Best Practices for Open Sound Control

Linux Audio Conference 2010 / Utrecht NL

Andrew Schmeder (presenting),
Adrian Freed,
David Wessel

Email Authors:
{andy,adrian,wessel}@cnmat.berkeley.edu

CNMAT / UC Berkeley
http://cnmat.berkeley.edu/

Overview:

- What is Open Sound Control?
- What does OSC practice include?
- Definition of audio control data, examples.
- Temporal quality assurance.
- Transport layer considerations.
- Description strategies for control data.
- Programming for audio control.

What is Open Sound Control?

Section 1

What is OSC?

- Open Sound Control (OSC) is a content format for messaging among computers, sound synthesizers, and other multimedia devices that are optimized for modern networking technology.
    - Wikipedia.org

What is OSC?

- A collection of ideas and practice for realtime audio control. Based around a descriptive document of the format and code (the "OSC-Kit") published by Matt Wright at CNMAT circa 2002.

- Now, lots of diverse implementations in applications and embedded software

- OSC, which is pronounced "oh-ess-cee", or sometimes "osk", stands for Open Sound Control.

- Actually not going to be called "Open Show Control" as of April 1st 2010.

What is Open?

- No license requirements
- No patented algorithms
- No conformance certification
- No strict specification of requirements
- Lots of open source code available

What is not Open?

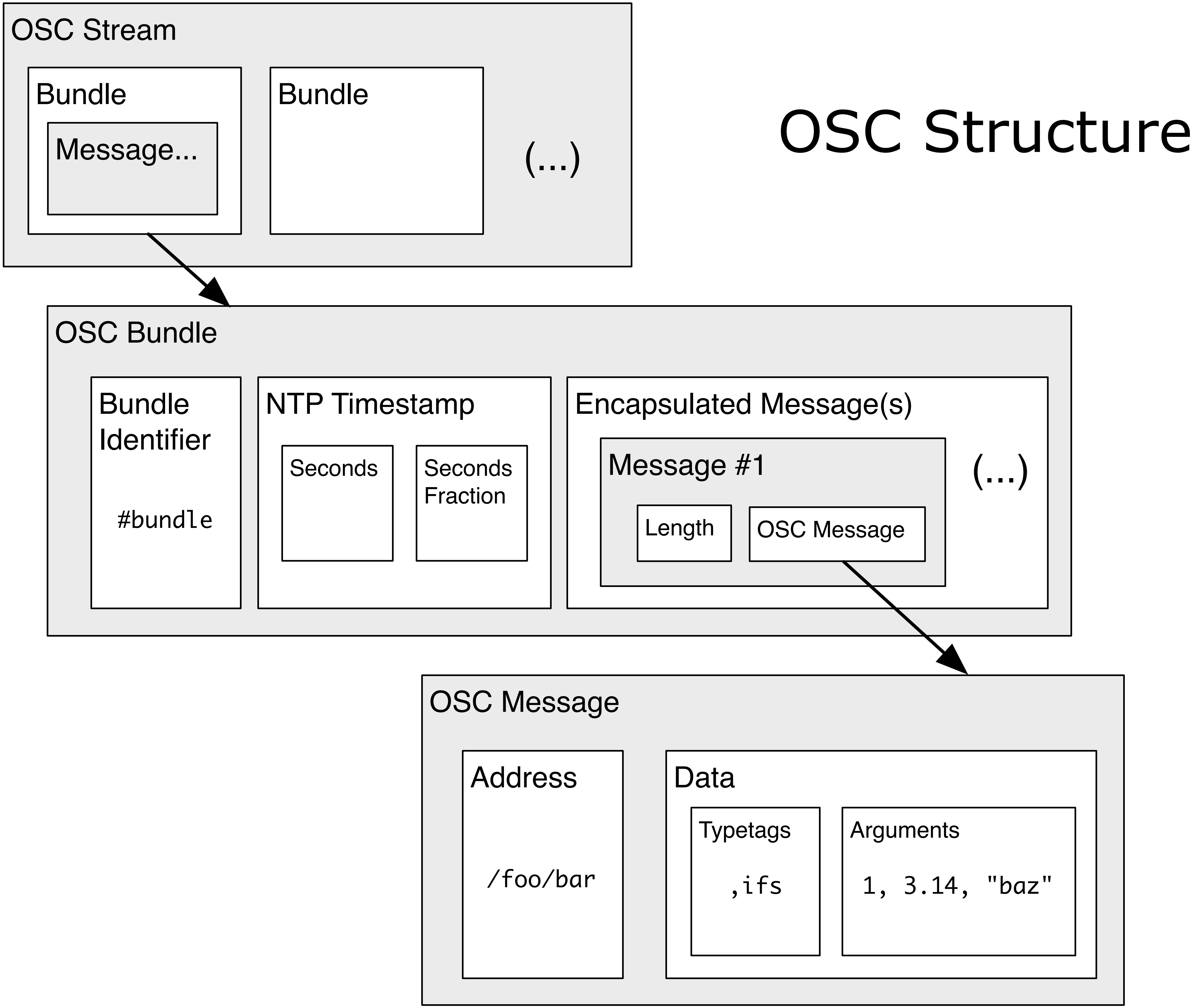
- Design: at the whim of its benevolent dictators
  - Acceptance criteria for a new idea:
    - Appropriate to the scope of OSC definition
    - Established need
- Can be used in closed-source products, provided the implementation has a compatible license.

OSC is a Content Format

- OSC is not a Standard:
    - no conformance certification.
- OSC is not a Protocol:
    - no convention for detection, negotiation, error handling
- OSC is a Content Format:
    - a content format is a structured container of *primitive* data types.

OSC Primitive Types

- strings (human readable)
- numbers: int32, IEEE 754 float single
  - optional types: int64, double, etc
- “blobs” (byte arrays)
- time
  - ‘t’ typetag as NTP time,
    - pair of uint32 {seconds, seconds fraction}


OSC Structure
OSC Stream
Bundle
Message...
Bundle
(...)
OSC Bundle
Bundle Identifier
#bundle
NTP Timestamp
Seconds
Seconds Fraction
Encapsulated Message(s)
Message #1
Length
OSC Message
(...)
OSC Message
Address
/foo/bar
Data
Typetags
,ifs
Arguments
1, 3.14, "baz"

OSC practice

OSI Layer	OSI Layer #	Topic
Application	7	Control Semantics, Choreography
Presentation	6	OSC Structure
Session	5	Discovery, Enumeration, Authentication
Transport	4	Quality of Service
Network	3	Bandwidth Reservation
Frame	2	Clock Synchronization
Hardware	1	Cabling, Wireless, Power

Definition for Audio Control Data and Examples

Section 2

Audio Control Data

- Any time-based information related to an audio stream other than the audio component
    - Non-time-based audio-related information are static stream properties
        - You can use OSC for this but its not “intended”

Properties of Audio Control

- Temporal errors can produce audible side-effects,
  - “zipper” noise
  - spatialization aliasing errors
  - low quality interactivity (boring instruments)
- Variable sample rates, mixed rates
- Audio systems may have sensitive or high-power hardware components needing robust control

Examples

- Instrumental gesture data
- Spatial auditory scene parameters
- Spatial rendering engine control
- Audio synthesis engine control

Instrumental Gestures

- Constrained by limits of mechanical and neural human body dynamics

- With training, gestures can be repeatable with very high precision in time and space

    - Delay Tolerance in performance: 20msec round trip delay (Chafe et al)

    - Temporal Repeatability: ~10hz continuous motion, 10msec accuracy, 1msec precision, 1000hz SR (Wessel)

Information Rate

$$I = \log_2\left(1 + \frac{\rho}{\sigma}\right) \text{bits}$$

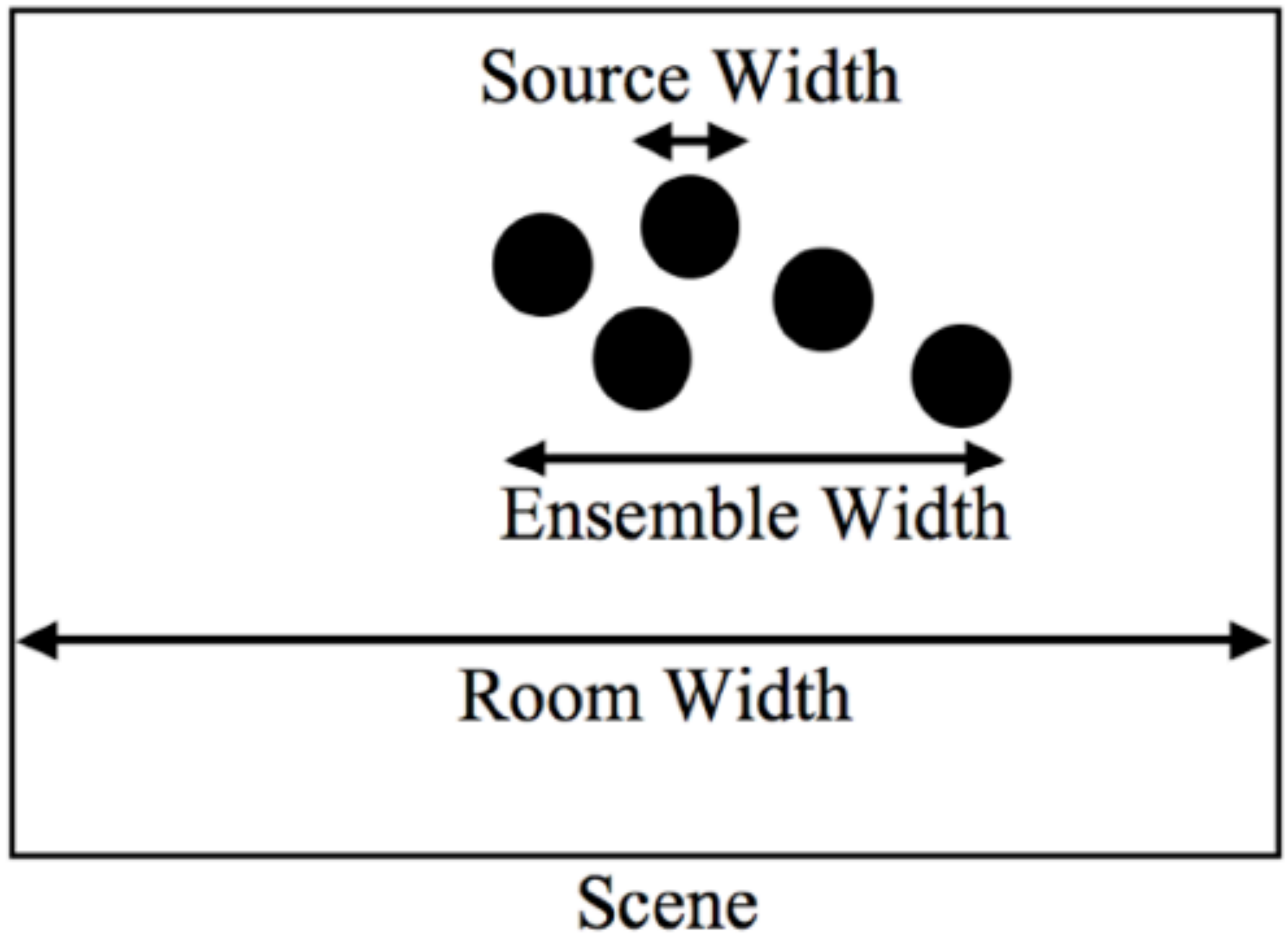
- Essentially, number of bits that change per second.

- Is the fundamental determinant in Fitt's Law

  - ISO-1941-9 (measurement of information transfer rate in target selection, mouse = ~3 bits/sec)

- Instrumental gestures are ~100bits/sec in time dimension alone. IR in space/force to be determined.

Spatial Scene Parameters



- Auditory Spatial Schemata (Gary Kendall)
  - Source location, width, directivity
  - Diffusion from enclosing geometry (rooms)

- Sub-audible frequency band (0-40hz)

Spatial Rendering Engine

- Examples including driving a distributed array with Ambisonic/Wave-Field-Synthesis filter coefficients:
    - Temporal error is equivalent to transducer positioning error
    - Temporal sync within 5% of sample frame at max controlled frequency
        - 500 microseconds (usec) at 96,000hz
        - AES2003-11 Best Practices for Network Audio

Audio Synthesis Engine

- Data-driven analysis and synthesis algorithms
  - Granular, concatenative, additive synthesis, large filter banks
- Can have very high bandwidth: thousands of entities per second
- Sub-sample accuracy (500 usec is good enough)
  - float32 is good enough for 500usec accuracy (but only just barely).

Temporal Quality Assurance

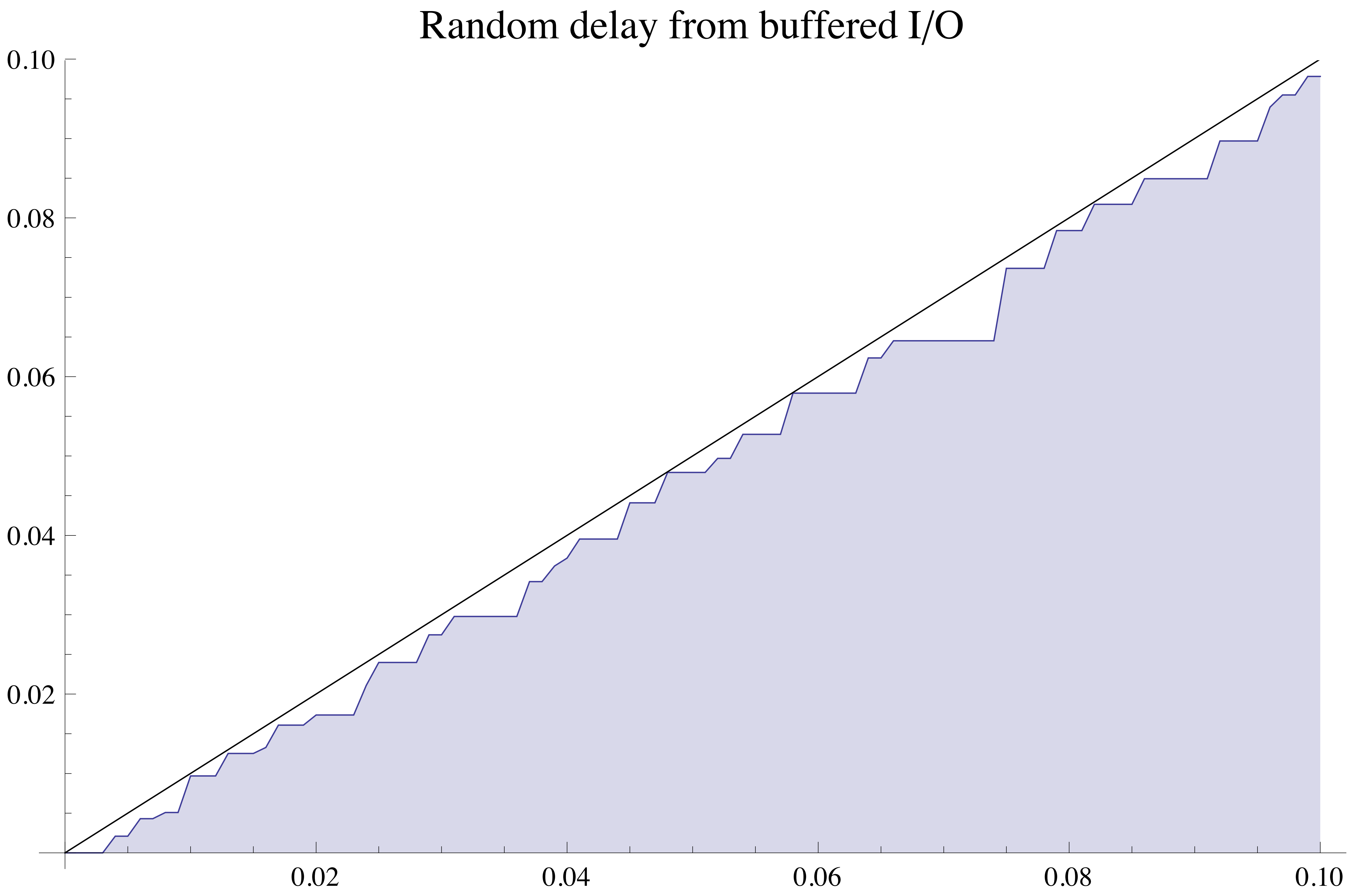
Section 3

Temporal Quality Assurance

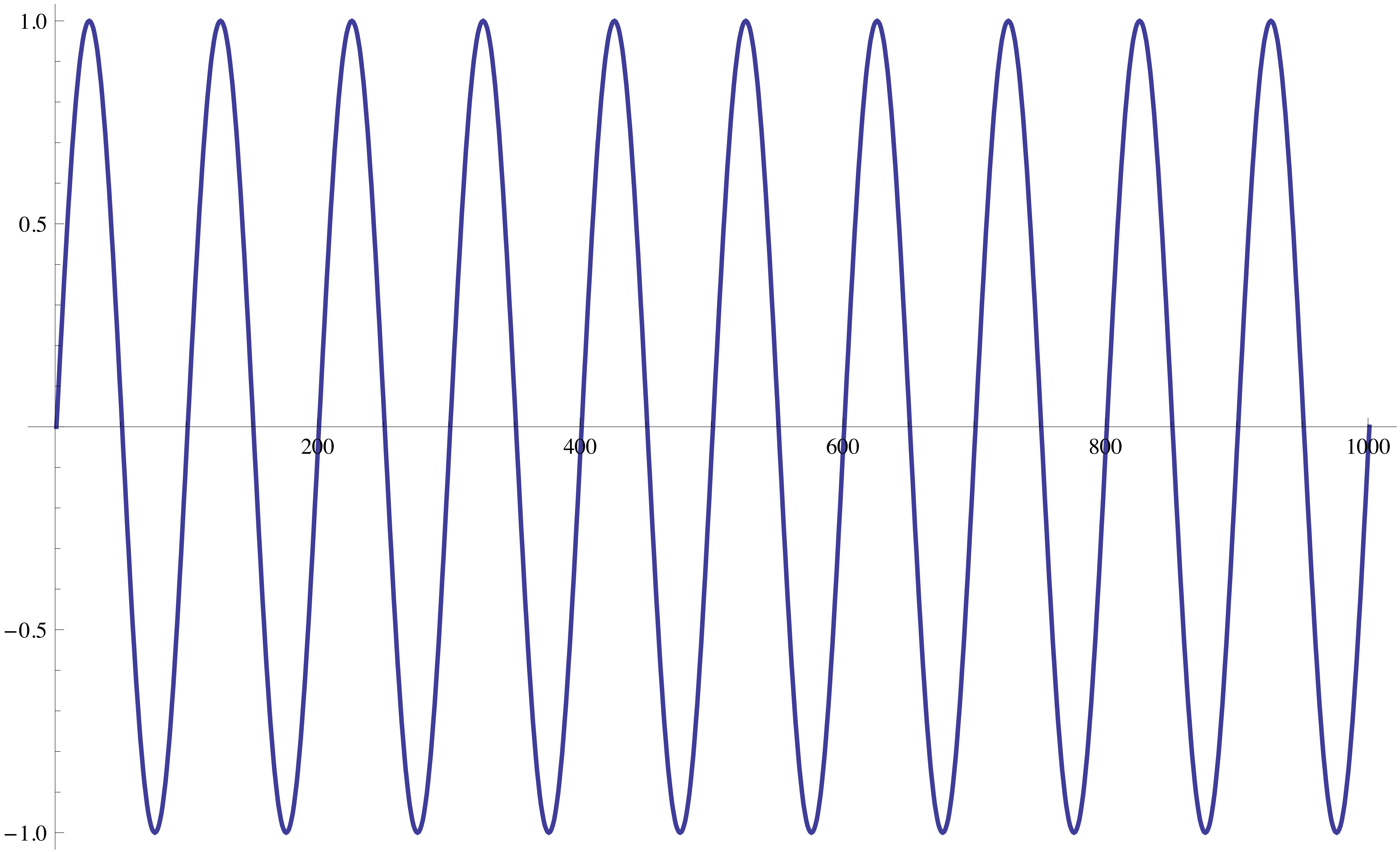
- Bounds on various delay properties:
  - maximum, minimum
  - variance.
  - accuracy and precision of scheduling

Interrupt Service Jitter

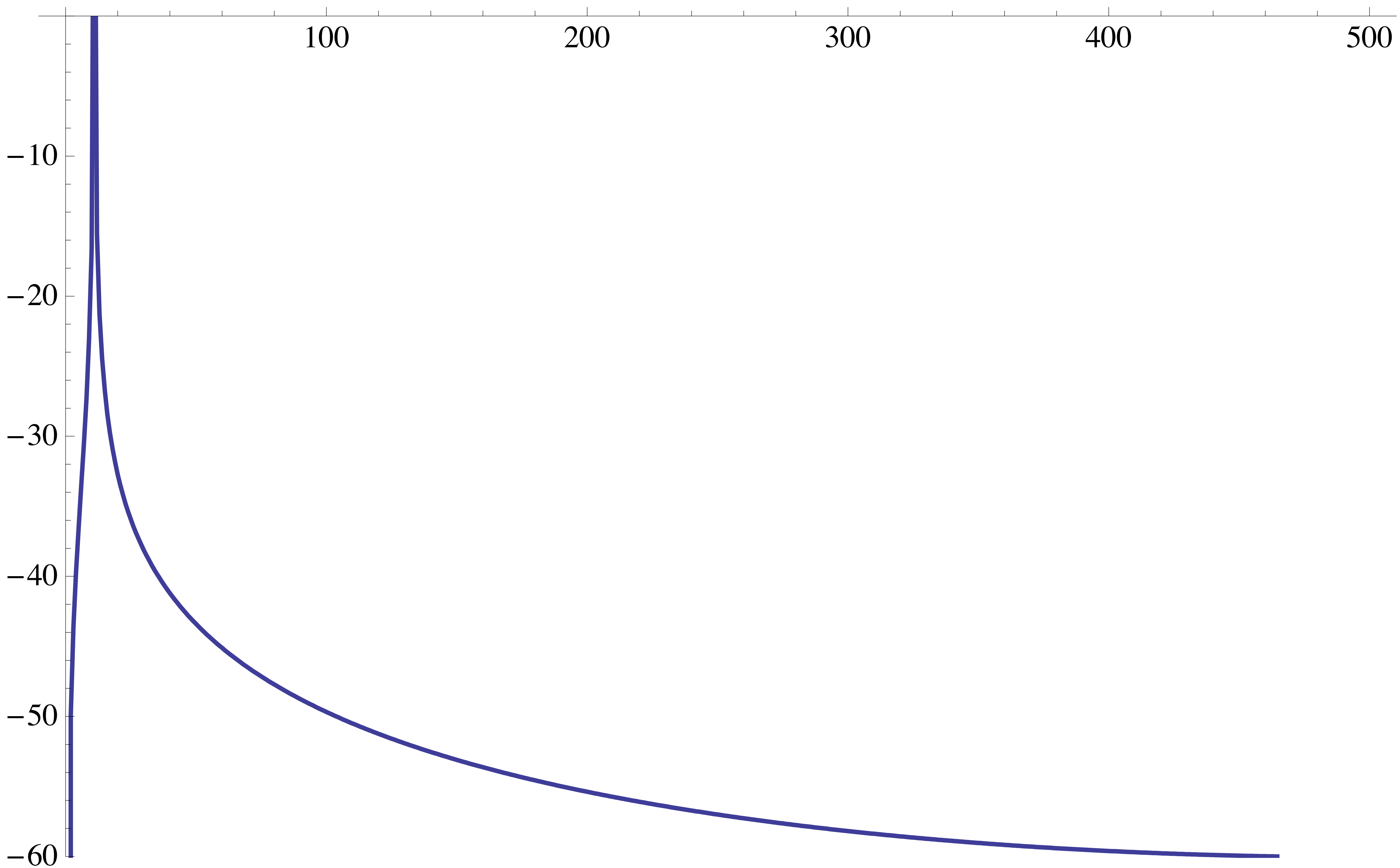
- Hardware or software gets some data and raises and interrupt service request.
- Data goes into a buffer until the interrupt is serviced by the system scheduler, then it gets delivered
- Interrupt servicing has delay distribution of a random wait-time queue



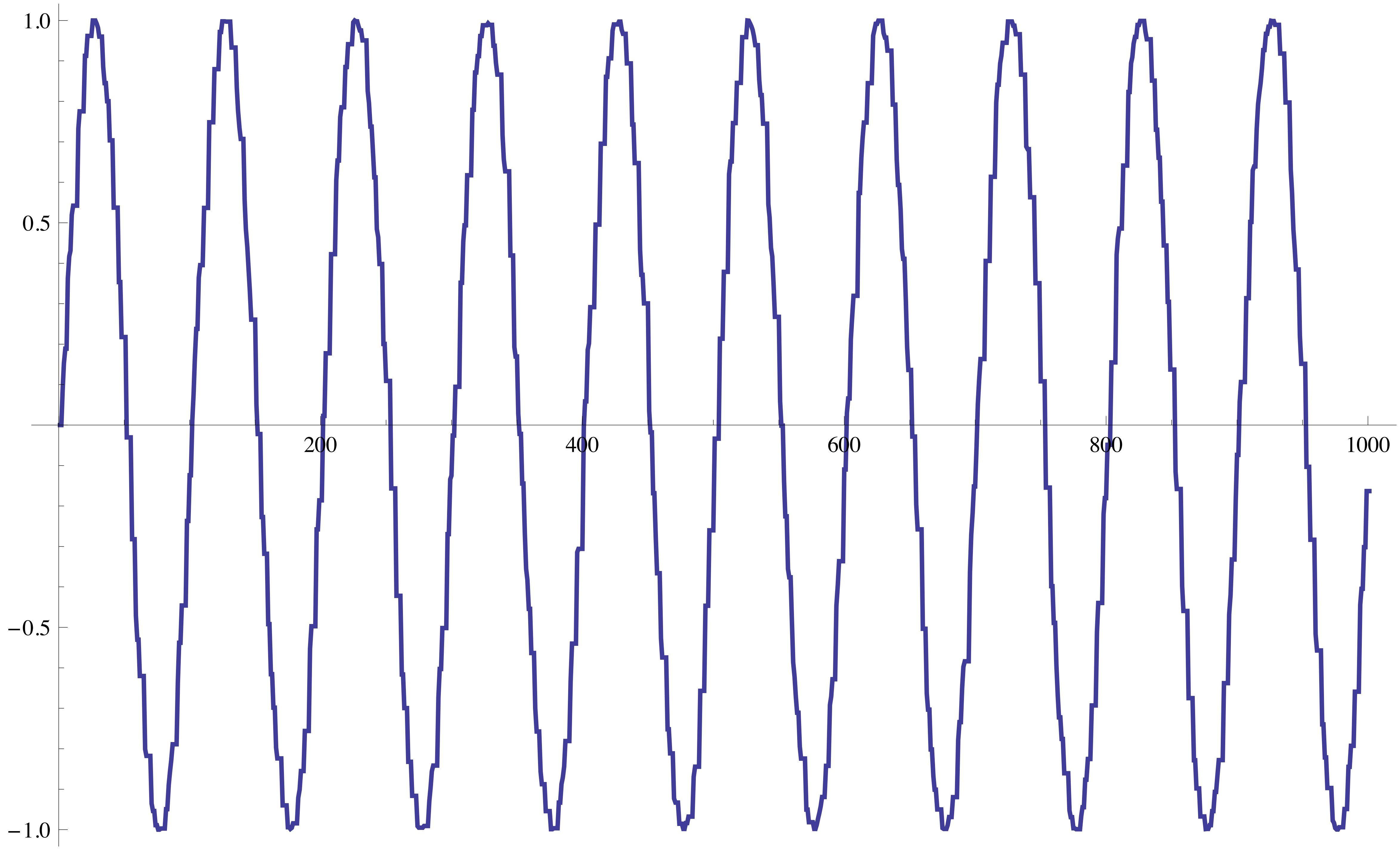
Typical ISR variable delay 3-10msec



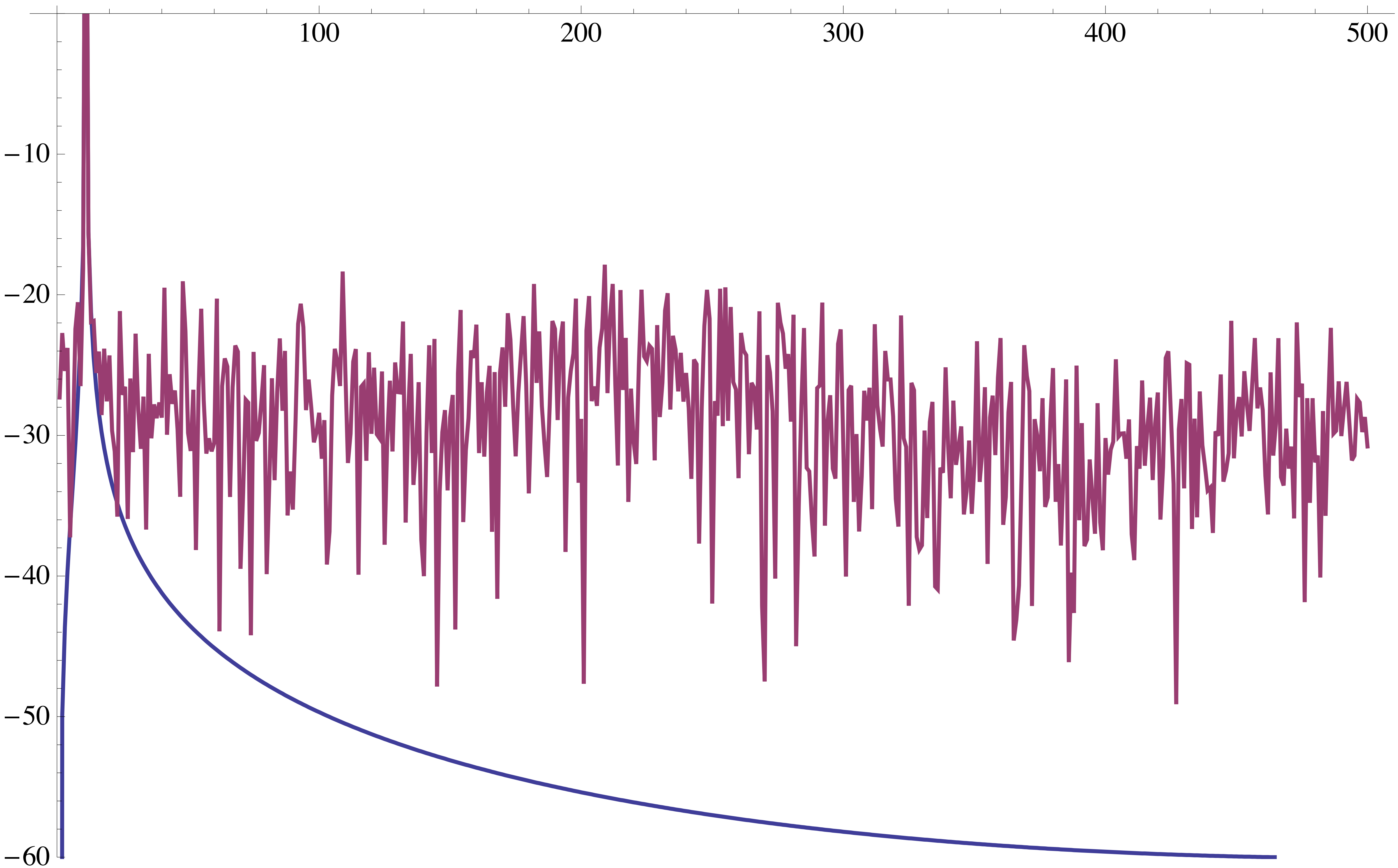
10hz Signal



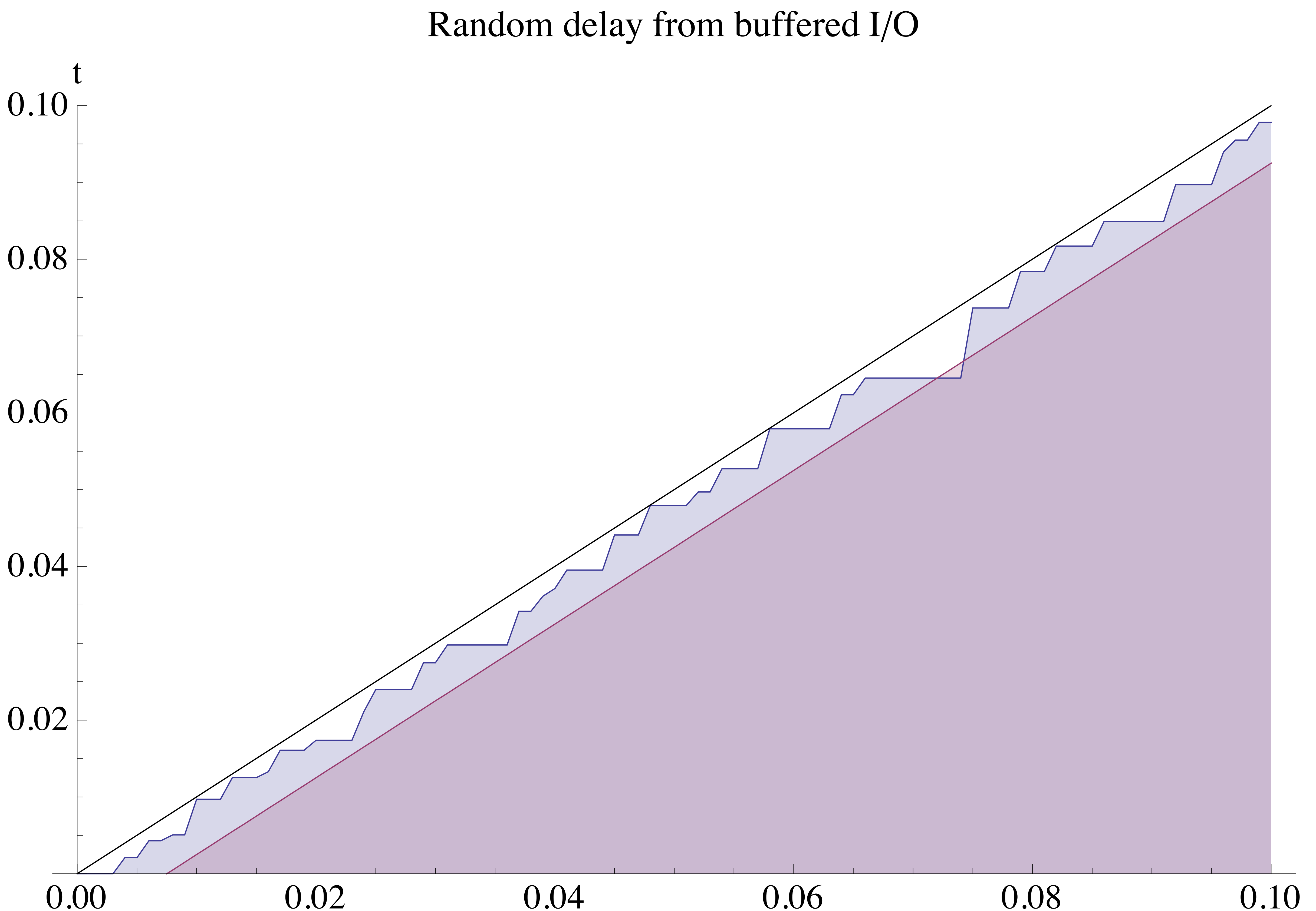
Spectrum of 10hz Signal



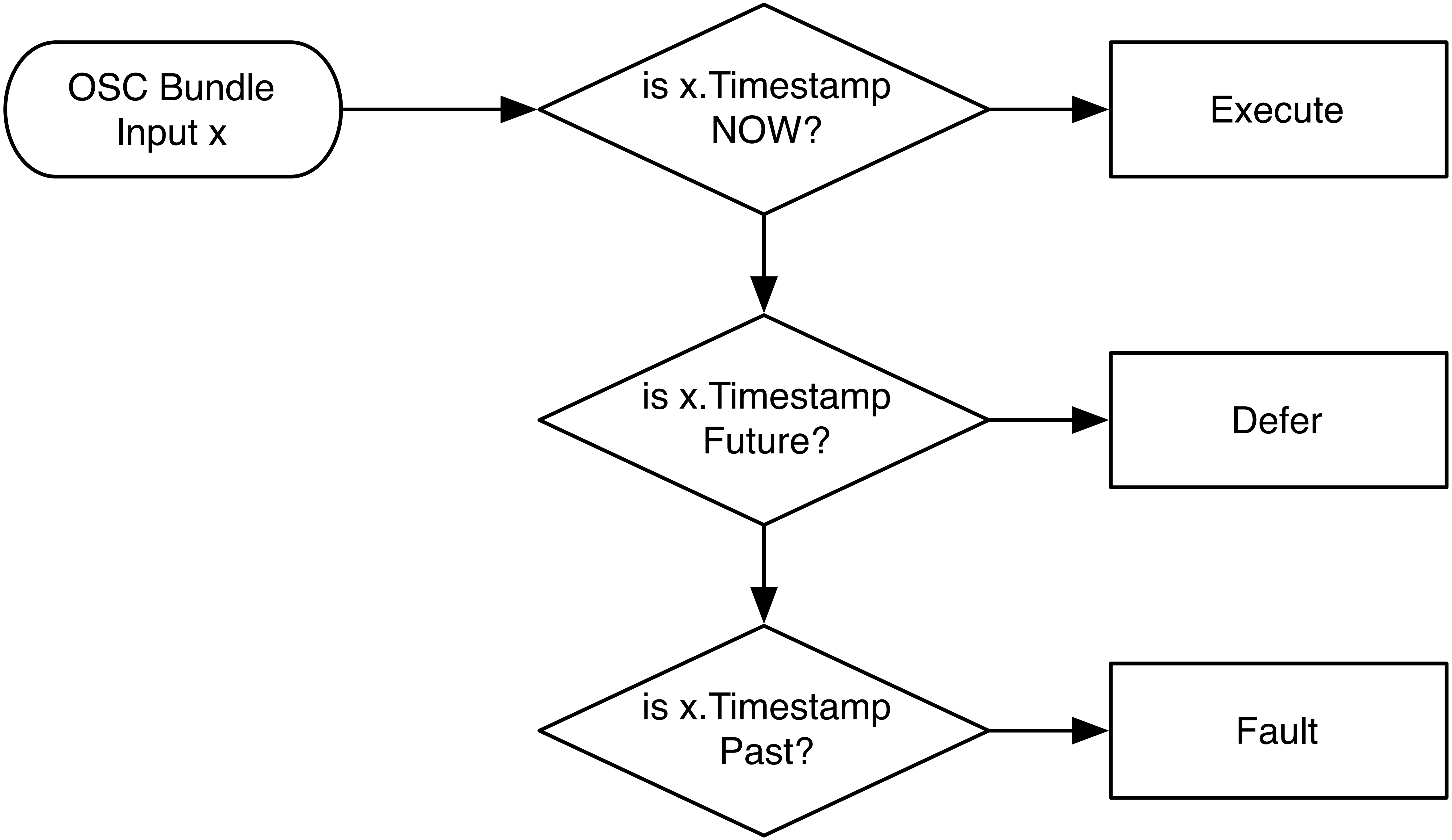
10hz Signal under 1-3msec variable delay



Jitter induced noise on 10hz carrier
(24db => ~4 bits resolution)



Jitter filter by rescheduling



Forward Synchronization Scheduler
(implementation is a priority queue)

	0.01 msec	0.1 msec	1. msec	2. msec	4. msec
0.5 Hz	100.806	80.942	60.5853	54.4588	48.2834
1. Hz	89.4672	69.2973	49.5129	**42.7719**	**37.1899**
2 Hz	83.5256	64.1865	**44.4936**	**37.811**	**32.166**
4 Hz	77.8606	58.3905	**38.2024**	**32.4498**	**25.4497**
8 Hz	72.3401	52.0053	**31.2989**	**25.7653**	**20.1786**
16 Hz	66.1133	**45.8497**	**25.8291**	**19.7408**	**14.3312**
32 Hz	60.2471	**39.6844**	**19.7202**	**13.546**	**8.26448**
64 Hz	53.9285	**33.8882**	**13.9203**	**7.90135**	**1.7457**

Carrier frequency vs jitter magnitude,
BOLD => less than 8-bits headroom

Summary of Jitter

- For typical control frequencies in the sub-audible bandwidth 0-40hz, typical transport jitter of a few milliseconds is unacceptable
  - Best effort is not good enough.
- Forward sync scheduling can remove some jitter problems (maybe to 0.1 msec)
- For audio apps a better solution is to synchronize physical-time with sample-time
  - Using a DLL filter, interpolation strategies etc.

Transport Considerations

Section 4

Transport Topics

- Types of transports and their properties, e.g.:
  - UDP
  - TCP
  - Serial (USB, RS232, file pointers)

Ethernet AVB

- A solution for the endpoint-discovery and connection management problem (Bonjour/mDNS + AVBC)
- Uses bandwidth reservation protocols to ensure network availability with bounded delay (2msec, Class A)
- Solves clock synchronization problem at the ethernet frame layer, (500 usec per AES2003-11)
- OSC can be sent over AVB streams using a MIME type identifier (1722.1 working group)

Describing Audio Control Data

Section 5

Describing control data

- Four design patterns for describing a software interface:
    - RPC: Remote Procedure Call
    - REST: Representational State Transfer
    - OOP: Object Oriented Programming
    - RDF: Resource Description Framework

By Example...

Channel #3

RPC

Remote Procedure Call

/setgain (channel number = 3) (gain value = x)

Functional, reference-oriented semantics
Good for highly-dynamic data structures
(granular synth)

REST

REpresentational State Transfer

/channel/3/gain (x)

Emphasis on enumeration of resources
Encourages stateless protocols

OOP

Object Oriented Programming

/channel/3@gain (x)

/channel/3/setgain (x)

“@” => attribute (after XPath)

RDF

Resource Description Framework

/channel,num,3/op,is,set/lvalue,is,gain (x)

Unordered set of semantic triples of {subject,predicate,object}

Programming Control Data

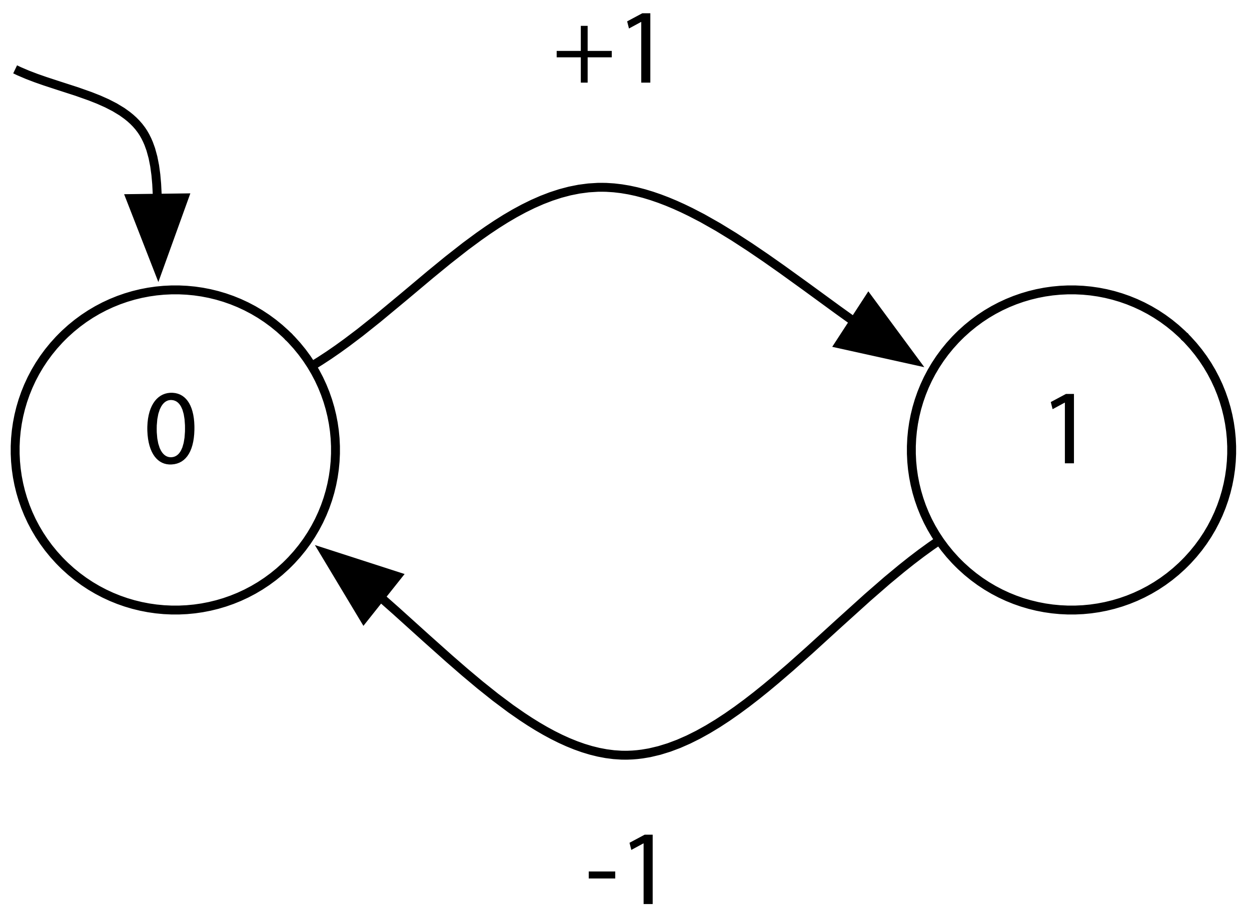
Section 5.3 - 5.4

Stateless Interfaces

- A stateful protocol is one where the meaning of a message has some dependence on a previously transmitted message.

Example

Button State Machine



Stateful Encoding

/button +1

/button -1

/button +1

/button -1

Error Robustness

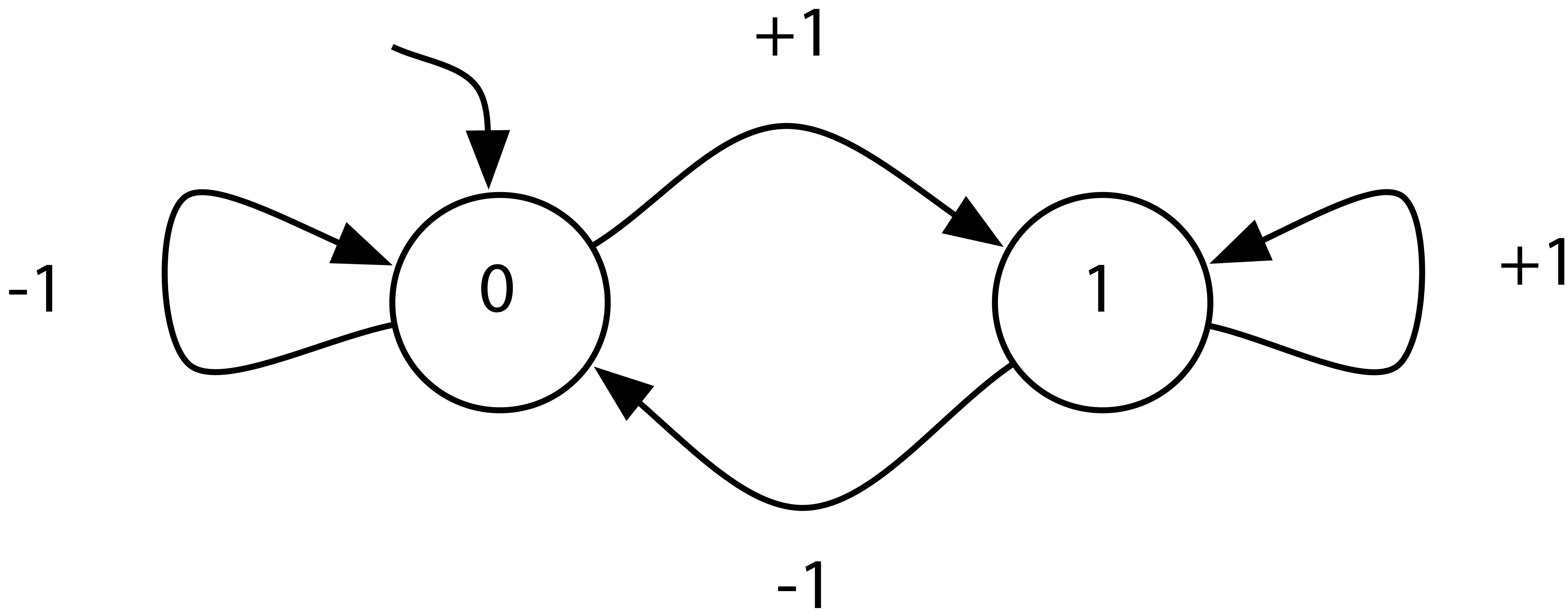
/button +1

/button +1

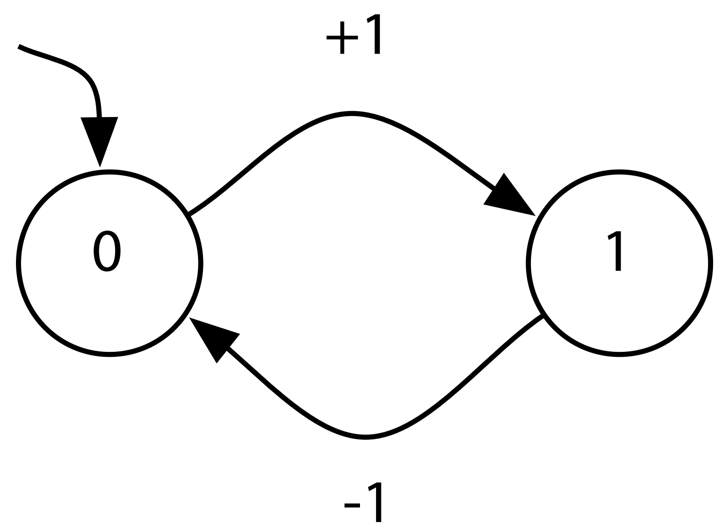
...?

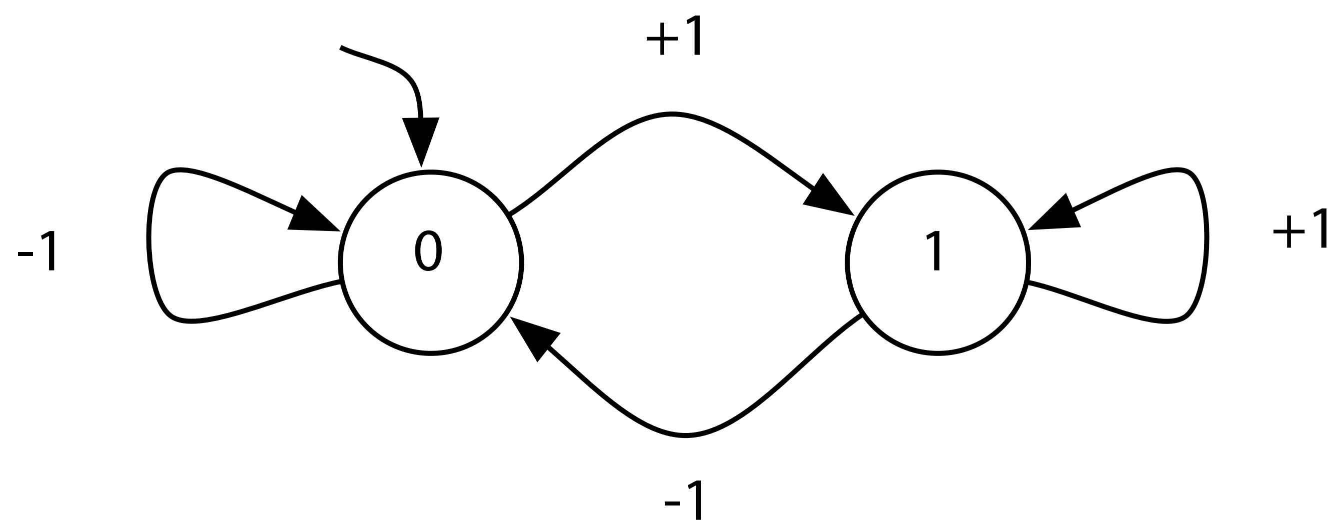
/button -1

Robust State Recovery



Parser Complexity is 2x!





Stateless Encoding (REST)

/button 0

/button 1

...?

/button 0

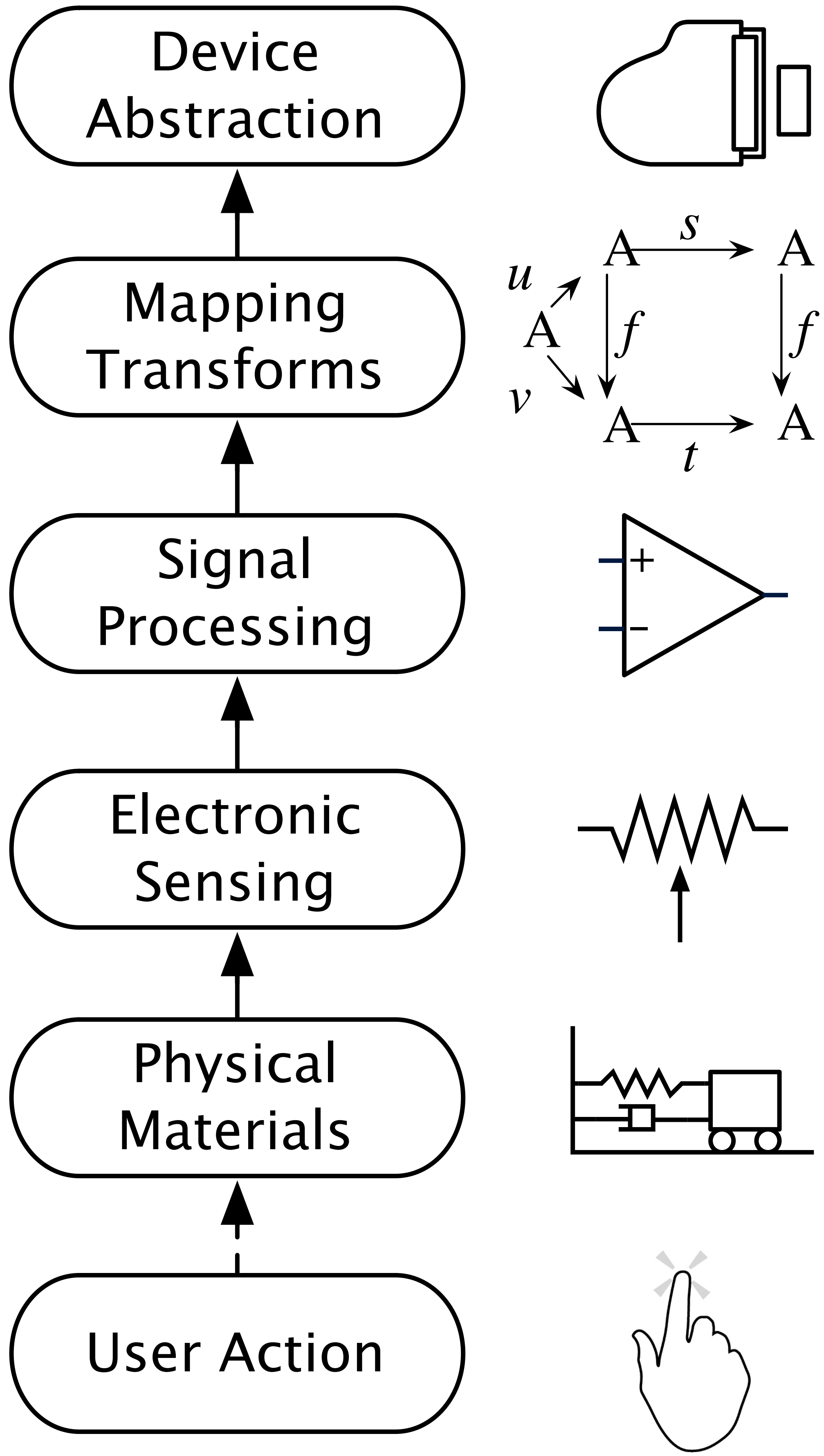
Stateless Summary

- Stateful protocols are an optimization that reduces protocol bandwidth at the expense of protocol implementation complexity,
  - Especially when error recovery is involved
    - Otherwise, use TCP to ensure no errors (pushes complexity down to transport layer)
- Stateless interfaces can more readily support temporal constraints such as leases and expiration timestamps.

Abstraction Layering

- Effective strategy for management of complexity by encapsulation

- Can have some non-trivial complications...


Device Abstraction
Mapping Transforms
Signal Processing
Electronic Sensing
Physical Materials
User Action

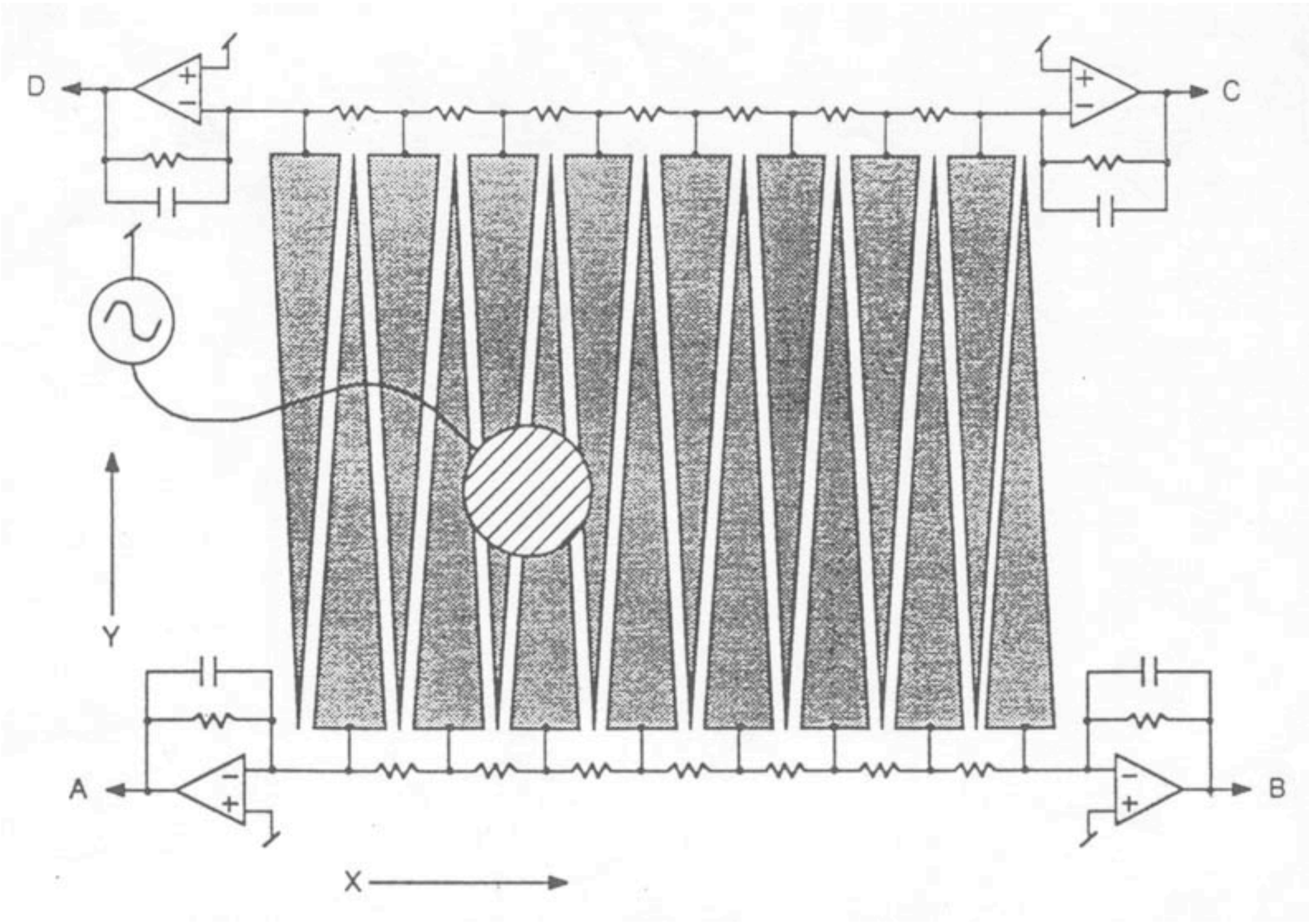
Multi-Layer Operations

- Some transformation operations transcend the layer structure
  - Especially *mapping* transformations!

Example: Radio Drum

Schloss/Matthews/Boie

Radio Drum Sensor



Radio Drum Mapping

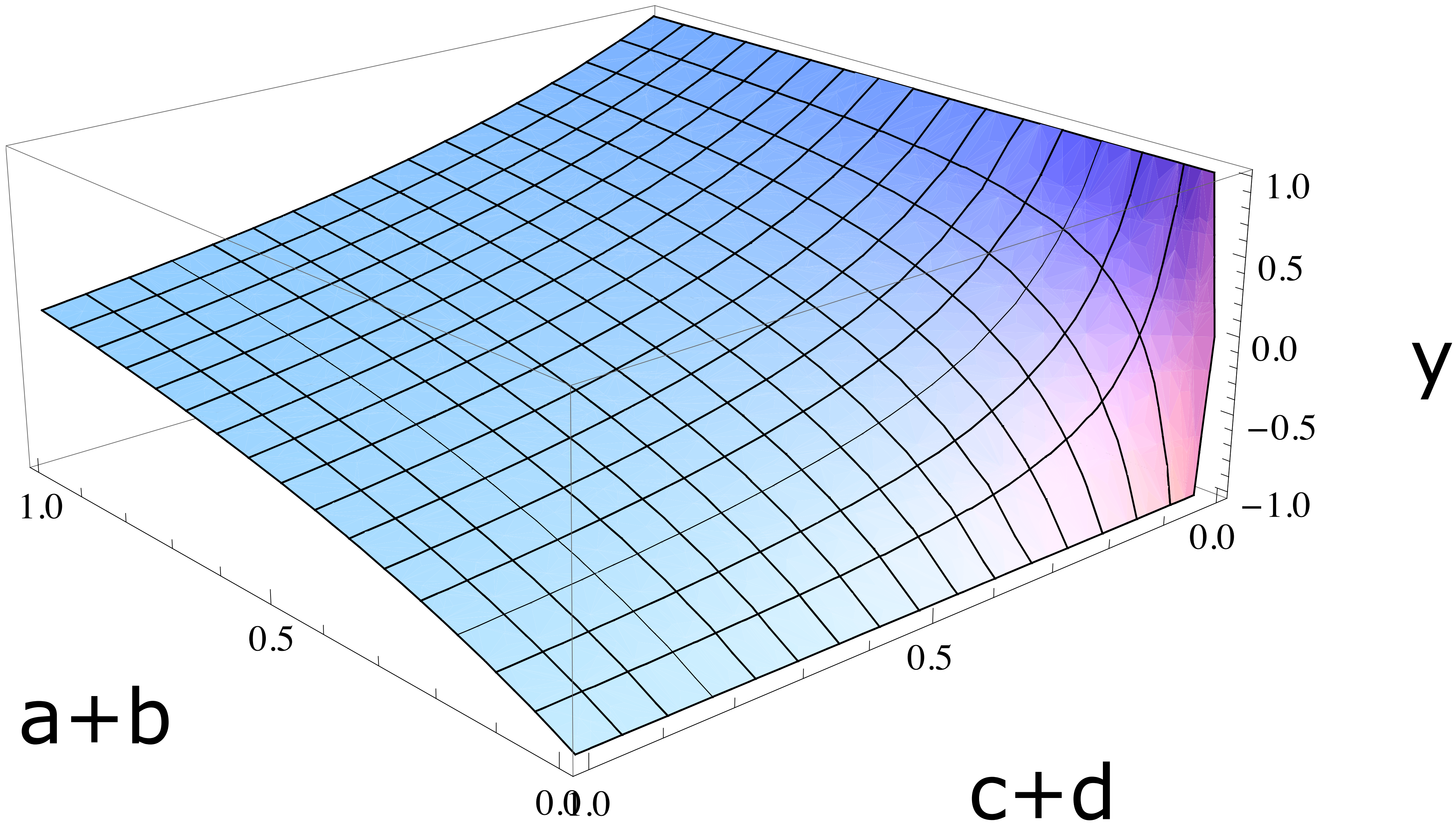
$$y = \frac{a+b-c-d}{a+b+c+d}$$

- Raw sensor to position

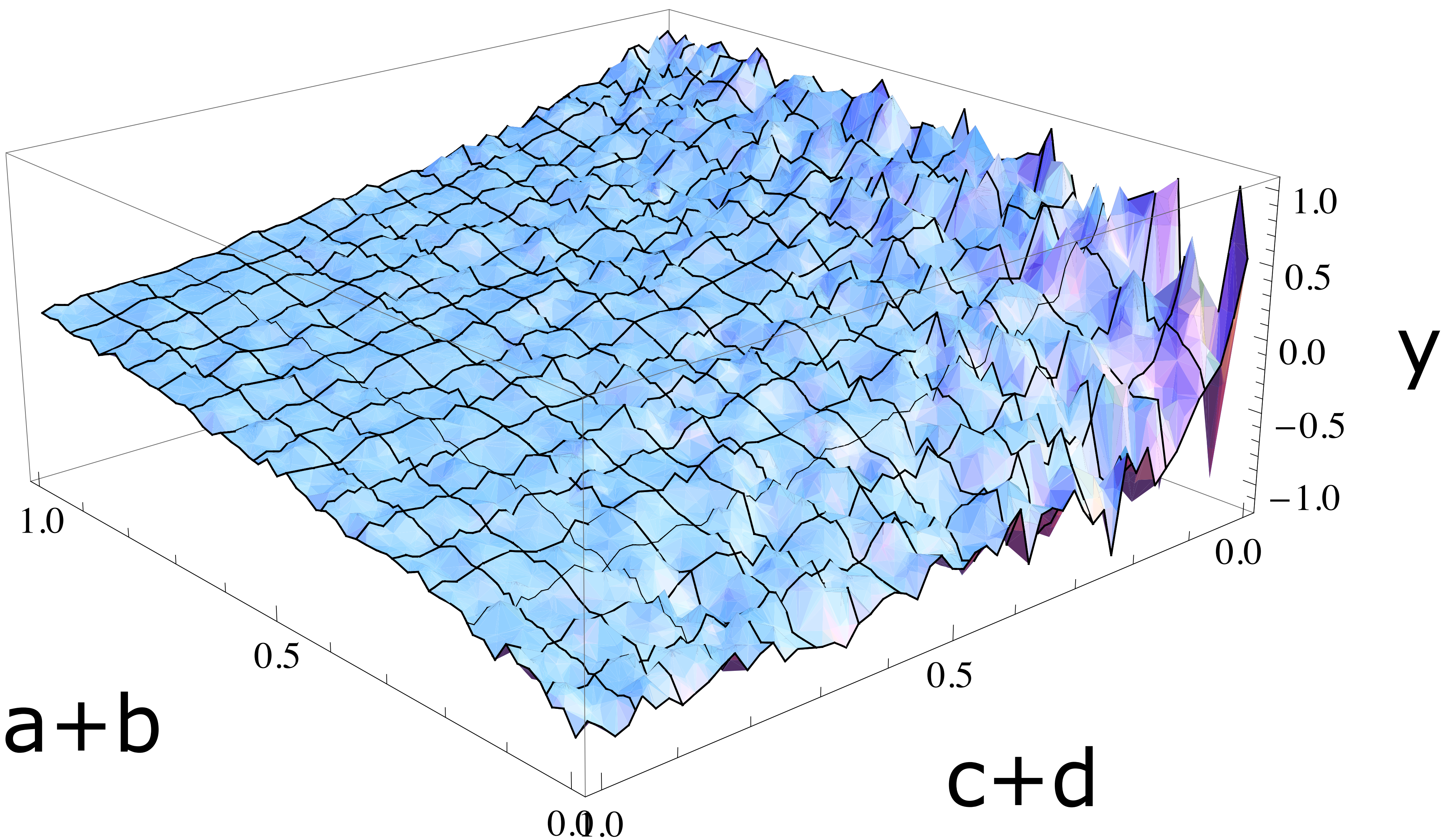
$$\|\mathrm{dy}\| = \sqrt{\left|-\frac{a+b-c-d}{(a+b+c+d)^2} - \frac{1}{a+b+c+d}\right|^2 + \left|\frac{1}{a+b+c+d} - \frac{a+b-c-d}{(a+b+c+d)^2}\right|^2}$$

- Norm of derivative wrt. a + b, c + d

Ideal Mapping



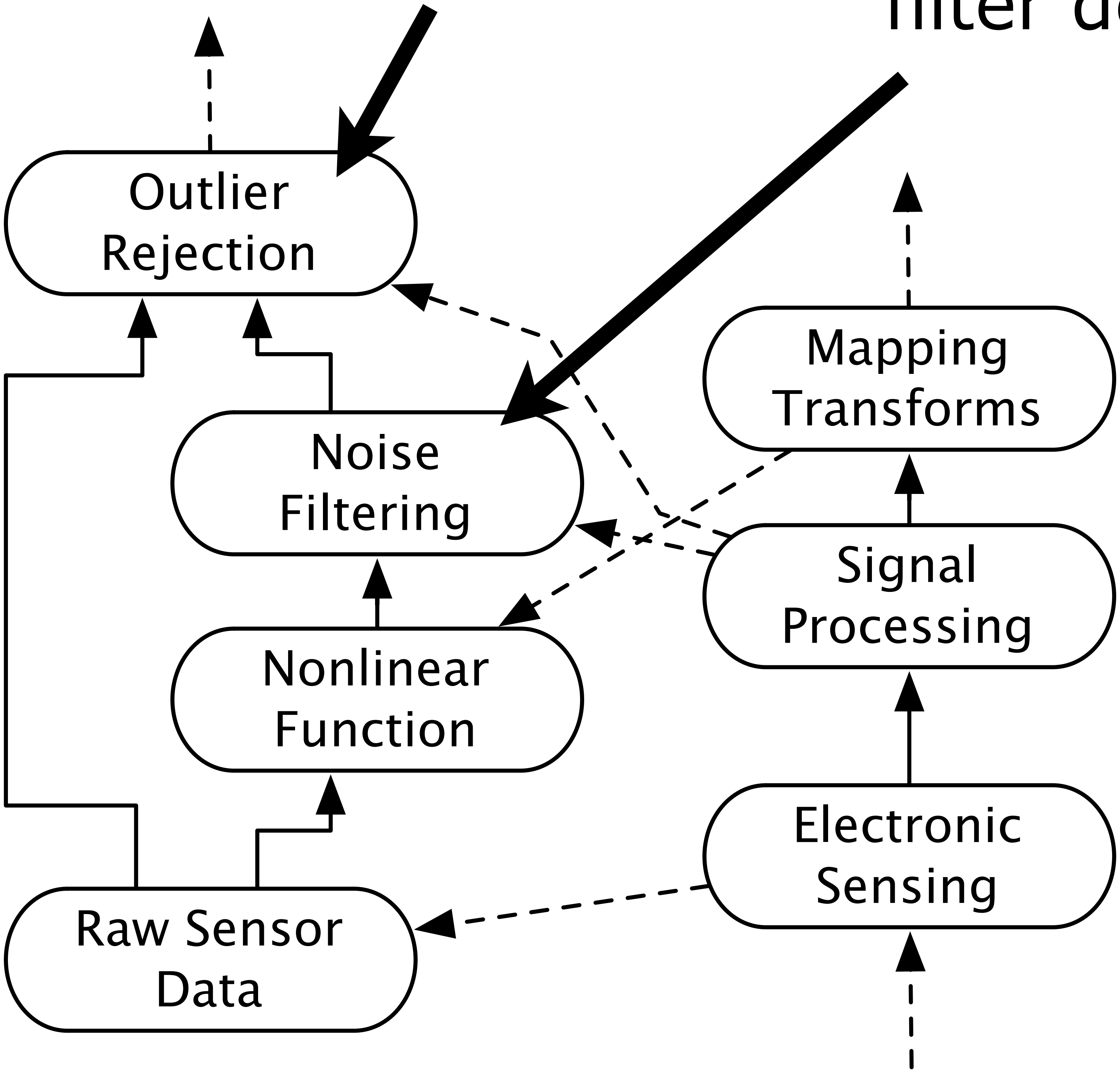
Noise Amplification



- ||dy|| -> infinity as (a+b,c+d) -> (0,0)

Needs raw data here

Noise reduction filter depends on *y*



Layering Summary

- Applications should maintain representations of control streams at multiple layers simultaneously when possible
    - This will support operations that need data from multiple layers.

Summary

- Clock synchronization tolerance depends on temporal control information and audio application needs (do the calculations, don’t just ignore it).

- Ethernet AVB meets all the synchronization needs for audio control data, as well as bandwidth reservation. (500usec error, 2msec delay)

- There are multiple effective strategies for describing audio control interfaces (RCP, REST, OOP, RDF)

- Statefree protocols and multilayered representations can improve program reliability, enable temporal features, and increase flexibility.

The End

Your comments / feedback:

andy@cnmat.berkeley.edu

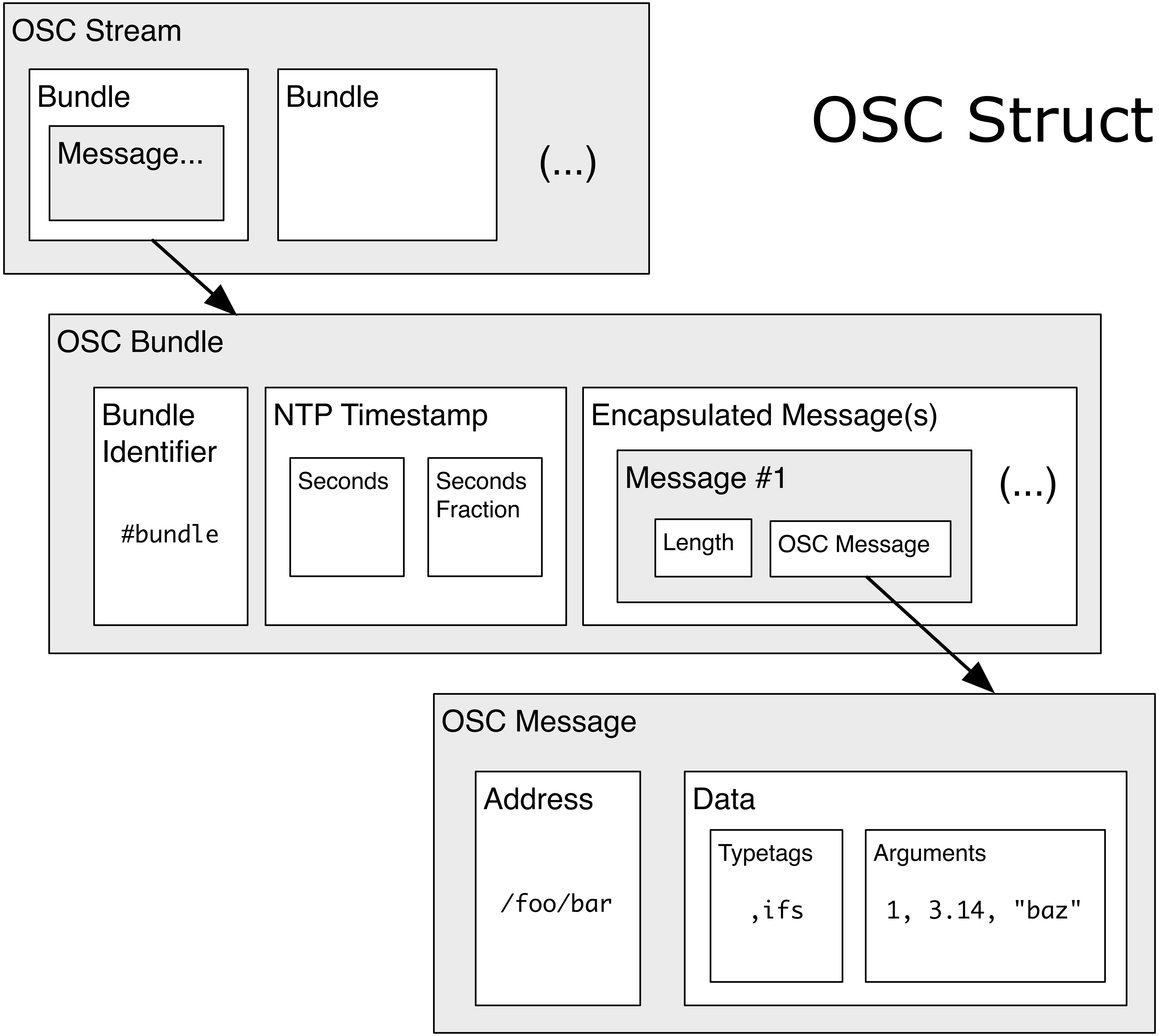
Appendix

(omitted slides)

Synchronization

- Suppose there is a set of changes to be applied all at once or not at all.
- Suppose there is a set of changes that should be committed by time T, after which the request is considered to be expired.
- In OSC we use Bundles to express frames of temporally-synchronized data.
- The quality of distributed synchronization is limited by the clock distribution error, which is a network layer or frame layer service.

OSC Structure



Framing

- OSC needs a transport that includes a framing structure (such as datagram messaging, UDP)

- Any serial transport can be adapted to support framing with a frame encoding:

    - SLIP RFC1055: a byte-quoted encoding that is robust to interruption (with double-ended variant)

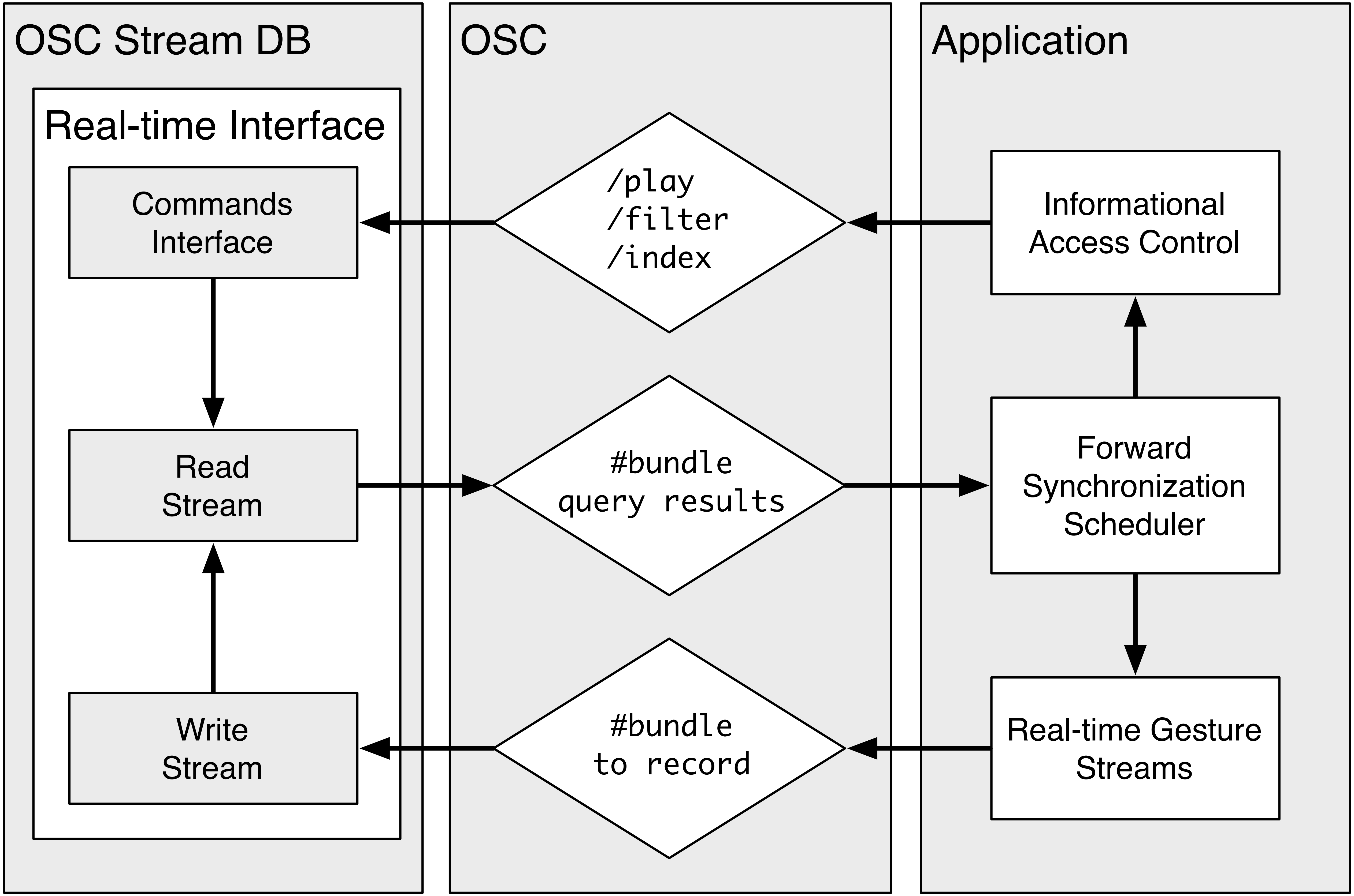
    - int32 length preamble: requires an assured serial transport (TCP) (see OSC 1.0)

Routing

- Implementations should expose as much detail as possible from the network routing layer, so that applications can make full use of routing capabilities of the transport.
  - e.g., bidirectional UDP
  - Reverse NAT traversal (OSCgroups)

Bulk Transports

- File pointers and databases can be treated as classes of serial transport having a bulk-IO delay distribution model (OSC Stream DB).
    - On serial transports we use a SLIP RFC1055 encoding to provide a datagram framing around OSC



OSC Stream DB