to j}(t) = \lfloor \theta_i(t - l_{i \to j}) \rfloor - \lfloor \theta_j(t) \rfloor + \lambda_{i \to j}$$

- history of clock phases $\theta$ determines location of every frame
- $\lambda_{i \to j}$ is a constant, determined by clock offsets at boot
- buffer occupancy is (roughly) phase difference between clocks at each end of the link

# Control loop

$$\frac{d\theta_i}{dt} = \omega_i$$

$$\omega_i = c_i + \omega_i^{\mathrm{u}}$$

*$\omega_i^u$ is uncorrected frequency, unknown*

*$c_i$ is frequency correction*

$$\beta_{i \to j}(t) = \lfloor \theta_i(t - l_{i \to j}) \rfloor - \lfloor \theta_j(t) \rfloor + \lambda_{i \to j}$$

$$r_i = \sum_{j \mid j \sim i} (\beta_{j \to i} - \beta^{\mathrm{off}})$$

*$\beta_{j \to i}$ is buffer occupancy*

*$r_i$ is sum of buffer occupancies relative to offset*

$$c_i = k_P r_i$$

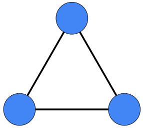
*proportional controller law*

# Control

# Control objectives

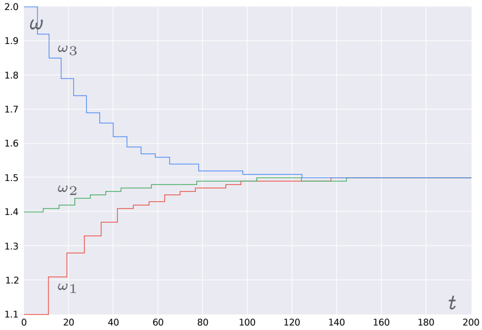
- frequencies cannot remain different for too long, otherwise buffers will over/underflow
- bittide performance requirement: maintain buffer occupancy within limits
- ideally buffer occupancies should be small, and frequencies large
- controller must be decentralized
- no *in-band* signalling
- failure handling, addition and removal of nodes, boot, *etc.*,

# Proportional control



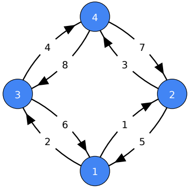
- $k_P = 0.01$
- $\texttt{latency} = 1.0$
- $\texttt{poll\_period} = 10$
- $\texttt{uncorrected\_frequency} = (1.1, 1.4, 2.0)$
- $\texttt{control\_delay} = 2$

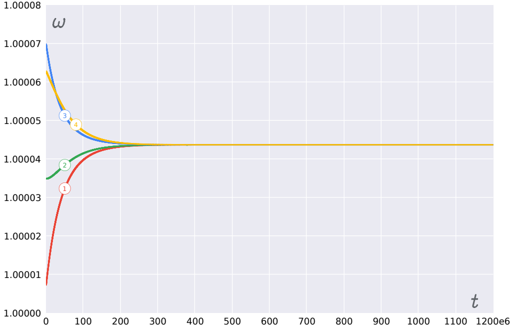
- varying step width is a consequence of periodic sampling w.r.t. the local clock

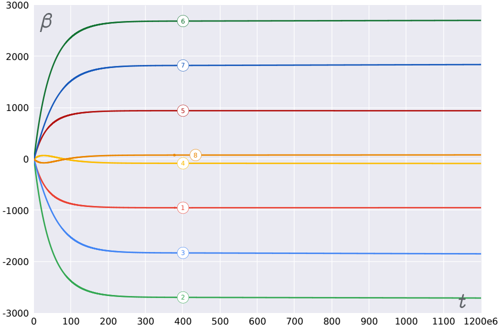


# Example: Proportional control

- frequency correction is proportional to sum of relative buffer offsets
- equilibrium buffer offsets nonzero

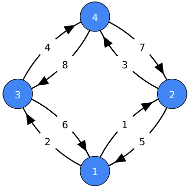


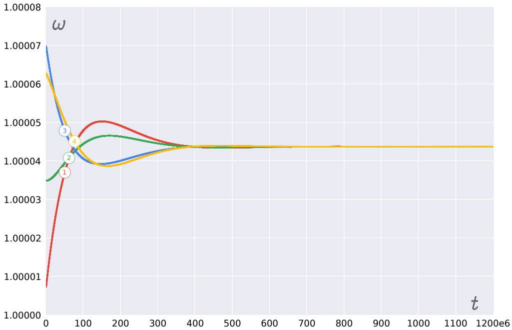



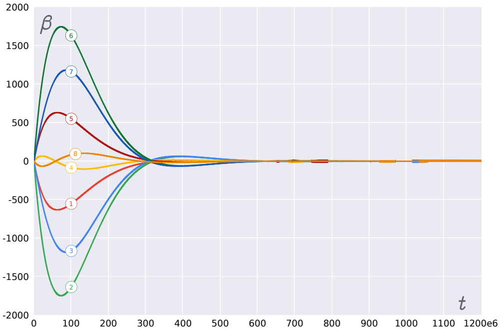



# Proportional-integral control

- ensures small steady state relative buffer occupancy



# Reset control

- first, apply proportional control

$$c_i = k_P \sum_{j \to i} \beta^{\text{rel}}_{i \to j}$$

- this converges $c_i(t) \to c_i^{\text{ss}}$
- at large time $T$ change to proportional-plus-offset

$$c_i = c_i^{\text{ss}} + k_P \sum_{j \to i} \beta^{\text{rel}}_{i \to j}$$

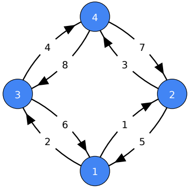
- immediately after the switch, nodes are at the wrong frequency
- system is stable, so will return to equilibrium, which happens when $c_i = c_i^{\text{ss}}$
- paper shows that for irreducible graphs with a linear model

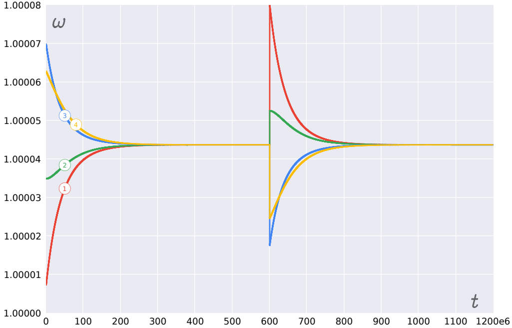
$$\sum_{j \to i} \beta^{\text{rel}}_{i \to j} \to 0$$

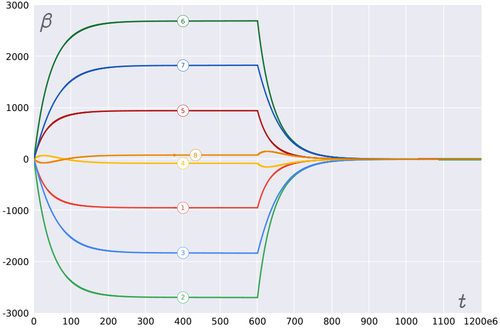
as reframing time $T$ and $t$ become large

# Reset

- controller shutdown at time $t \approx 4e9$
- at time $t \approx 6e9$, turn on the controller with new offset



# Soft reset

- first, apply proportional control

$$c_i = k_P \sum_{j \to i} \beta^{\text{rel}}_{i \to j}$$

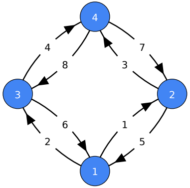
- this converges $c_i(t) \to c_i^{\text{ss}}$
- now *slowly* change to proportional-plus-offset

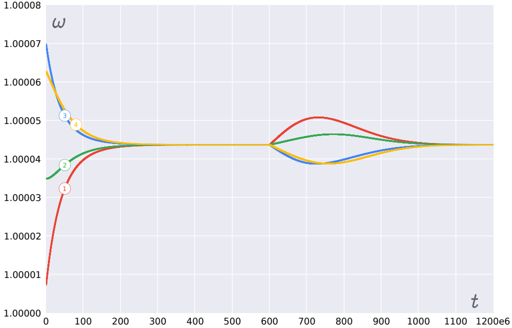
$$c_i = f(t) c_i^{\text{ss}} + k_P \sum_{j \to i} \beta^{\text{rel}}_{i \to j}$$

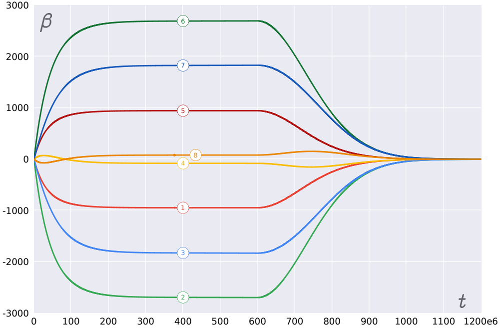
where $f(t)$ slowly changes from 0 to 1 over some interval

# Example: soft reset

- controller shutdown at time $t \approx 4e9$
- soft reset over approximate interval $[6e9, 1e10]$



# Summary

- soft reset
  - use P control initially
  - shutdown after convergence
  - turn on P control plus offset
- retains stability, keeps buffers at (or near) midpoint
- does not require integral control, spanning tree or global coordination

# Thank you!