Clock Synchronization for Interactive Music Systems

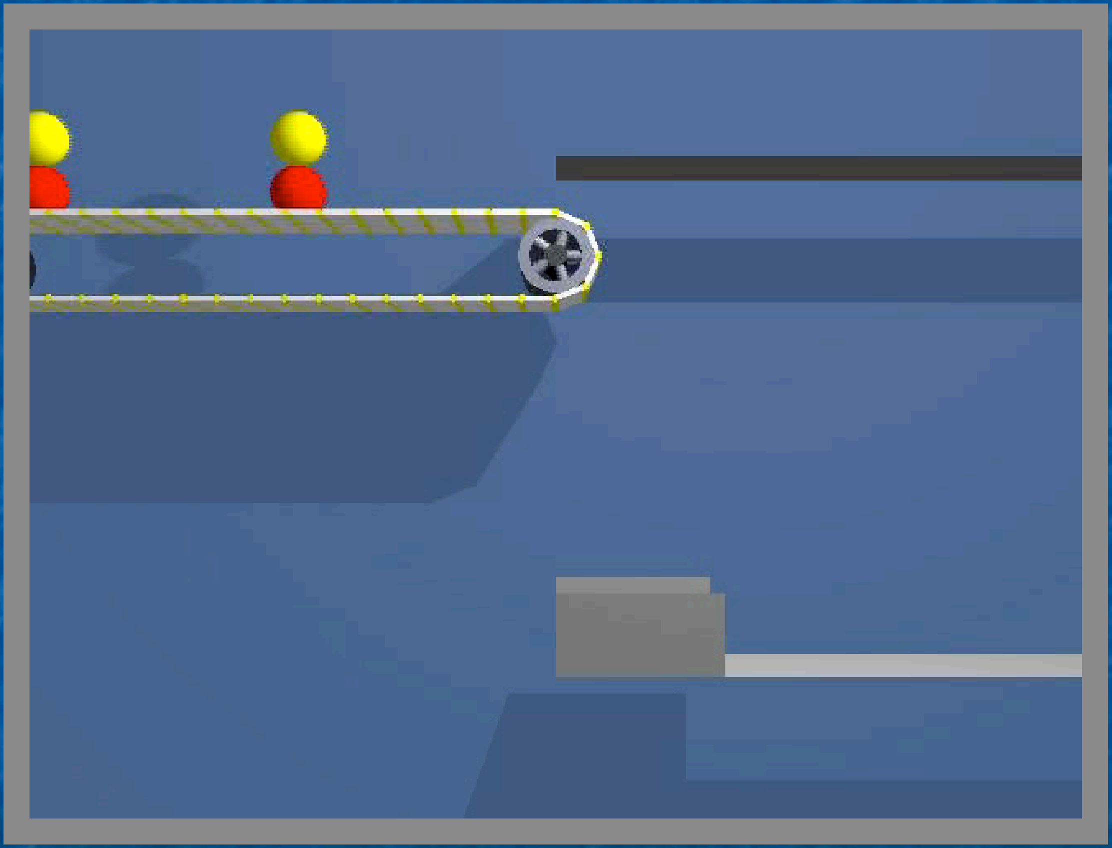
Roger B. Dannenberg
Carnegie Mellon University

Overview

- Why clock synchronization?
- Characterize the problem
- Simple solution
- Some more elaborate approaches
- What next?

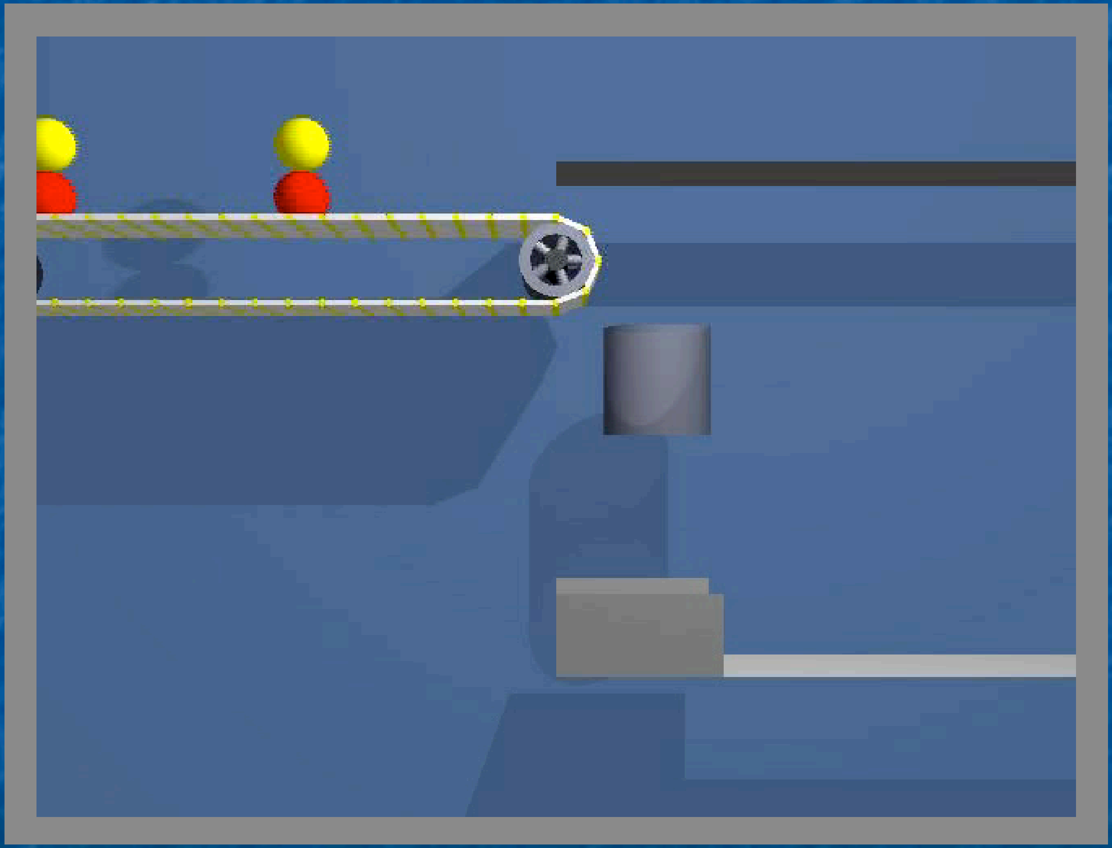
Why Clock Synchronization?

If you have low-latency communication, you do *not* need clock synchronization …



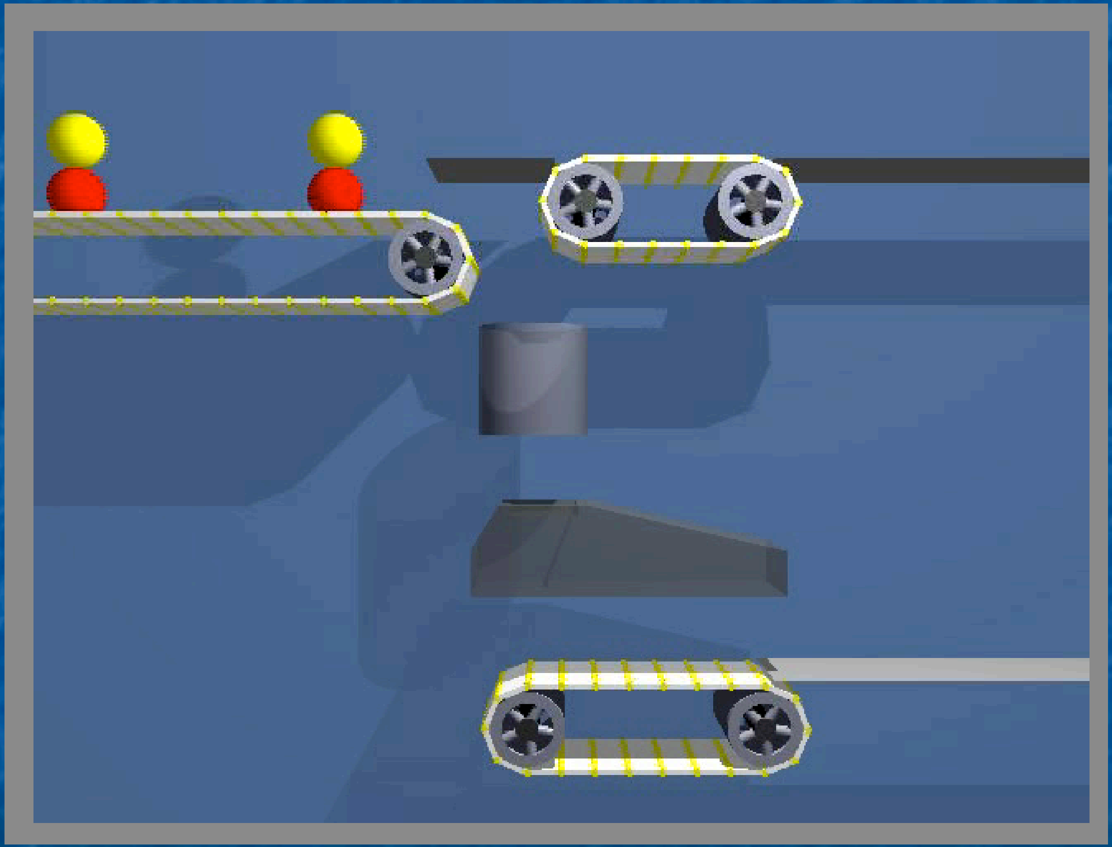
Why Clock Synchronization? (2)

If network communication *sometimes* has high delays (latency), then event synchronization is difficult…



Why Clock Synchronization? (3)

- Scheduling according to timestamps can overcome some synchronization problems (but not latency problems)…



Why Clock Synchronization? (4)

- Timestamps are only as good as the local clock…
- …therefore the goal is:
  Synchronize clocks to a precision that is much better than network latency and jitter.

The Design Space

- What do we synchronize to?
  - Global consensus (internal synchronization)
  - Master reference clock (external synch.)
- Who's in charge?
  - No one (symmetric)
  - Master (asymmetric, master-controlled)
  - Slave (asymmetric, slave-controlled)
- Special synchronization hardware?
  - Yes: hardware synchronization
  - No: software synchronization

Clock and Network Characteristics

- Crystal clock accuracy: +/-0.02%
- Frequency drift: low
- Network latency: <1ms
- Network jitter: long tail (0.5s)
- Jitter reading clock or frame #: <1ms

- This should be easy…

Network Latency and Jitter

- Interactive music systems
  - not compute bound
  - short or empty network and task queues
  - Messages *usually* get through quickly
- To read remote system time:
  - send message; wait for reply
  - quick reply => low latency and jitter
  - add half of transit time to compensate for latency
  - result should be well below 1ms error

Logical Clock Model

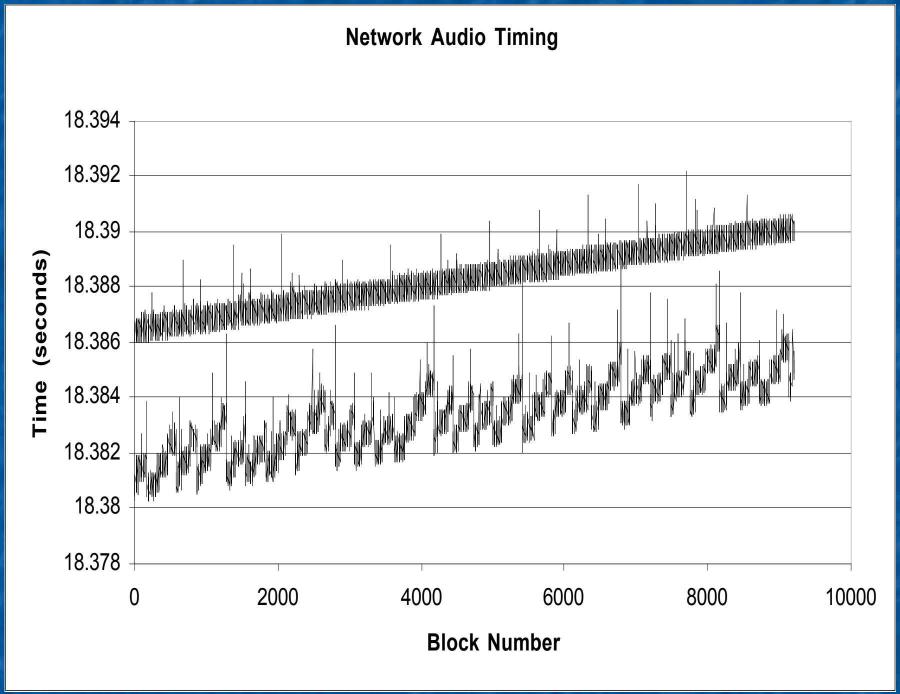
- Assume that time is a linear function of the local clock or sample count:

  *LogicalTime = offset + rate * LocalTime*

- Clock synchronization amounts to updating *offset* and *rate.*

Simple Solution

- Periodically read remote “master” clock
- If reply returns quickly, update local time
- Otherwise, continue with previous model until next period.



Audio block arrival time (lower), and block write time (upper).

More Elaborate Approaches

- Dominique Fober:
  - Use window of recent timestamp messages
  - Reject outliers, estimate offset and rate
  - Use exponential smoothing
- Brandt and Dannenberg:
  - Treat logical clock as feedback control system
  - In simulation, achieved 1.1ms clock error with 5ms error reading sample clock.

What Next?

- How do you handle dropped frames?
  - If time is measured in frames, time can jump.
  - You could inform the slaves when time jumps.
  - Or slaves could try to guess when time jumps.
  - In general, fast recovery is in conflict with stability and low error.
- How do you deal with unmatched sample rates?
  - Resample?
  - Ignore it and work at control level?

Conclusions

- Clock synchronization is critical for networked interactive systems
  - *Assuming that network latency is significant!*
- Clocks and networks have almost ideal properties.
- Simple approaches work well to ~1ms.
- Advanced techniques can achieve near-frame accuracy over ordinary networks.