Real Logic

ACCELERATING SOFTWARE

Aeron

High-Performance Open Source Message Transport

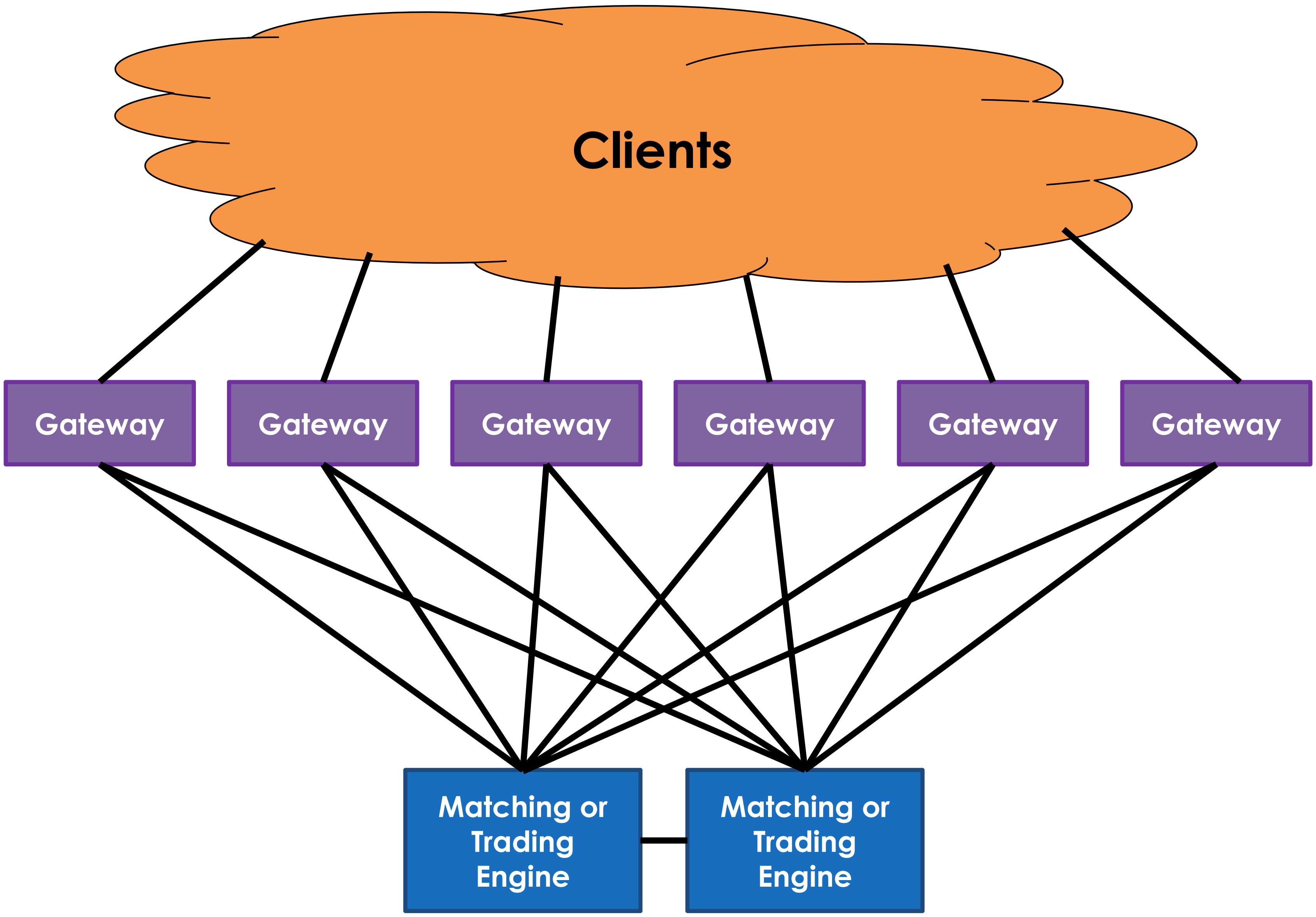
Martin Thompson - @mjpt777

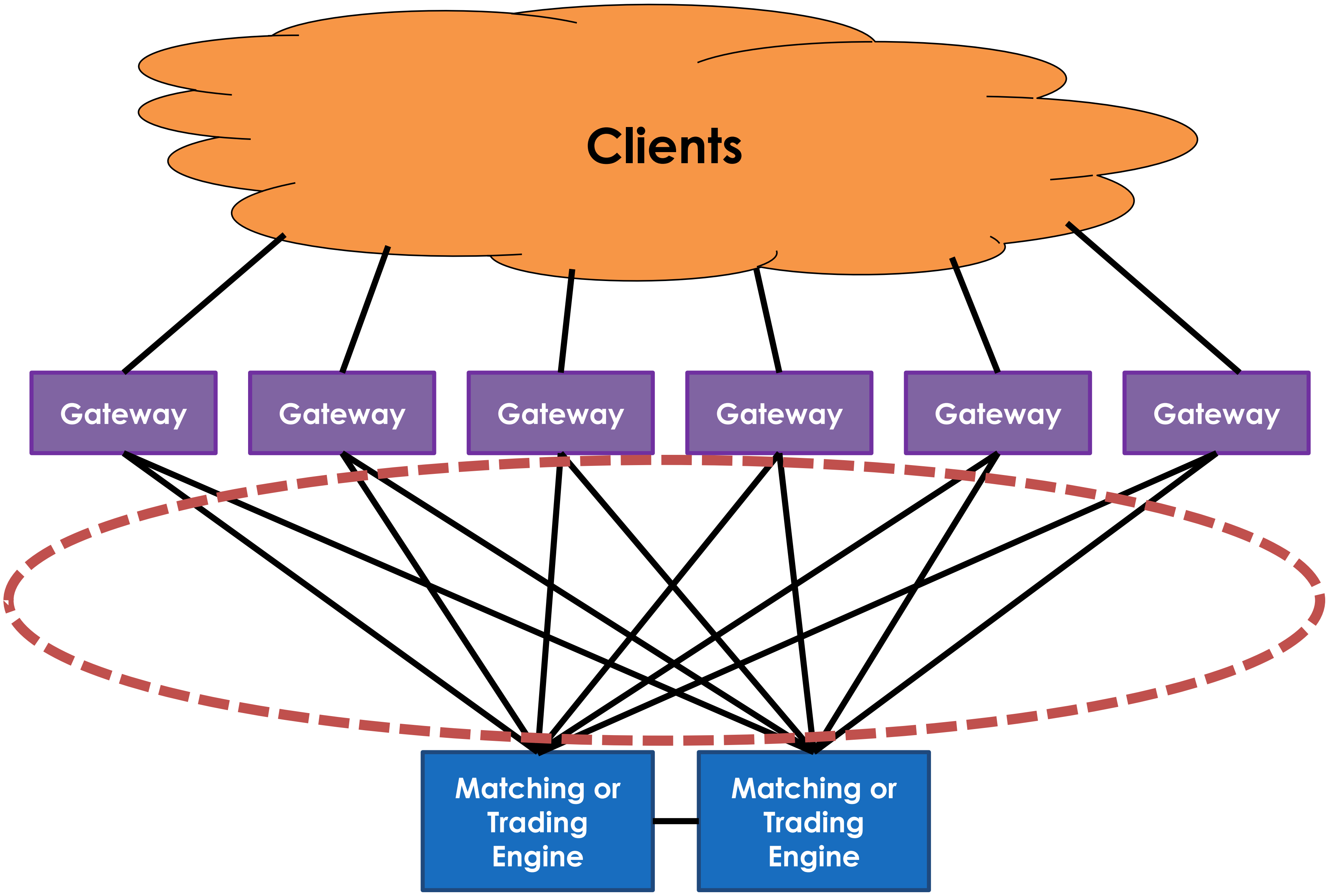
1. Why build another Product?
2. What Features are really needed?
3. How does one Design for this?
4. What did we Learn on the way?
5. What's the Roadmap?

1. Why build another product?

Not Invented Here!

There's a story here...

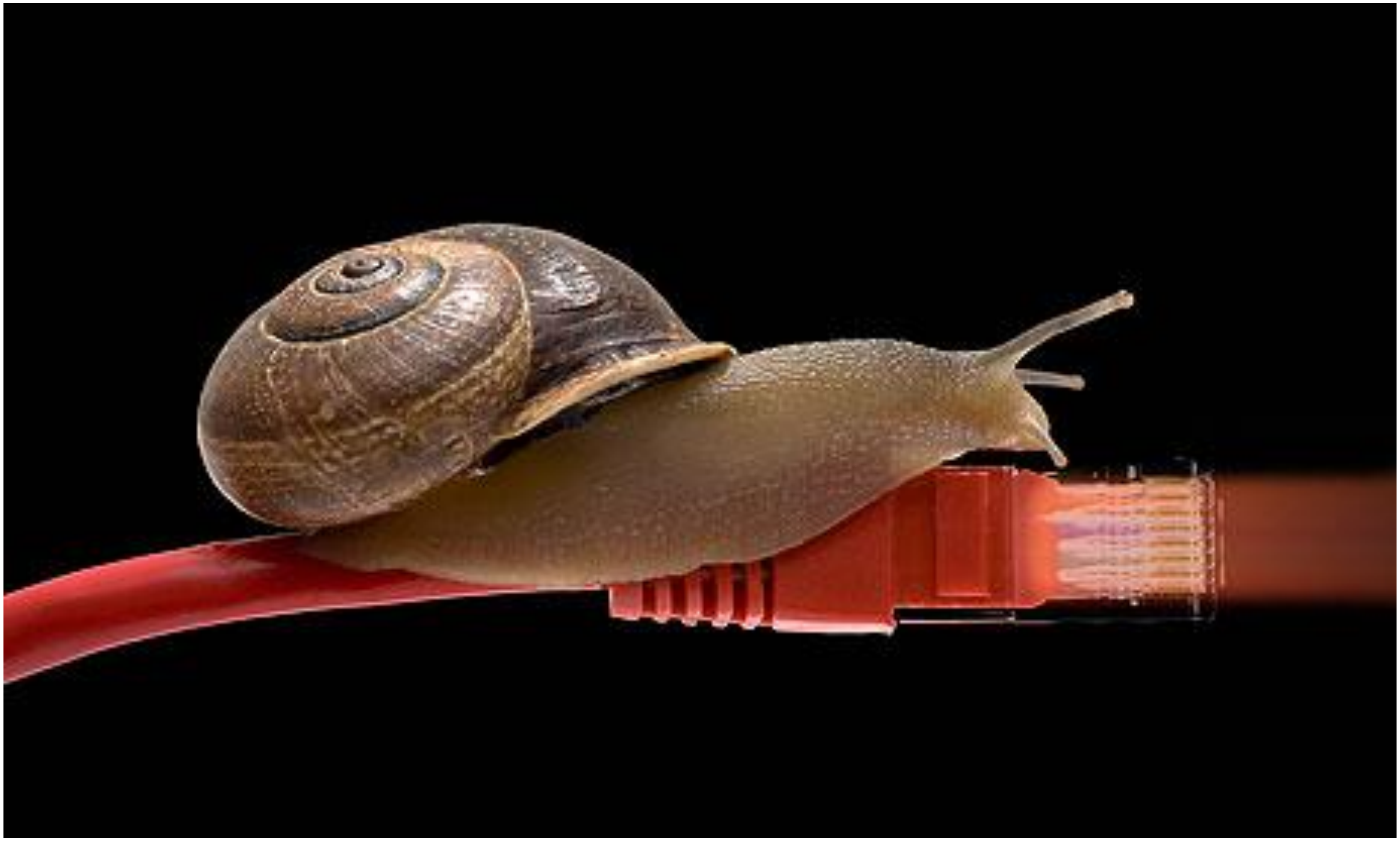

Clients
Gateway
Gateway
Gateway
Gateway
Gateway
Gateway
Matching or Trading Engine
Matching or Trading Engine


Clients
Gateway
Gateway
Gateway
Gateway
Gateway
Gateway
Matching or Trading Engine
Matching or Trading Engine

But many others could benefit

Feature Bloat & Complexity

Not Fast Enough



Low & Predictable Latency is key

We are in a new world

Multi-core, Multi-socket, Cloud…

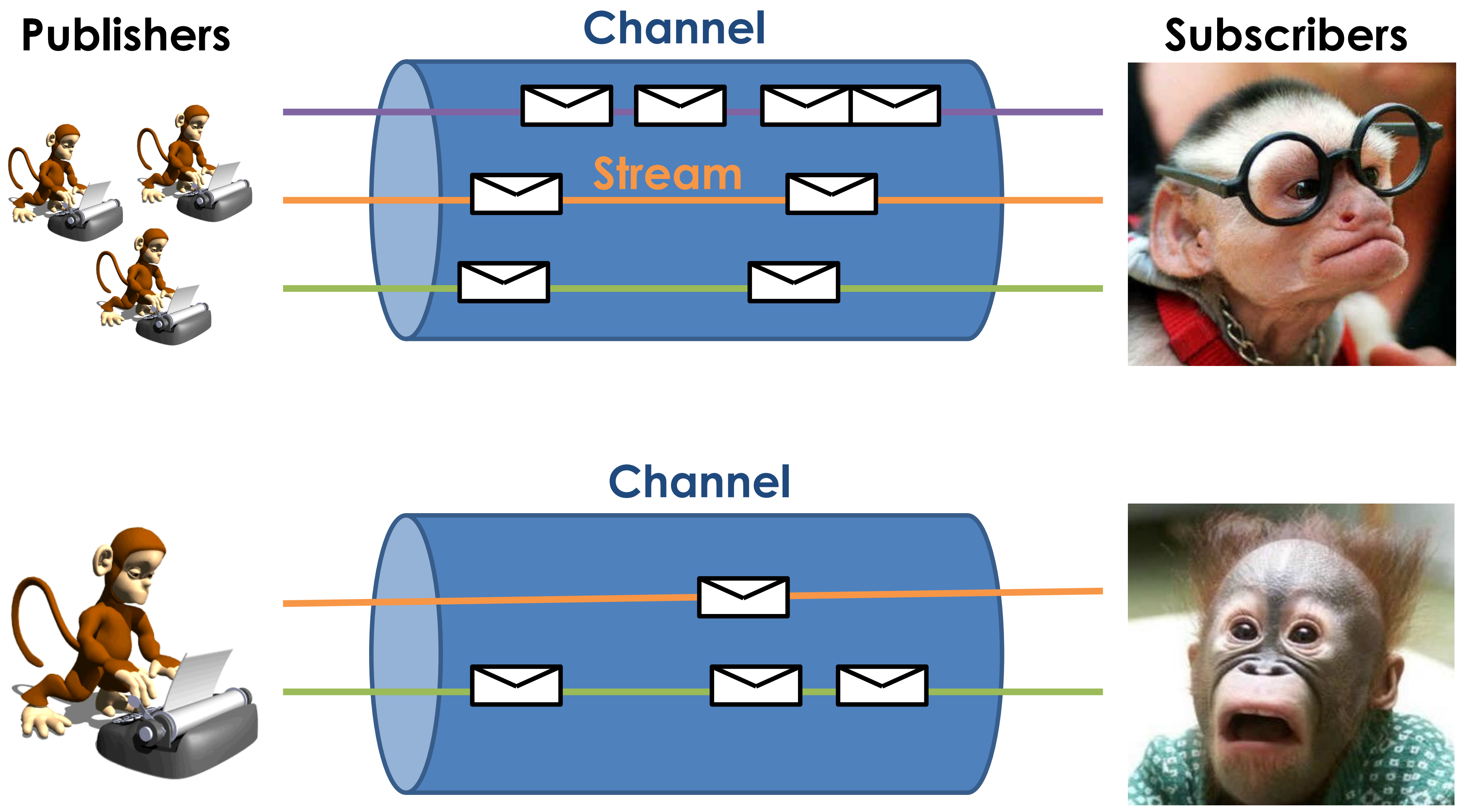
We are in a new world

Multi-core, Multi-socket, Cloud...

UDP, IPC, InfiniBand, RDMA, PCI-e

2. What features are really needed?

Messaging



A library, not a framework, on which other abstractions and applications can be built

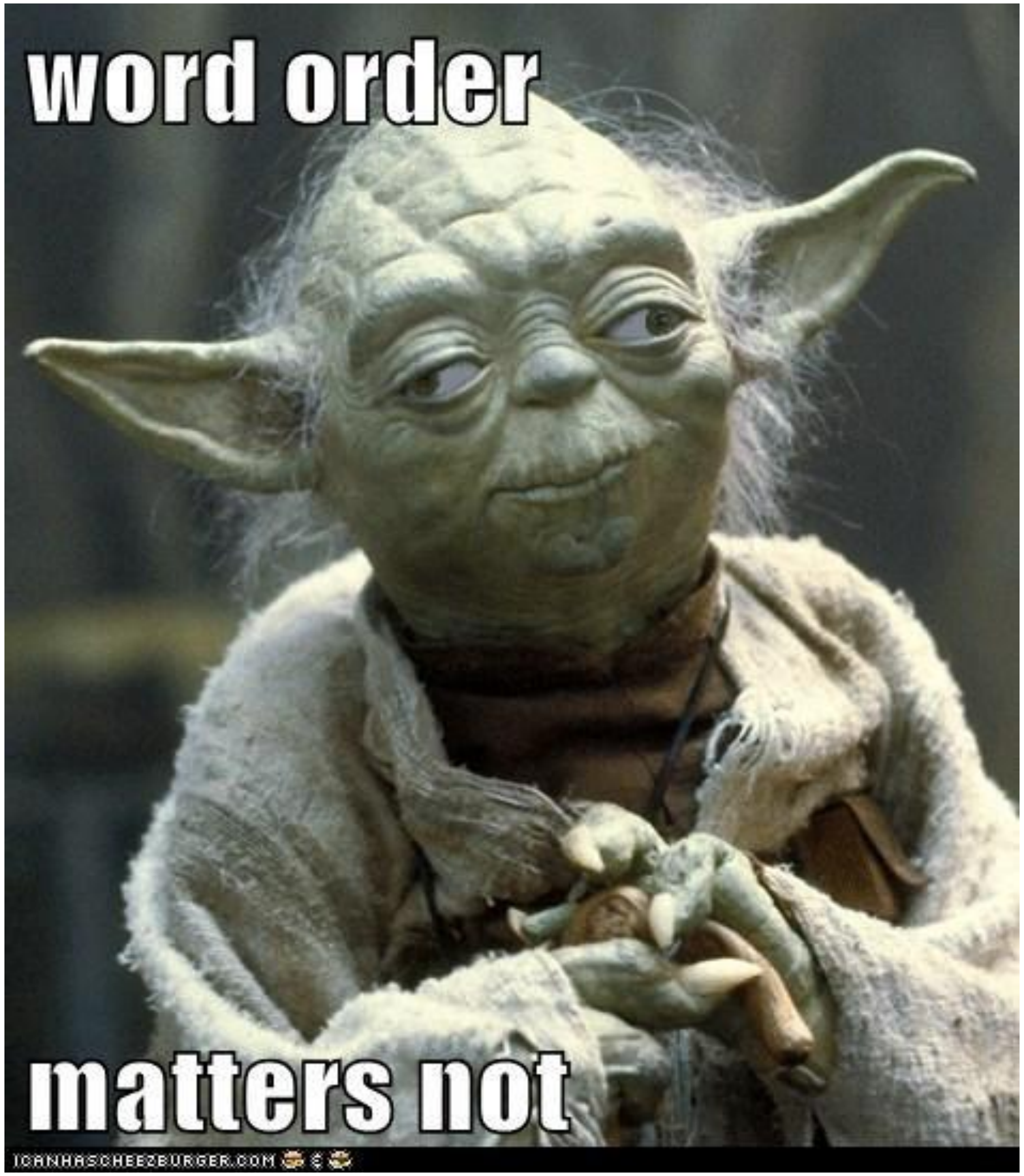
Composable Design

OSI layer 4 Transport for message oriented streams

OSI Layer 4 (Transport) Services

1. Connection Oriented Communication
2. Reliability
3. Flow Control
4. Congestion Avoidance/Control
5. Multiplexing

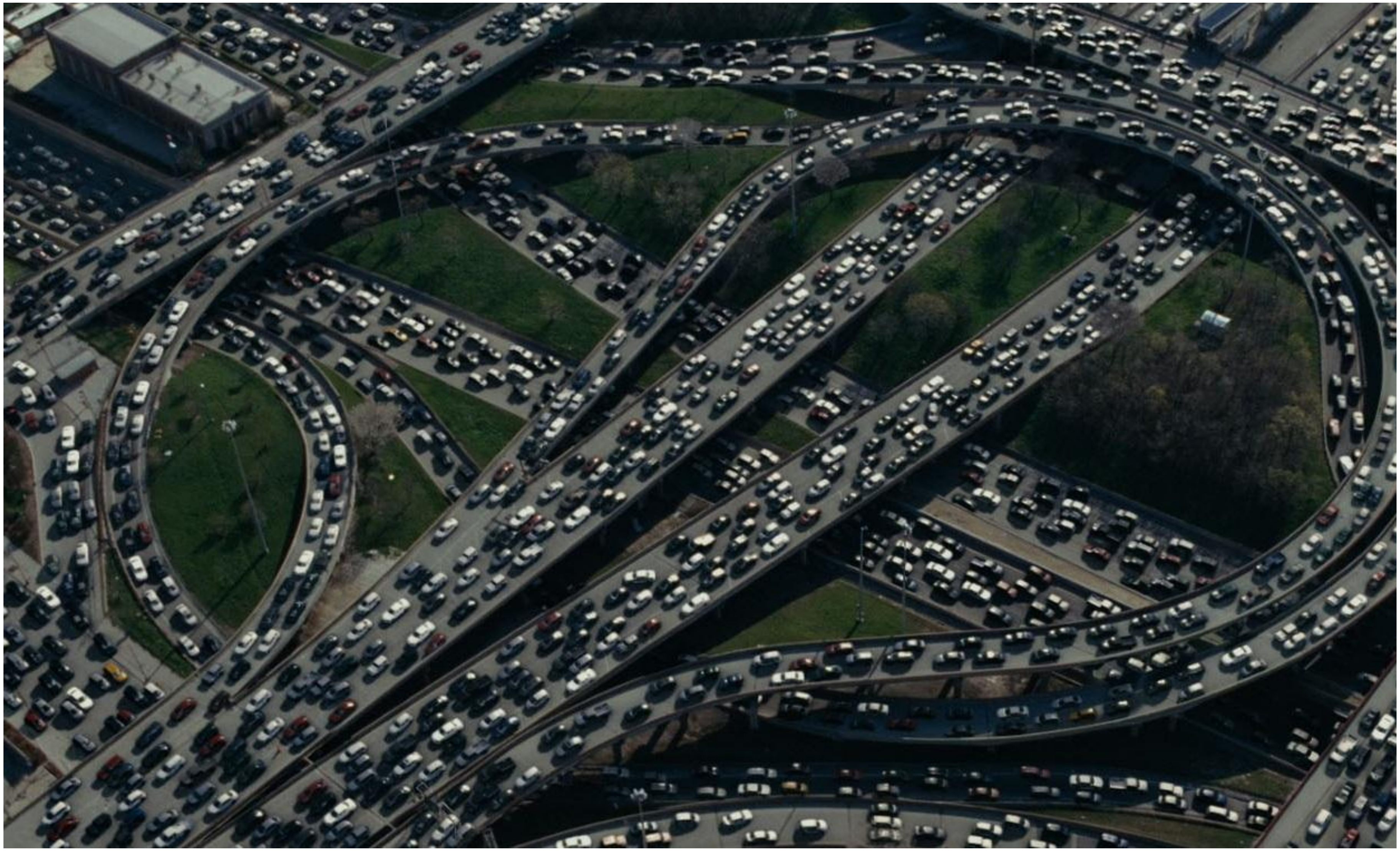
Connection Oriented Communication



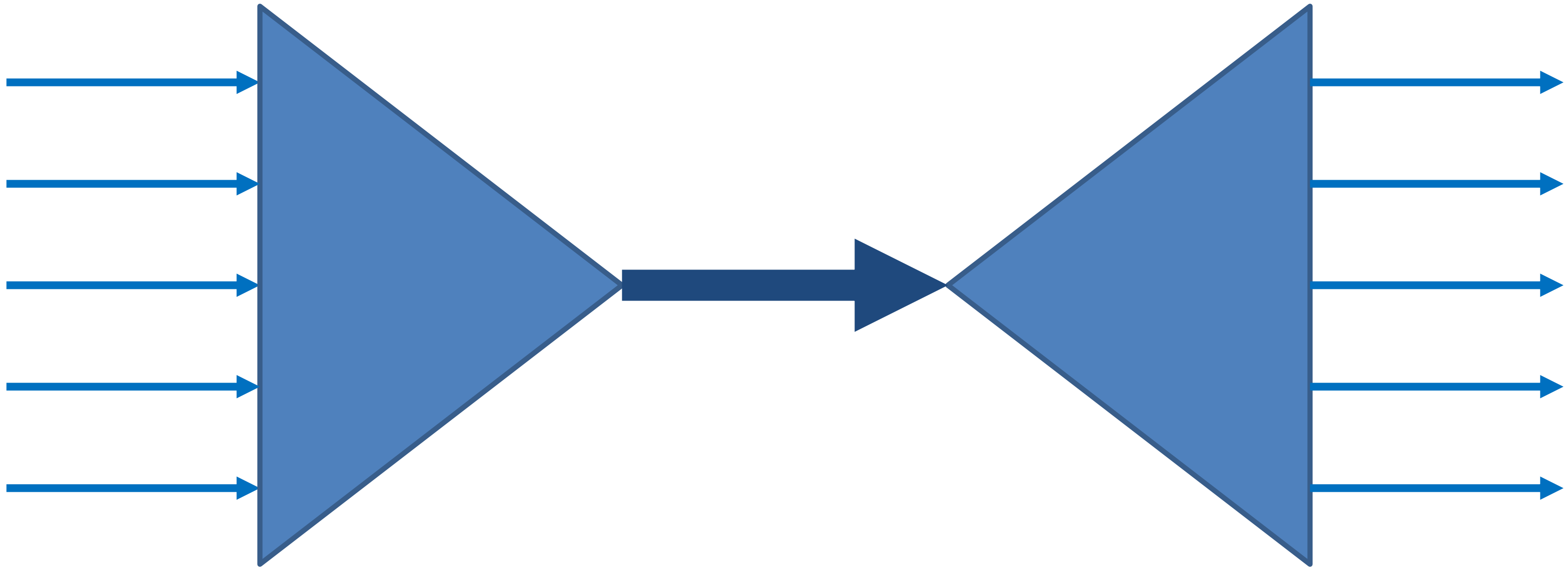
Reliability

Flow Control

Congestion Avoidance/Control

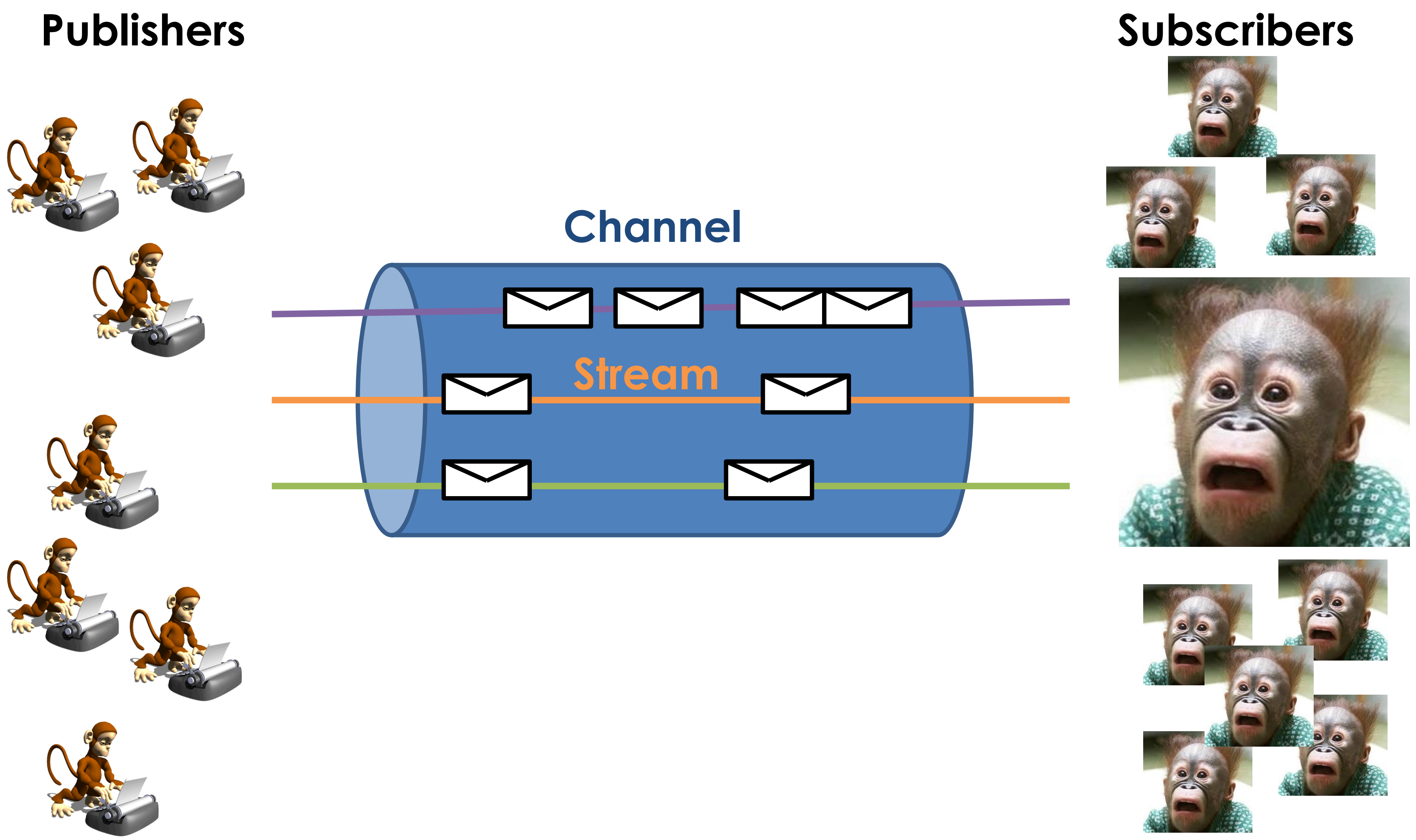


Multiplexing



Multi-Everything World!

Multi-Everything World



3. How does one design for this?

Design Principles

1. Garbage free in steady state running
2. Smart Batching in the message path
3. Lock-free algos in the message path
4. Non-blocking IO in the message path
5. No exceptional cases in message path
6. Apply the *Single Writer Principle*
7. Prefer unshared state
8. Avoid unnecessary data copies

It’s all about 3 things

It’s all about 3 things

1. System Architecture

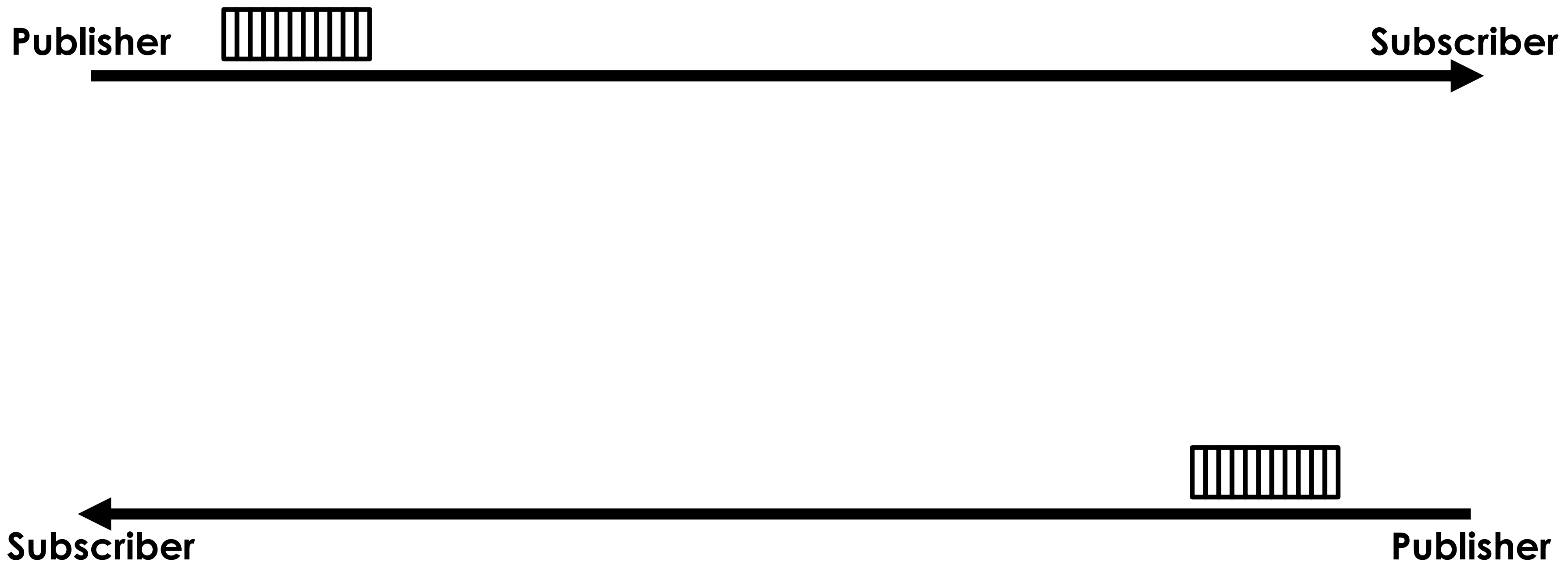
It’s all about 3 things

1. System Architecture
2. Data Structures

It's all about 3 things

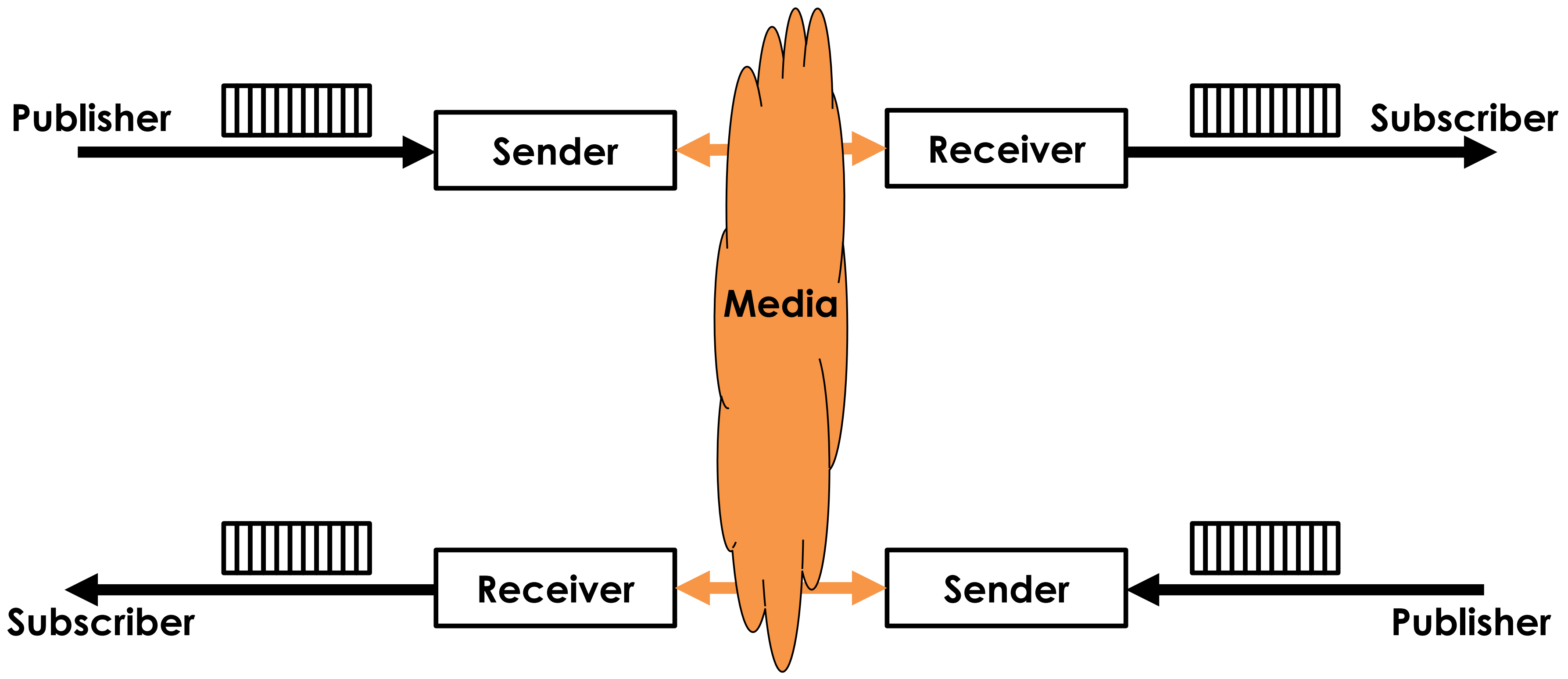
1. System Architecture
2. Data Structures
3. Protocols of Interaction

Architecture



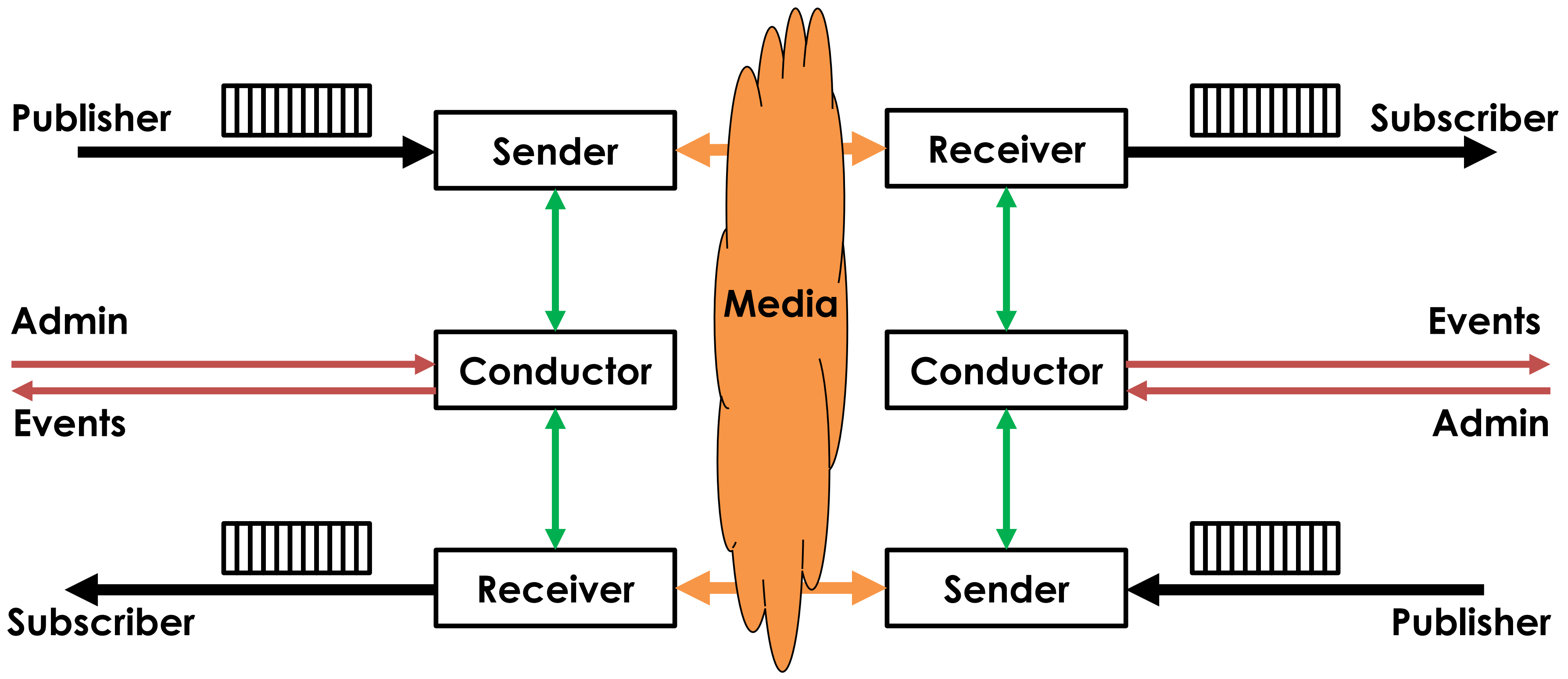
— IPC Log Buffer

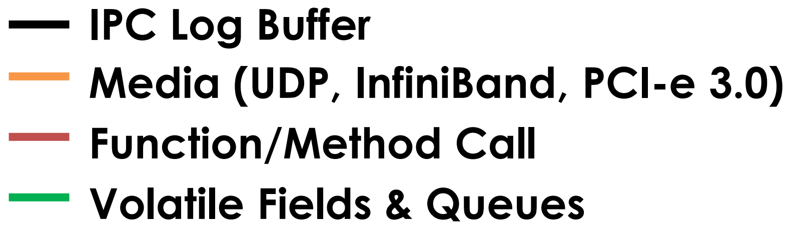
Architecture



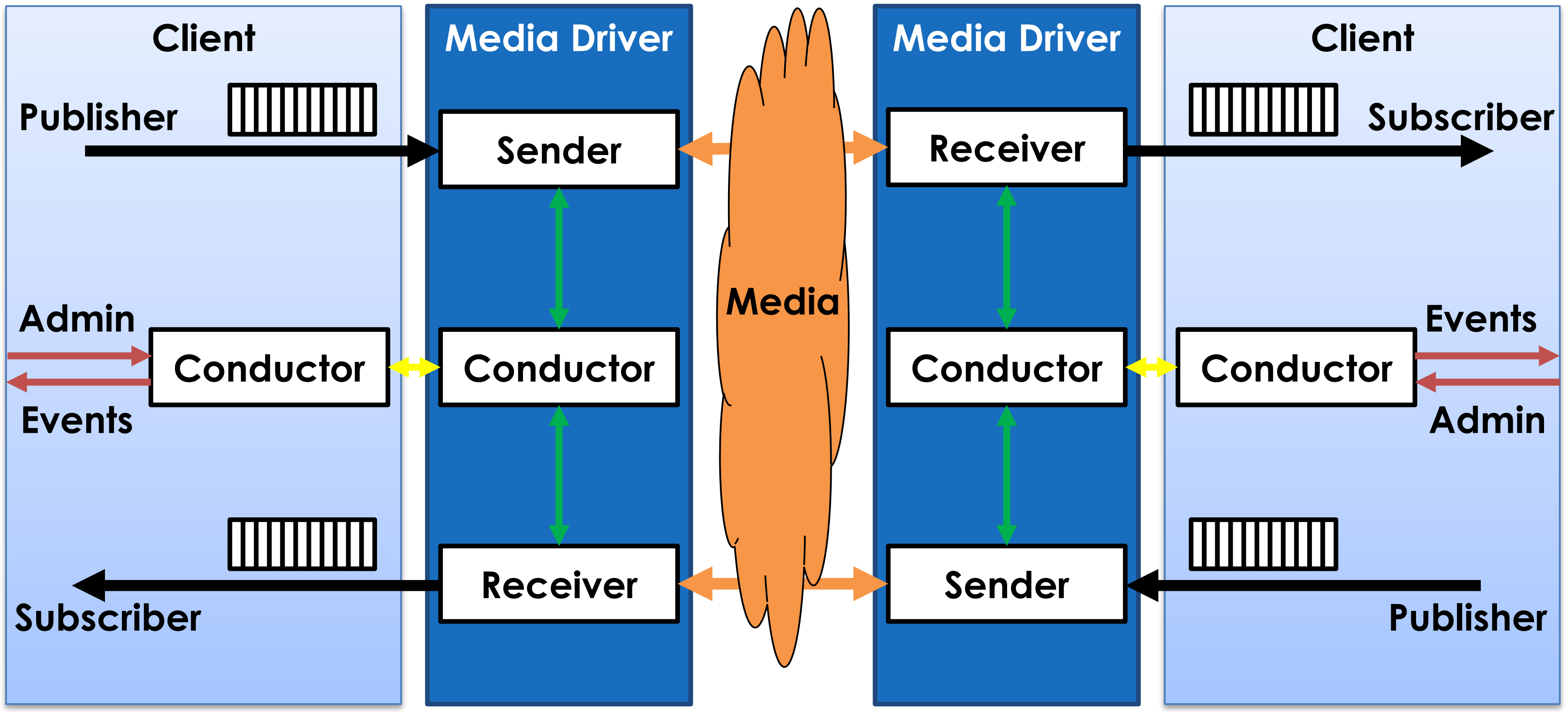
▬ IPC Log Buffer
🟧 Media (UDP, InfiniBand, PCI-e 3.0)

Architecture





Architecture



- ▬ IPC Log Buffer
- 🟧 Media (UDP, InfiniBand, PCI-e 3.0)
- 🟥 Function/Method Call
- 🟩 Volatile Fields & Queues
- 🟨 IPC Ring/Broadcast Buffer

Data Structures

- Maps
- IPC Ring Buffers
- IPC Broadcast Buffers
- ITC Queues
- Atomic Buffers
- Log Buffers

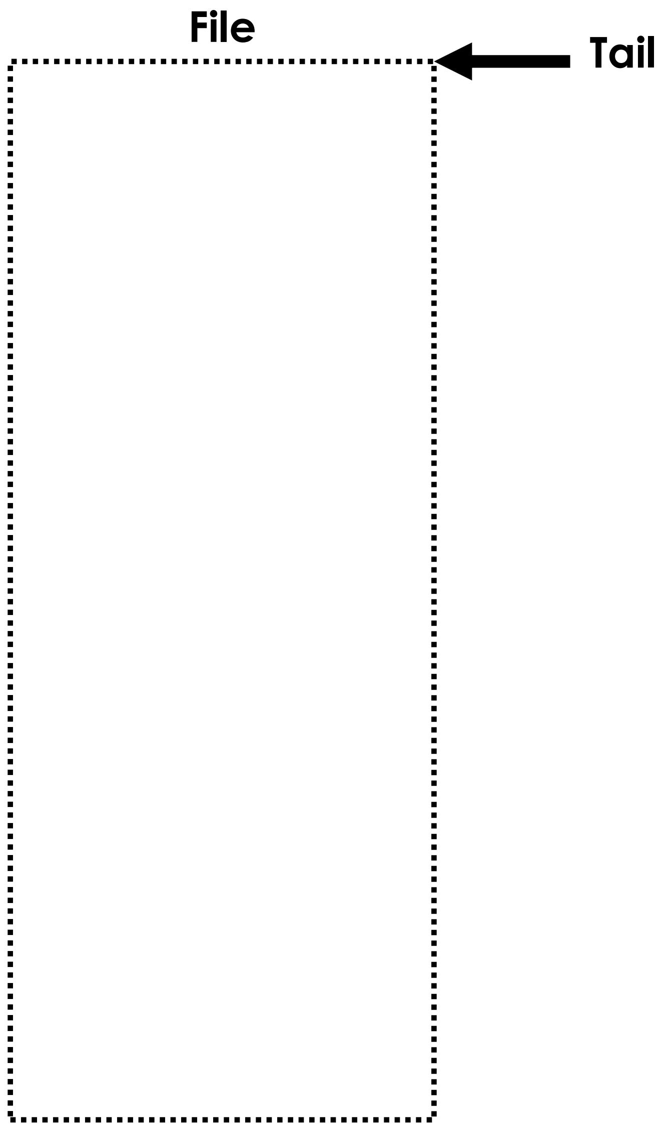
What Aeron does

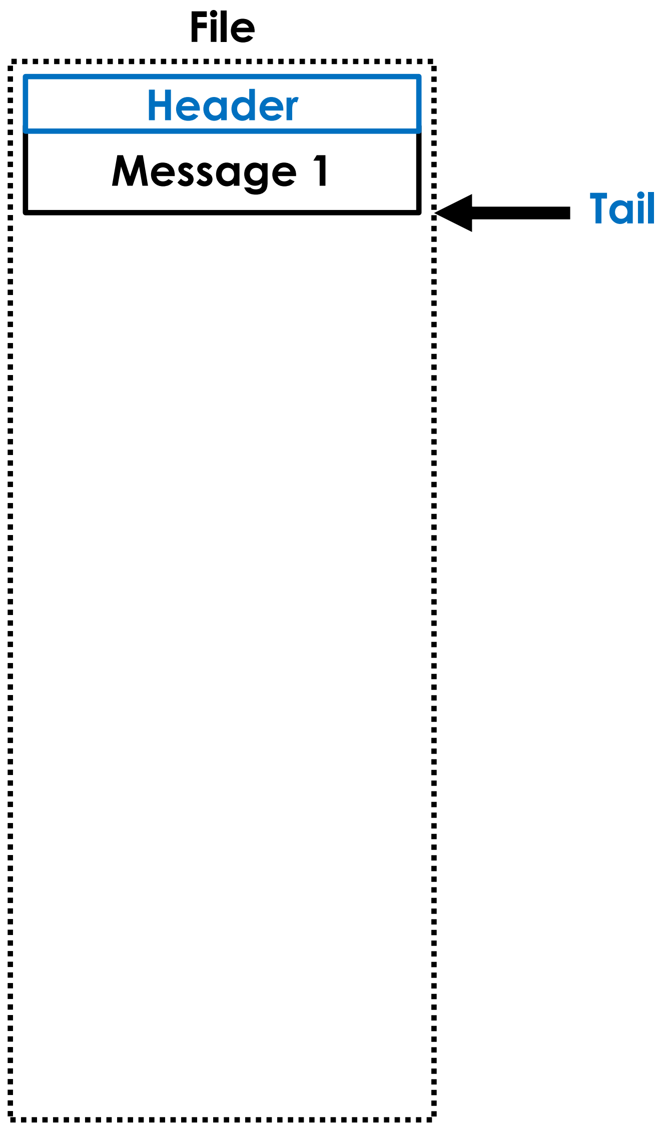
Creates a

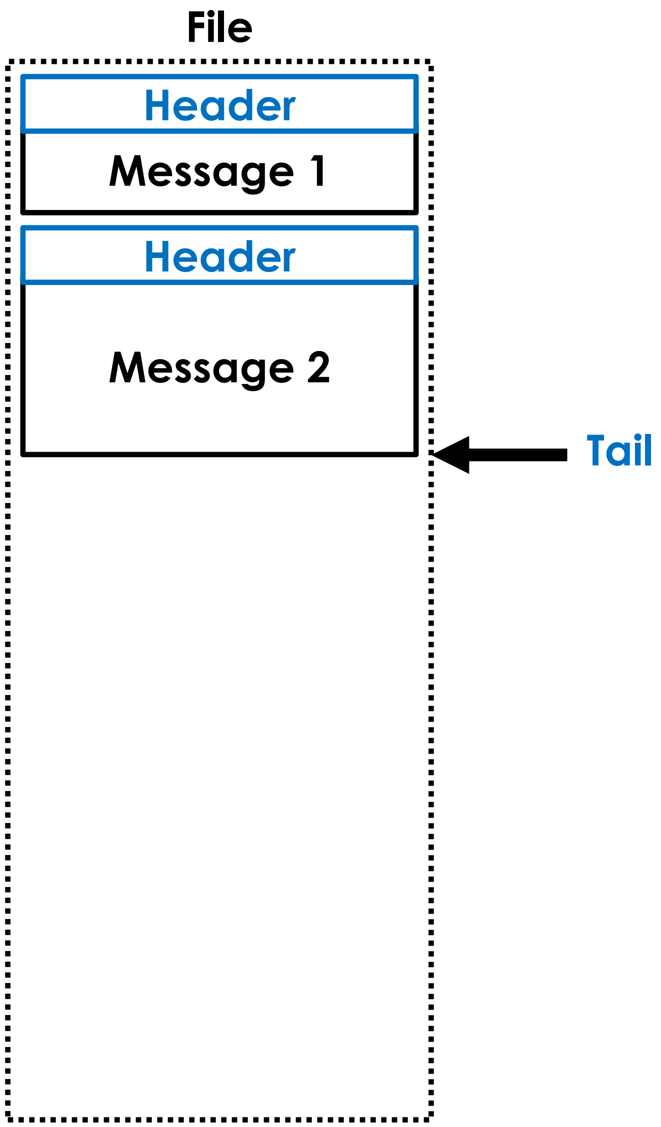
replicated append-only log

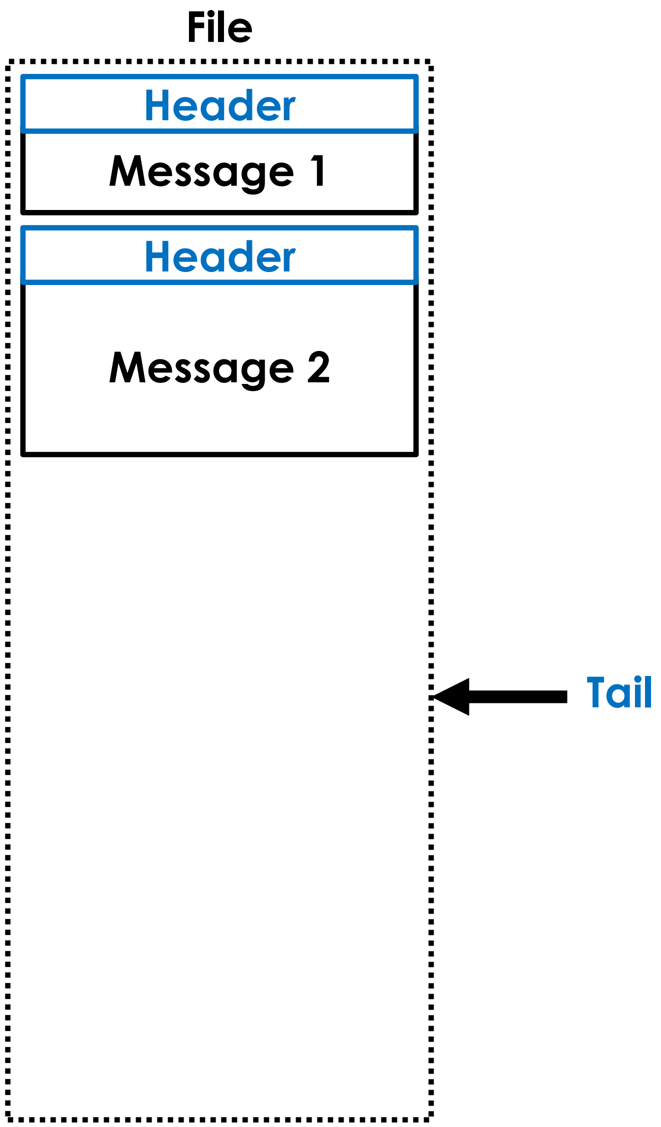
of messages

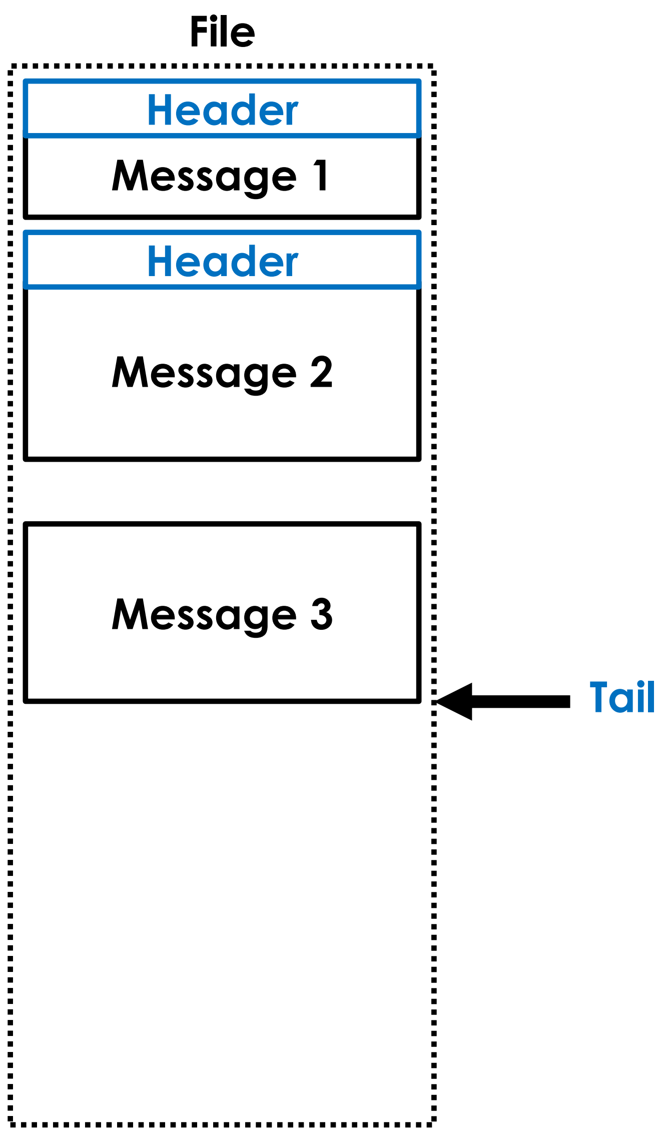
How would you design a log?

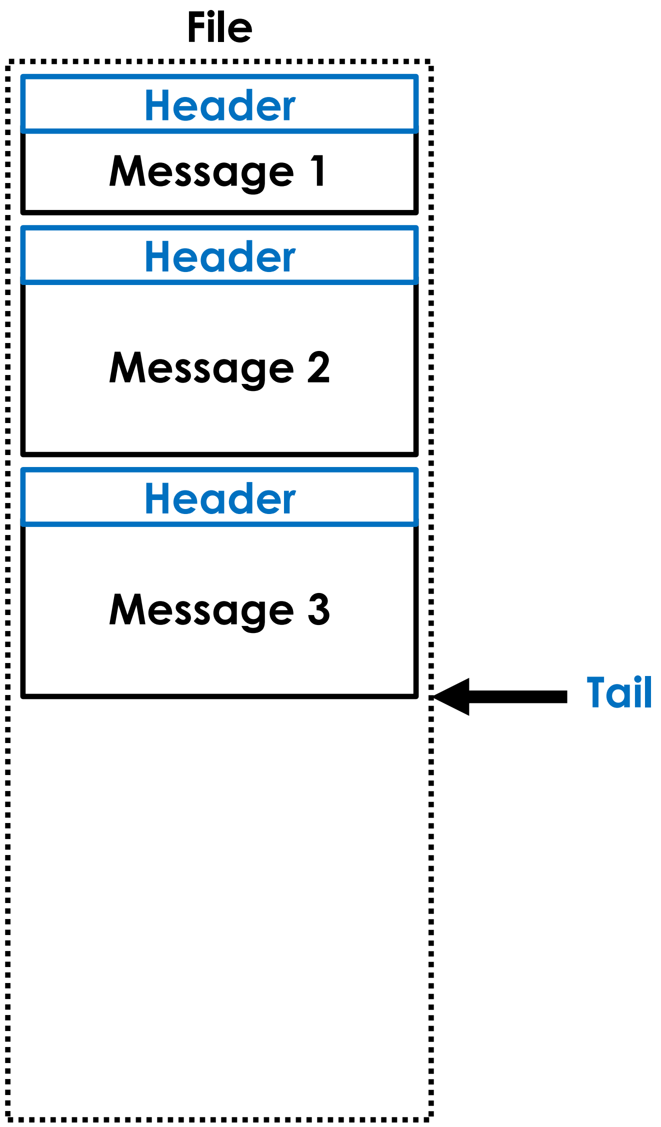

File
Tail


File
Header
Message 1
Tail


File
Header
Message 1
Header
Message 2
Tail


File
Header
Message 1
Header
Message 2
Tail


File
Header
Message 1
Header
Message 2
Message 3
Tail

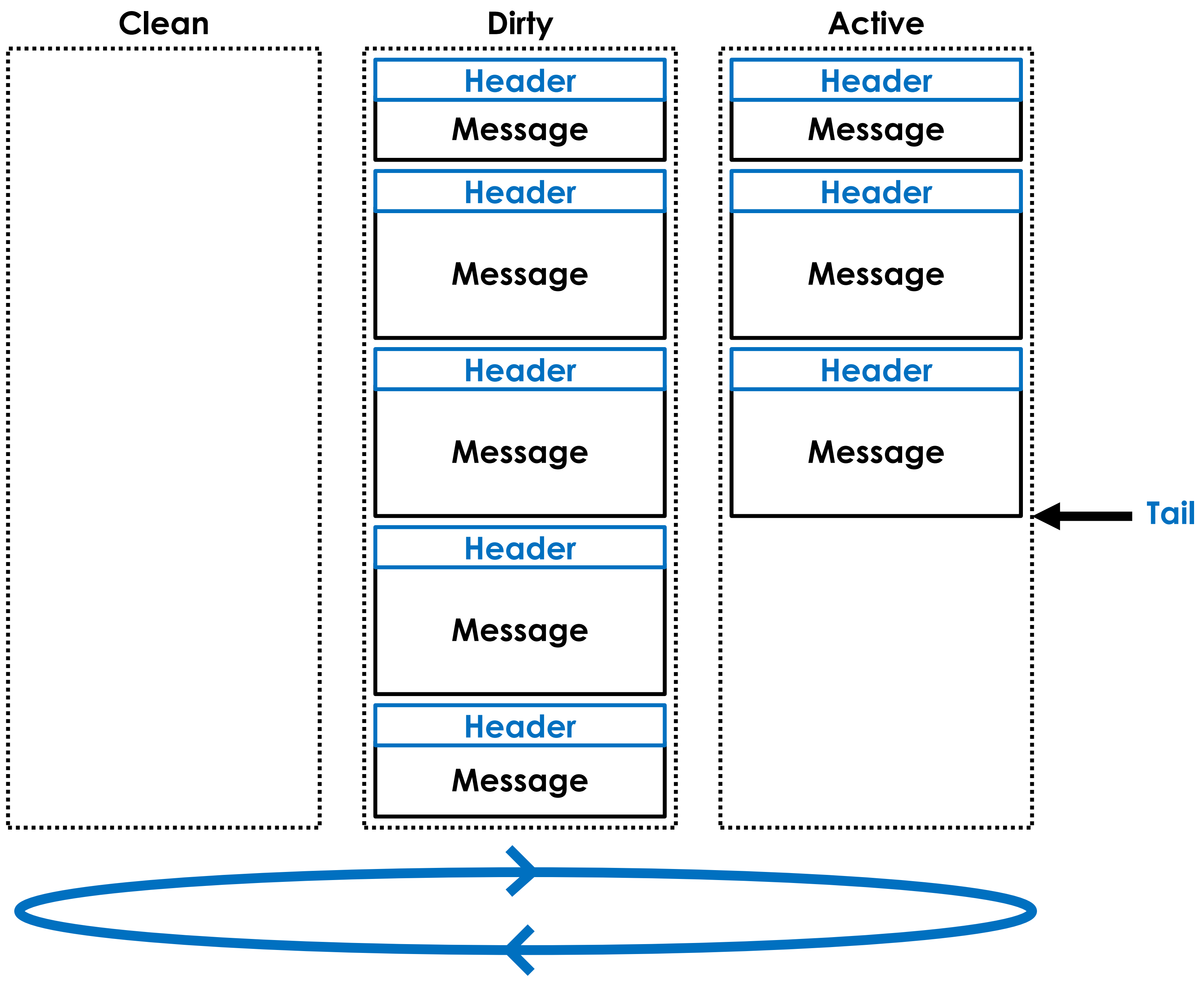

File
Header
Message 1
Header
Message 2
Header
Message 3
Tail

Append-only data structures can be safe to read without locks

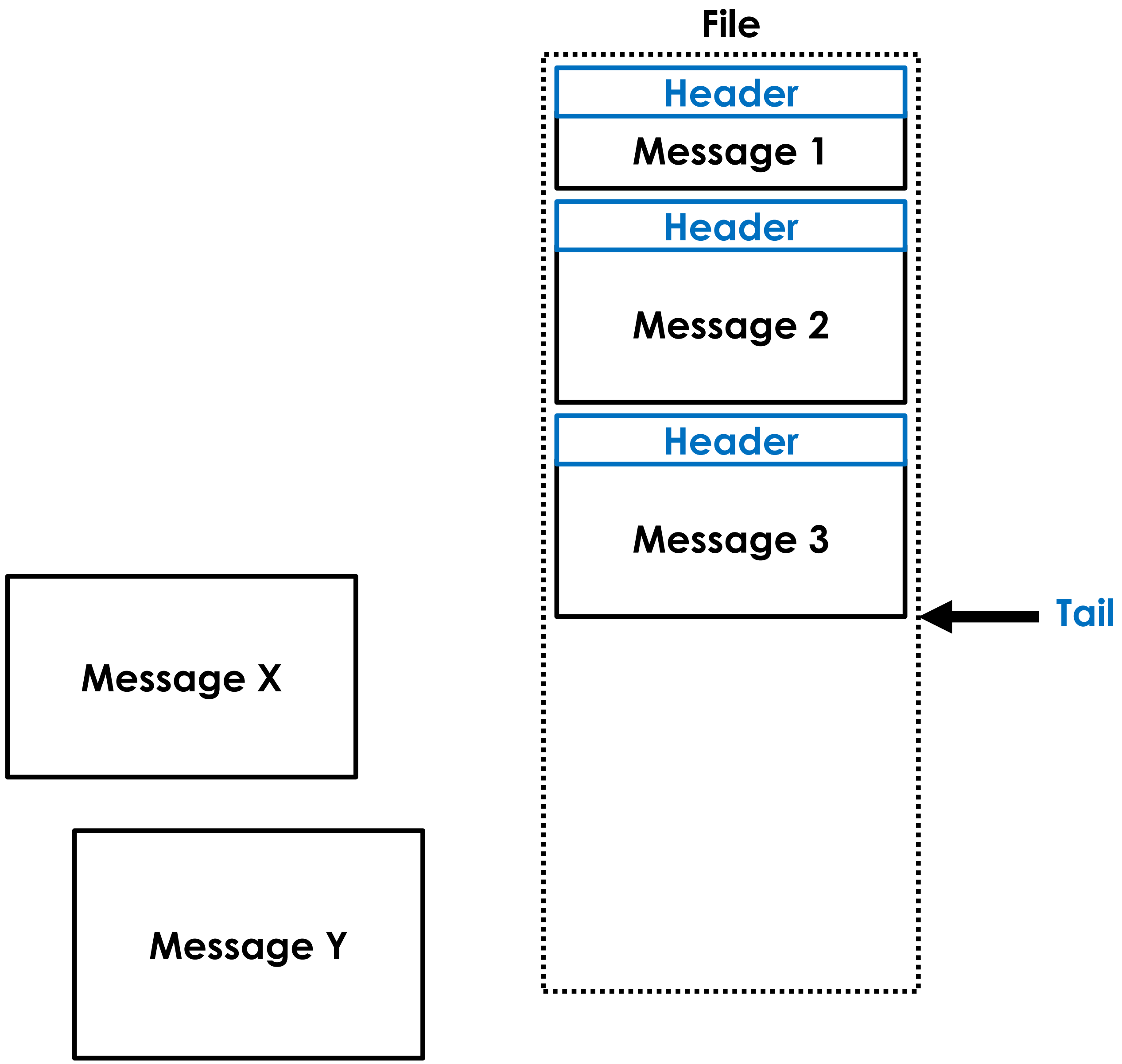
One big file that goes on forever?

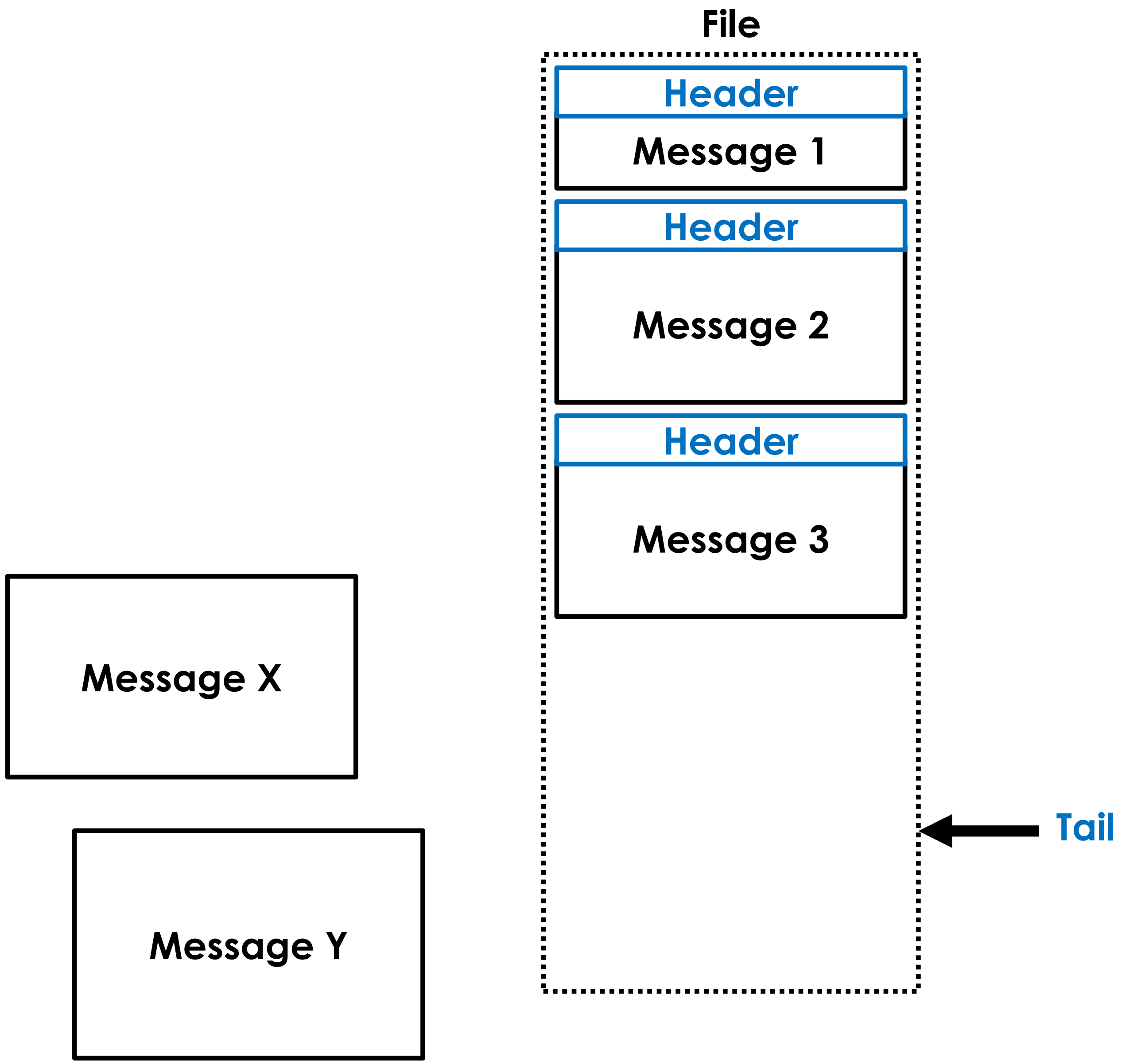
No!!!

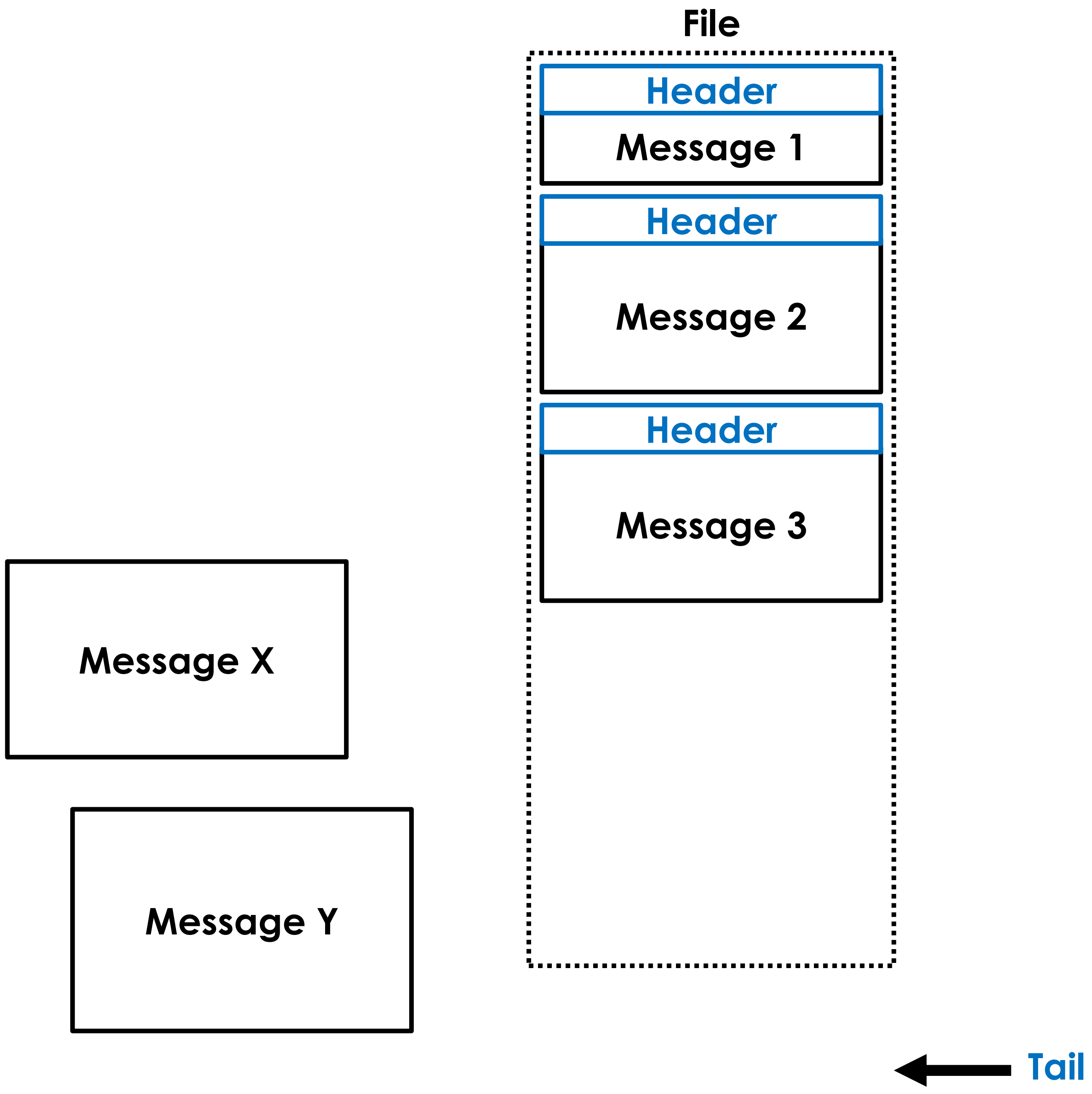
Page faults, page cache churn, VM pressure, ...

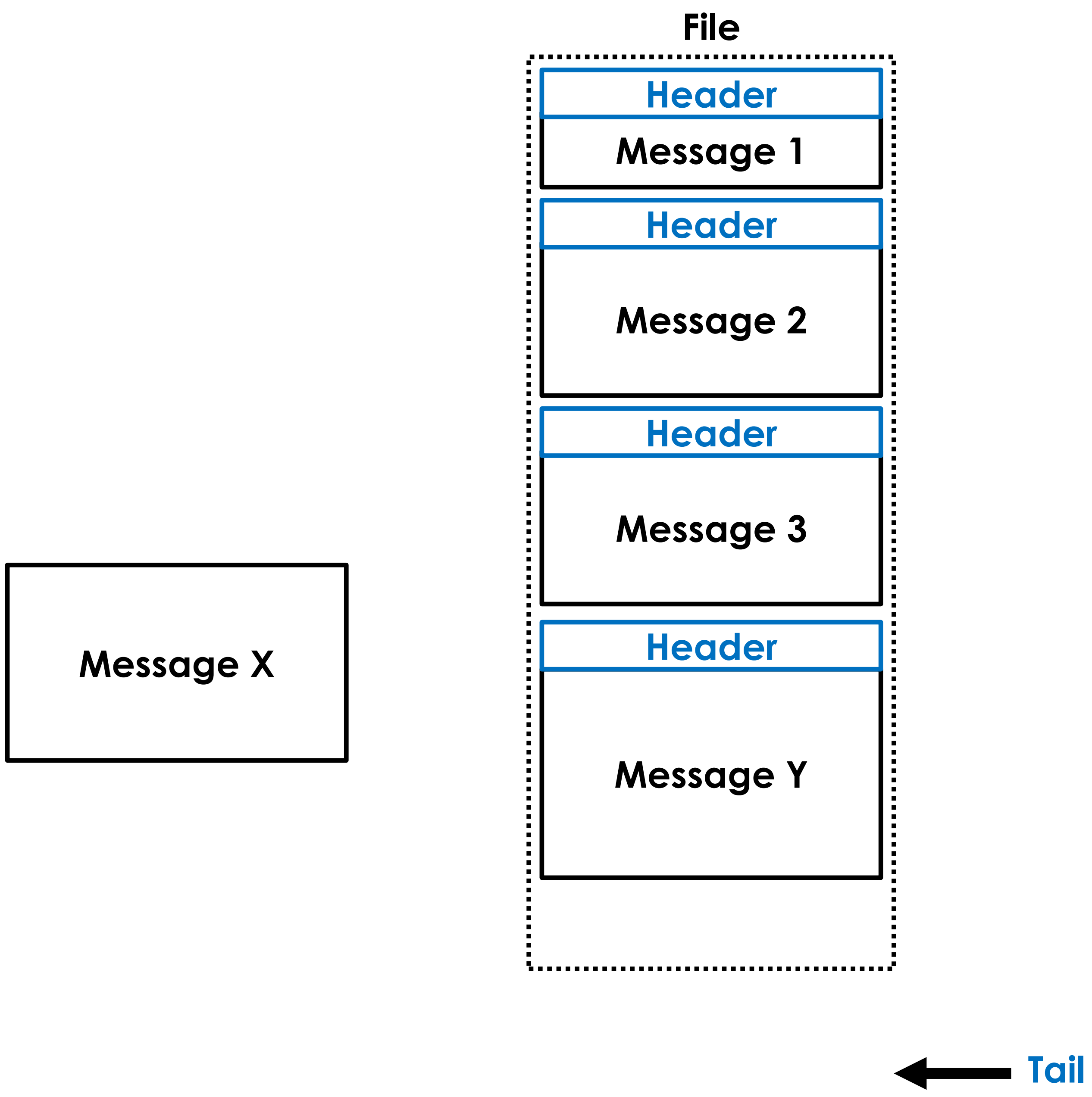

Clean
Dirty
Active
Header
Message
Header
Message
Header
Message
Header
Message
Header
Message
Header
Message
Header
Message
Header
Message
Tail

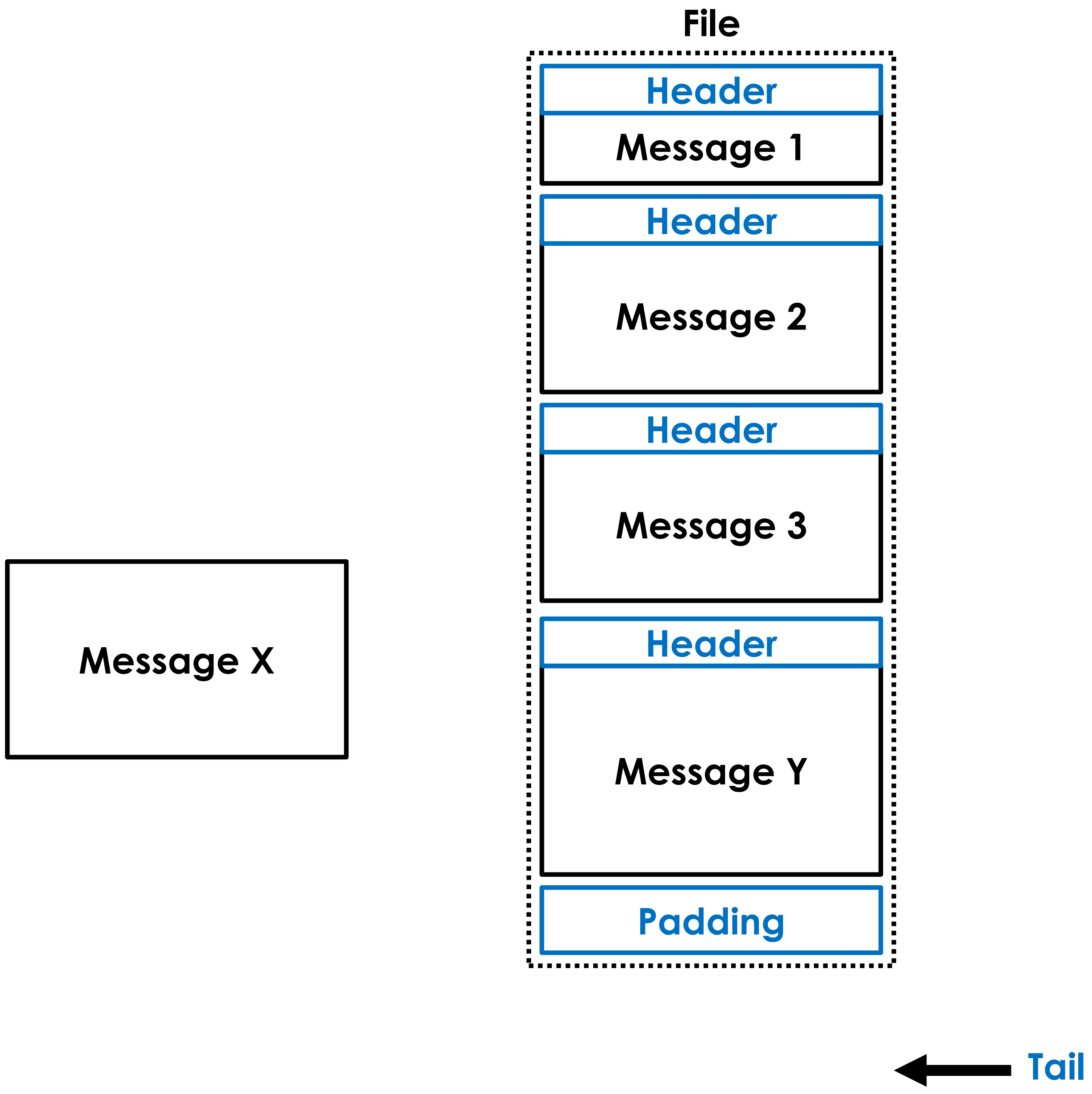
You don’t block publishers?

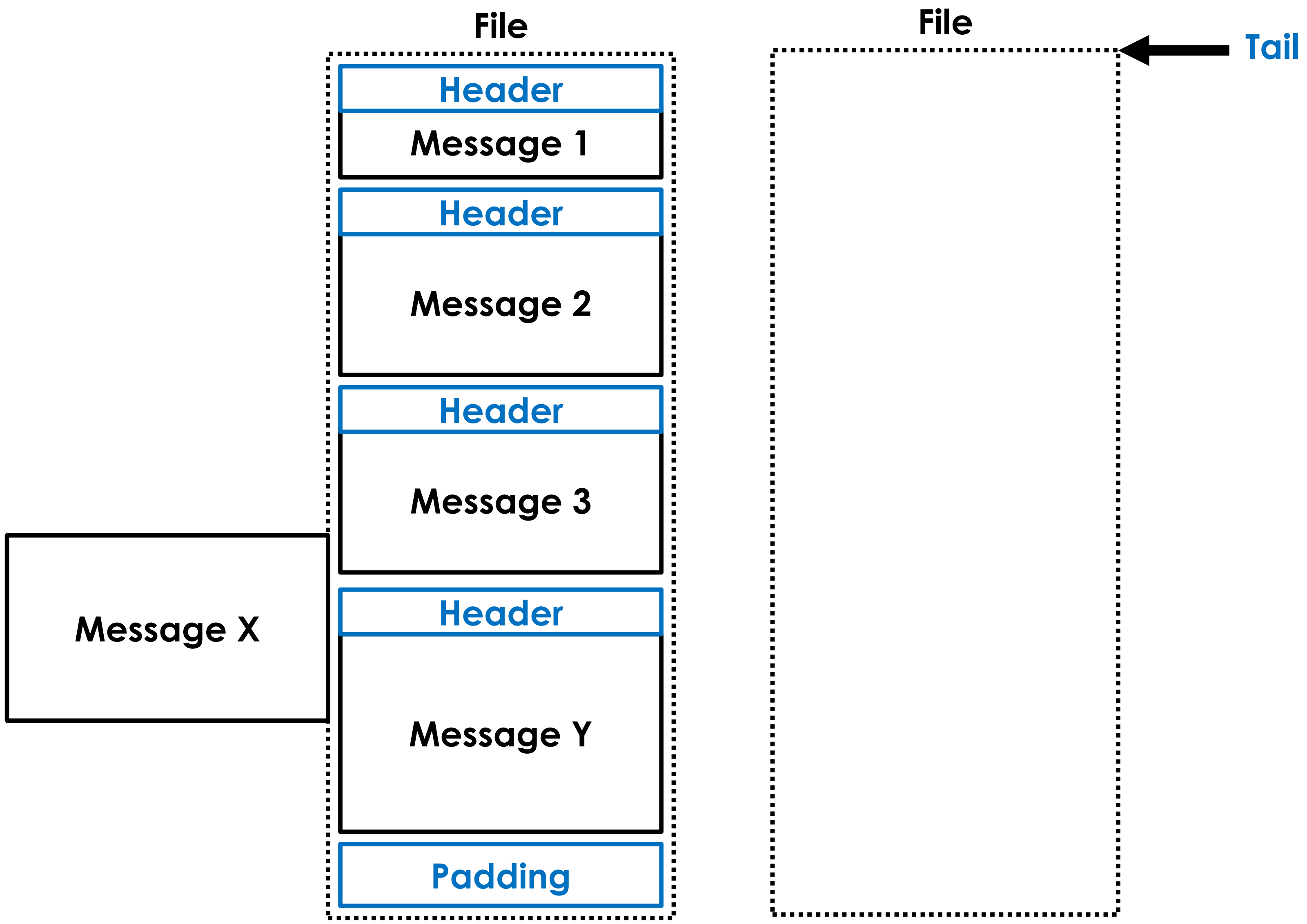

File
Header
Message 1
Header
Message 2
Header
Message 3
Tail
Message X
Message Y

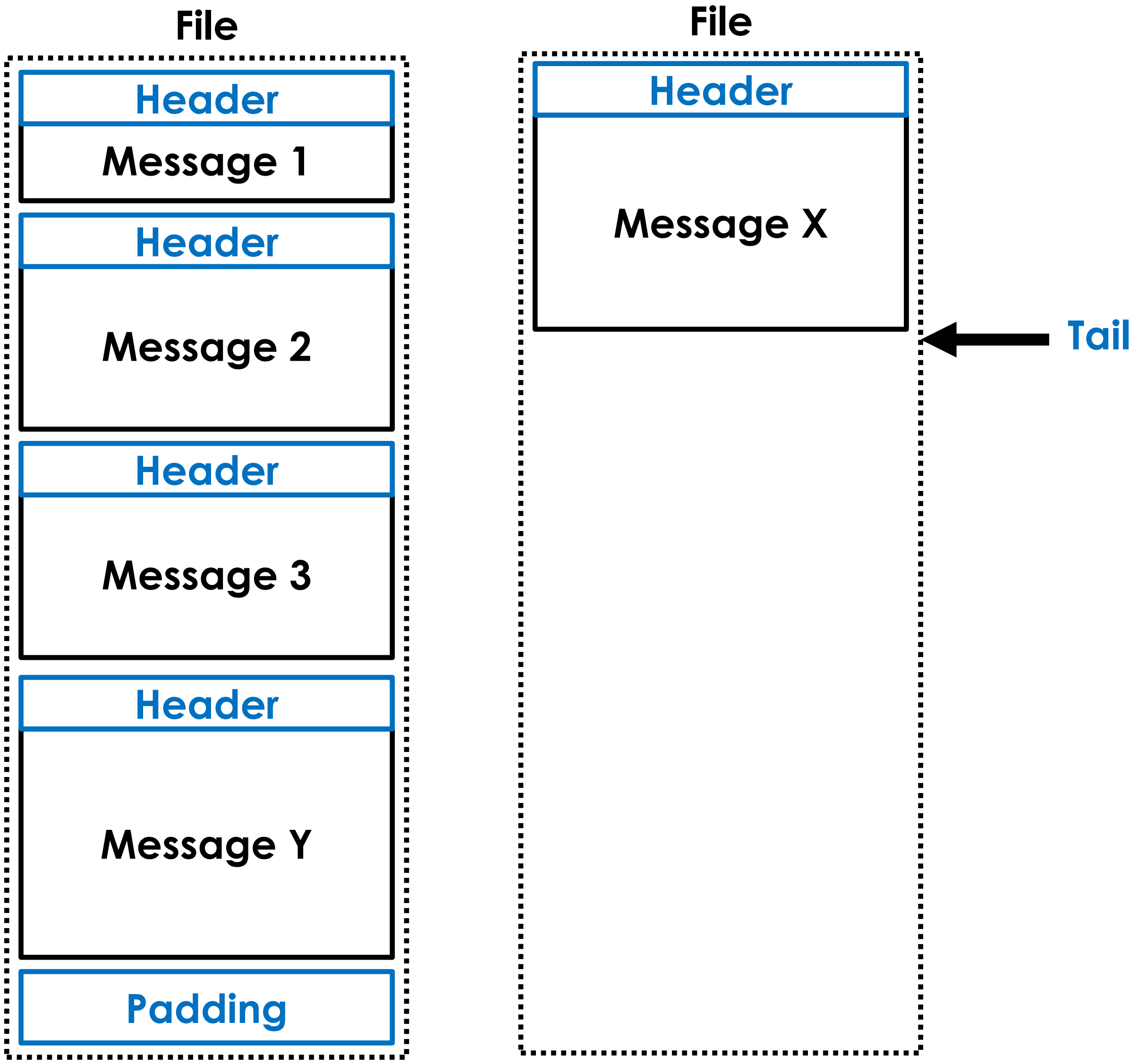

File
Header
Message 1
Header
Message 2
Header
Message 3
Tail
Message X
Message Y


File
Header
Message 1
Header
Message 2
Header
Message 3
Message X
Message Y
Tail


File
Header
Message 1
Header
Message 2
Header
Message 3
Header
Message Y
Message X
Tail


File
Header
Message 1
Header
Message 2
Header
Message 3
Header
Message Y
Padding
Message X
Tail


File
Header
Message 1
Header
Message 2
Header
Message 3
Header
Message Y
Padding
Message X
File
Tail


File
Header
Message 1
Header
Message 2
Header
Message 3
Header
Message Y
Padding
File
Header
Message X
Tail

What’s in a header?

Data Message Header

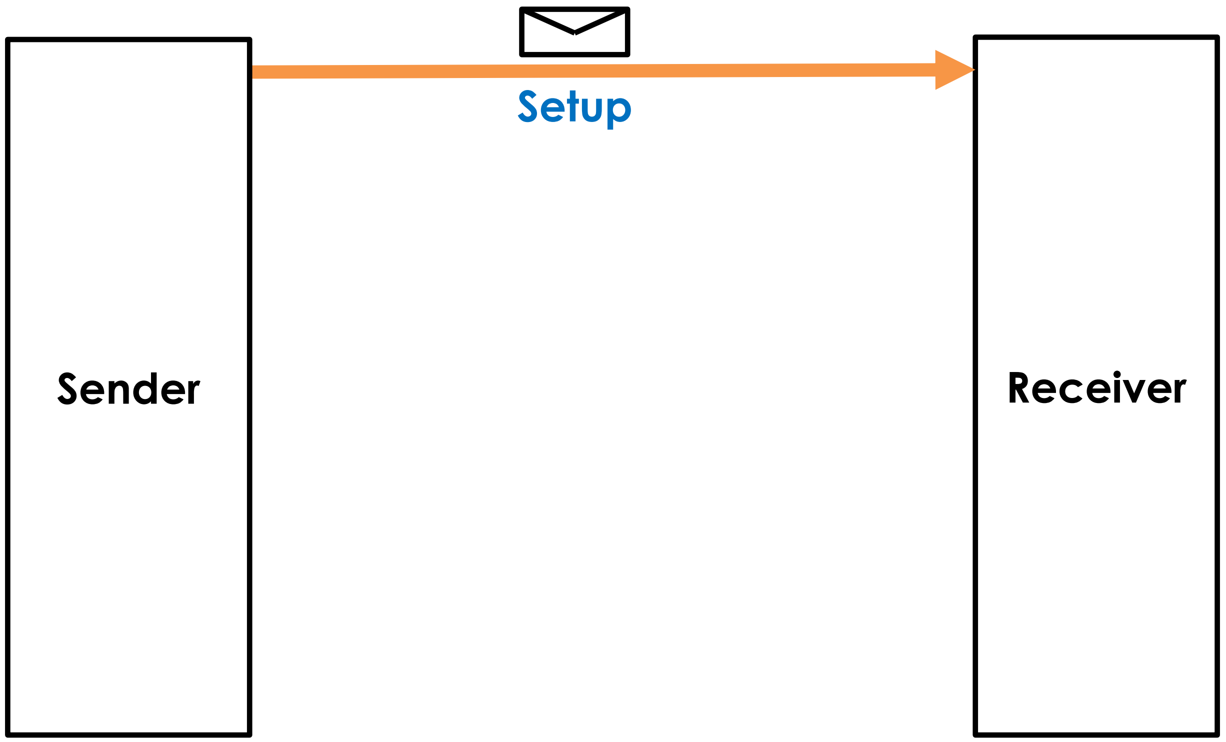
```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-------------------------------------------------------------+
 |R|                       Frame Length                          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-------------------------------+
 |    Version    |B|E|   Flags   |             Type              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-------------------------------+
 |R|                       Term Offset                           |
 +-+-------------------------------------------------------------+
 |                         Session ID                            |
 +---------------------------------------------------------------+
 |                          Stream ID                            |
 +---------------------------------------------------------------+
 |                           Term ID                             |
 +---------------------------------------------------------------+
 |                        Encoded Message                       ...
...                                                              |
 +---------------------------------------------------------------+
```

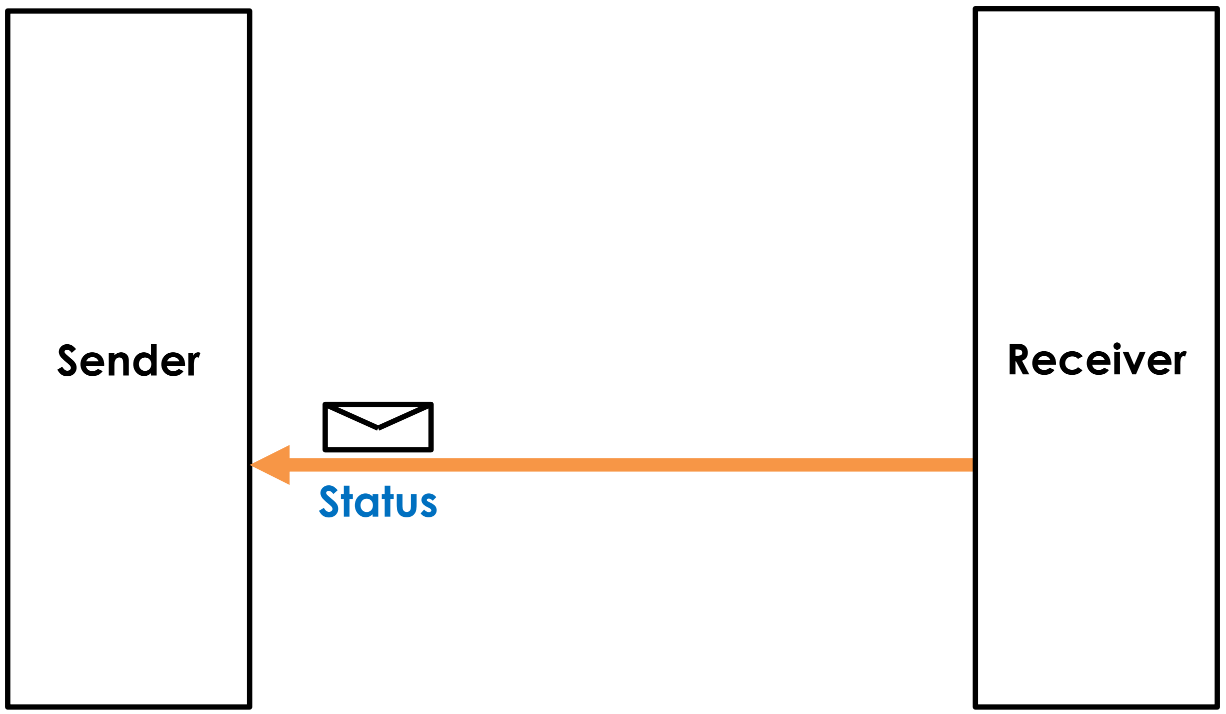
Unique identification of a byte within each stream across time (`streamId, sessionId, termId, termOffset`)

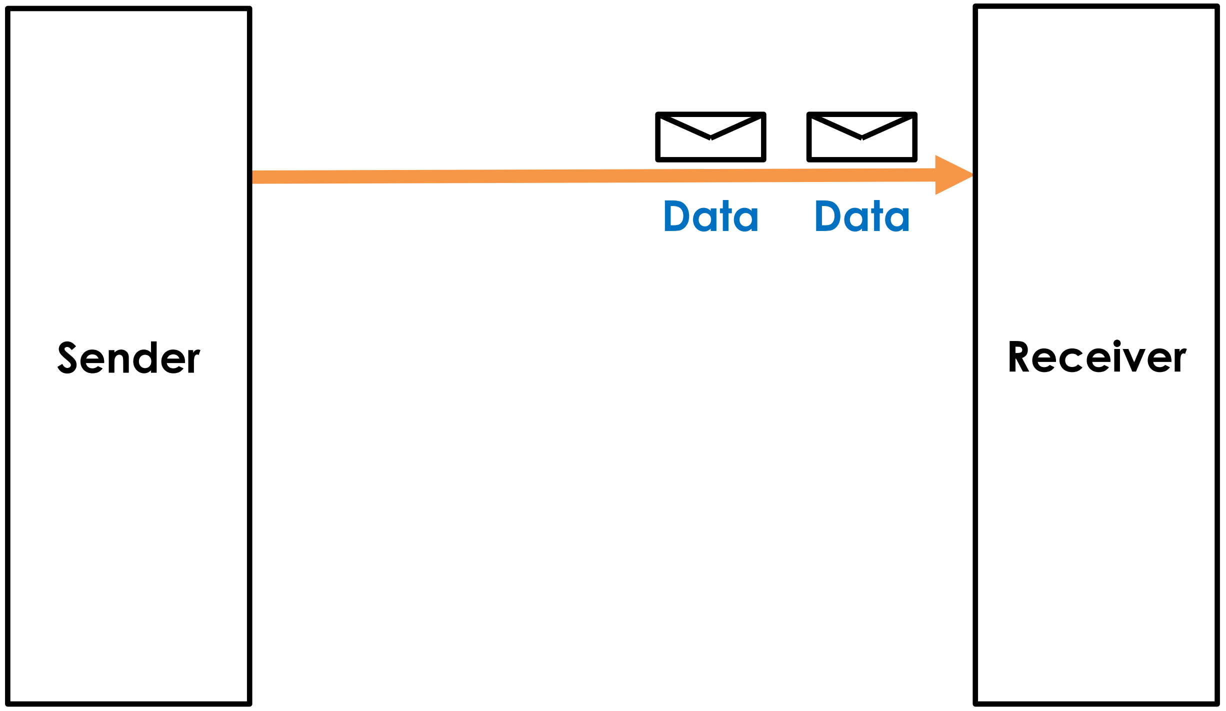
How do we replicate a log?

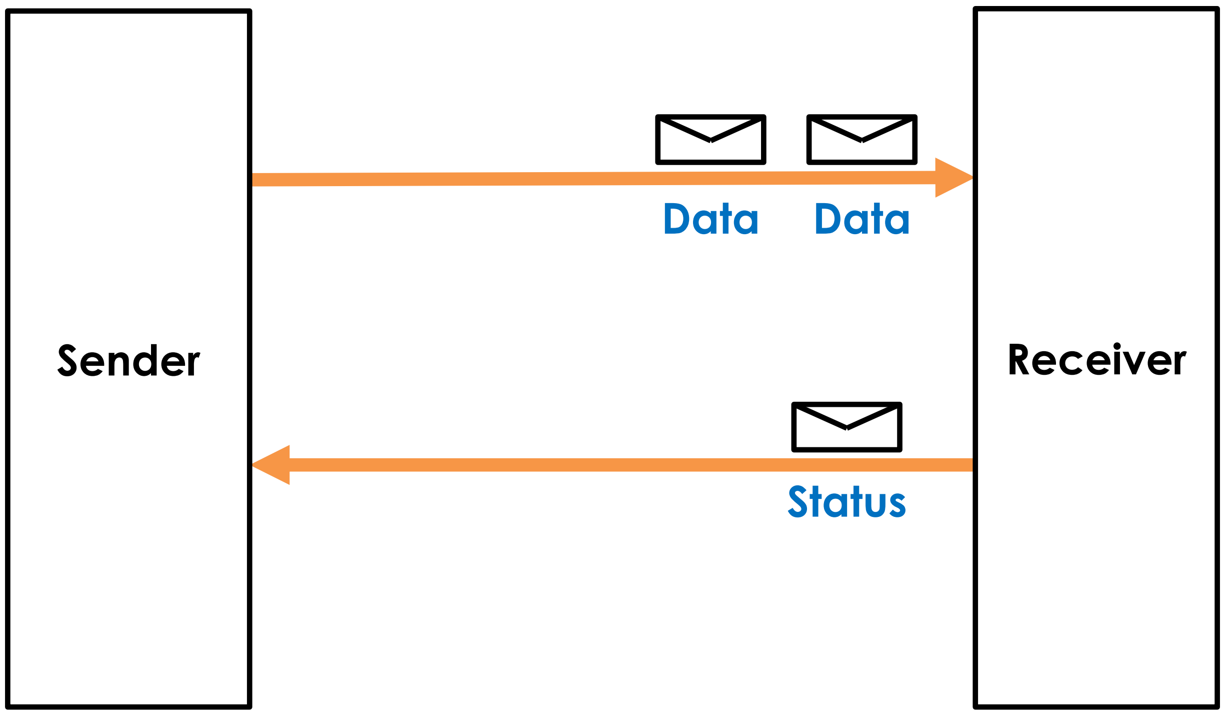
We need a **Protocol** *of messages*

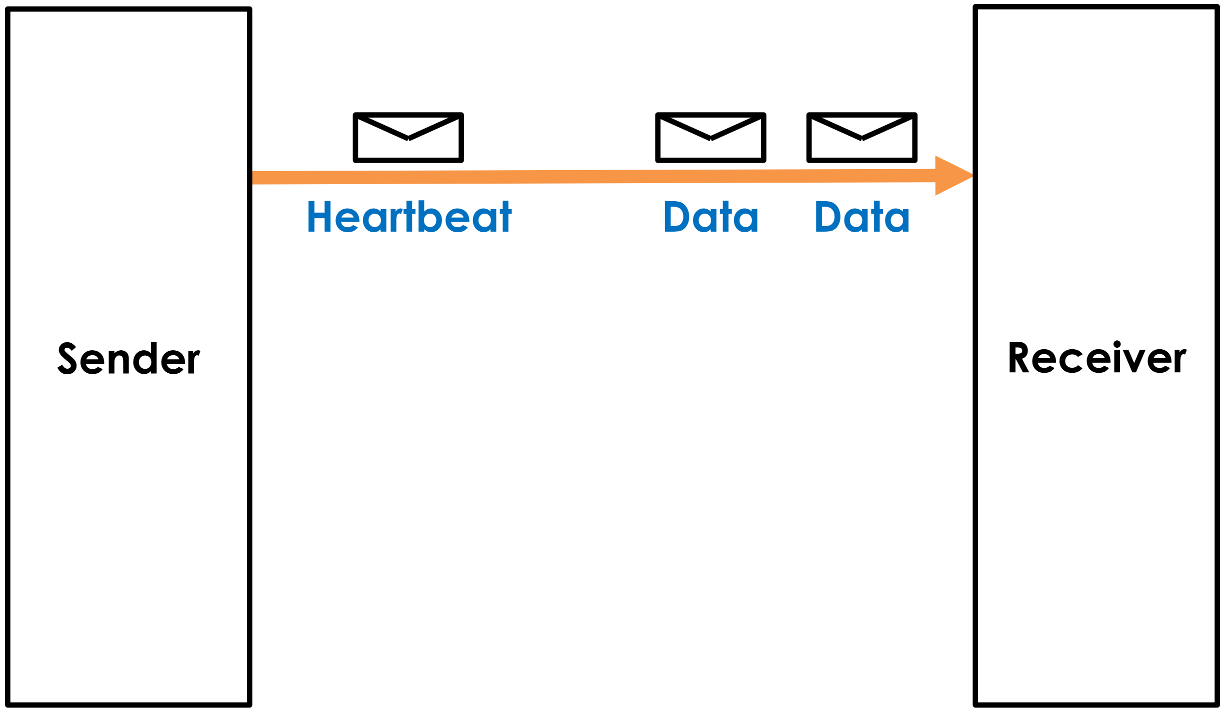
Sender
Receiver

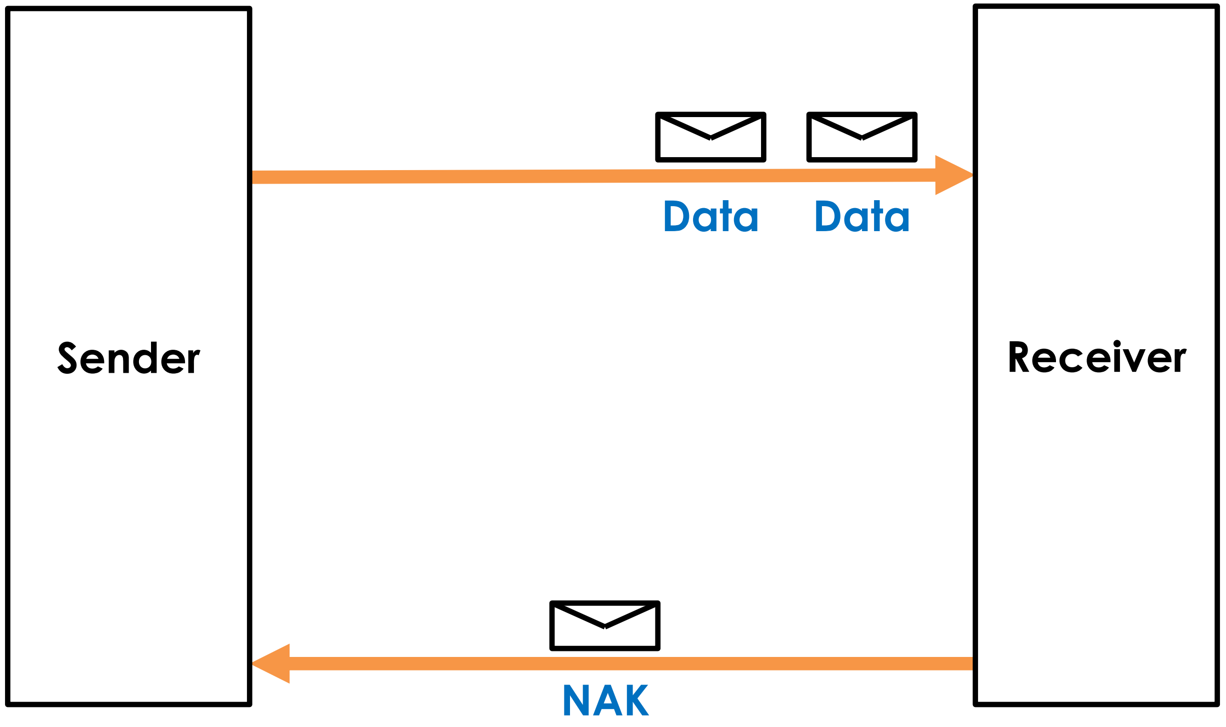

Sender
Setup
Receiver

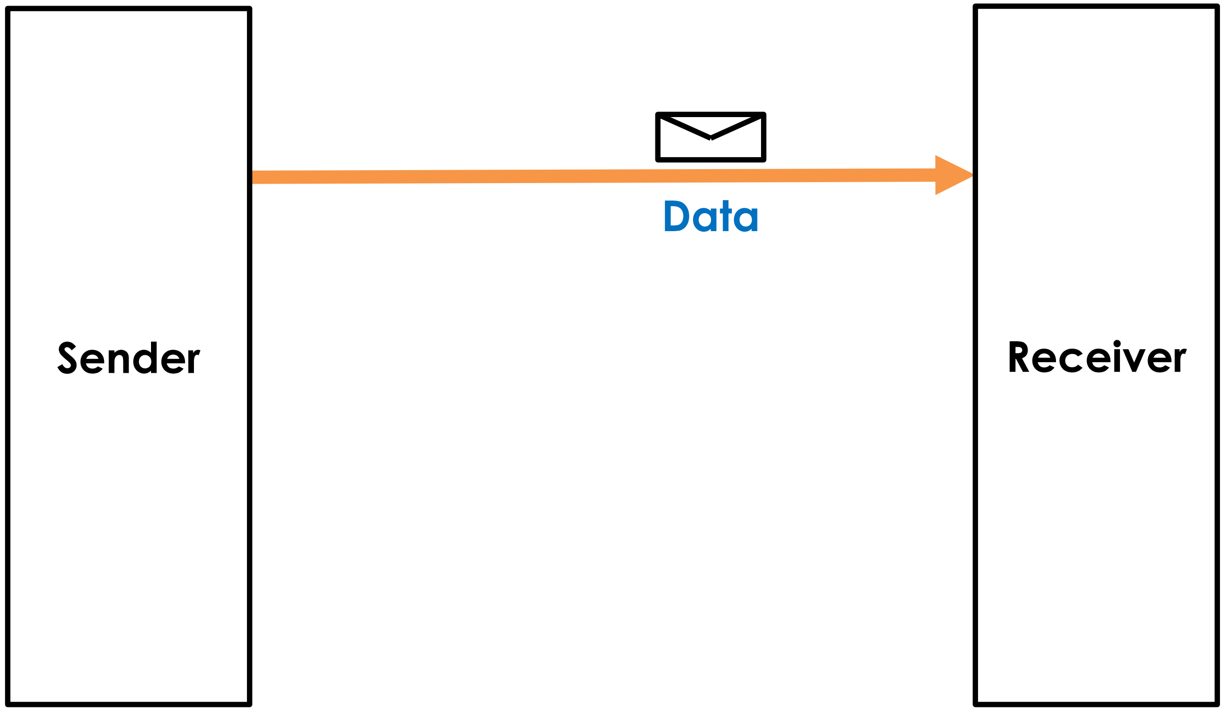

Sender
Receiver
Status


Sender
Data
Data
Receiver

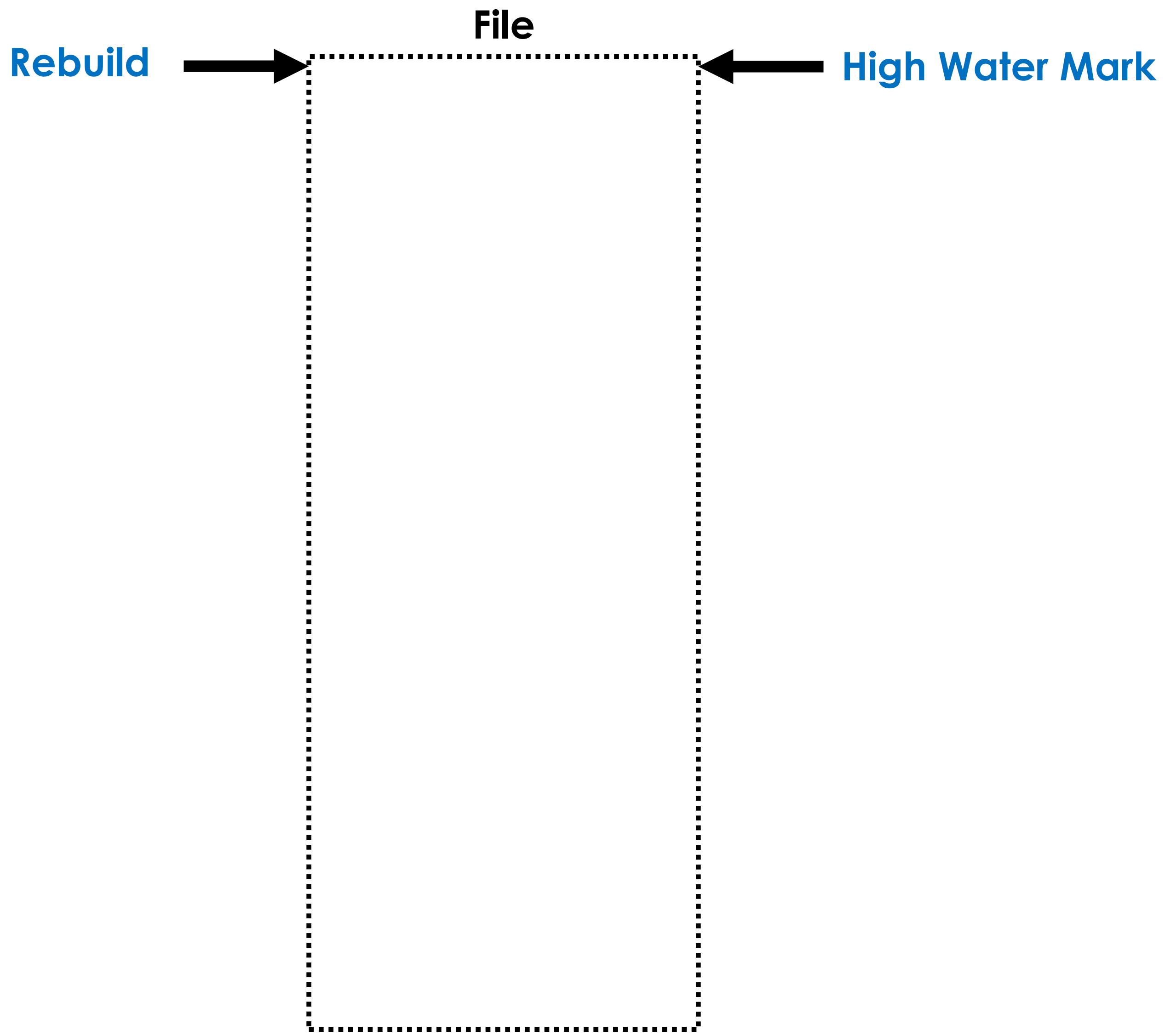

Sender
Data
Data
Receiver
Status

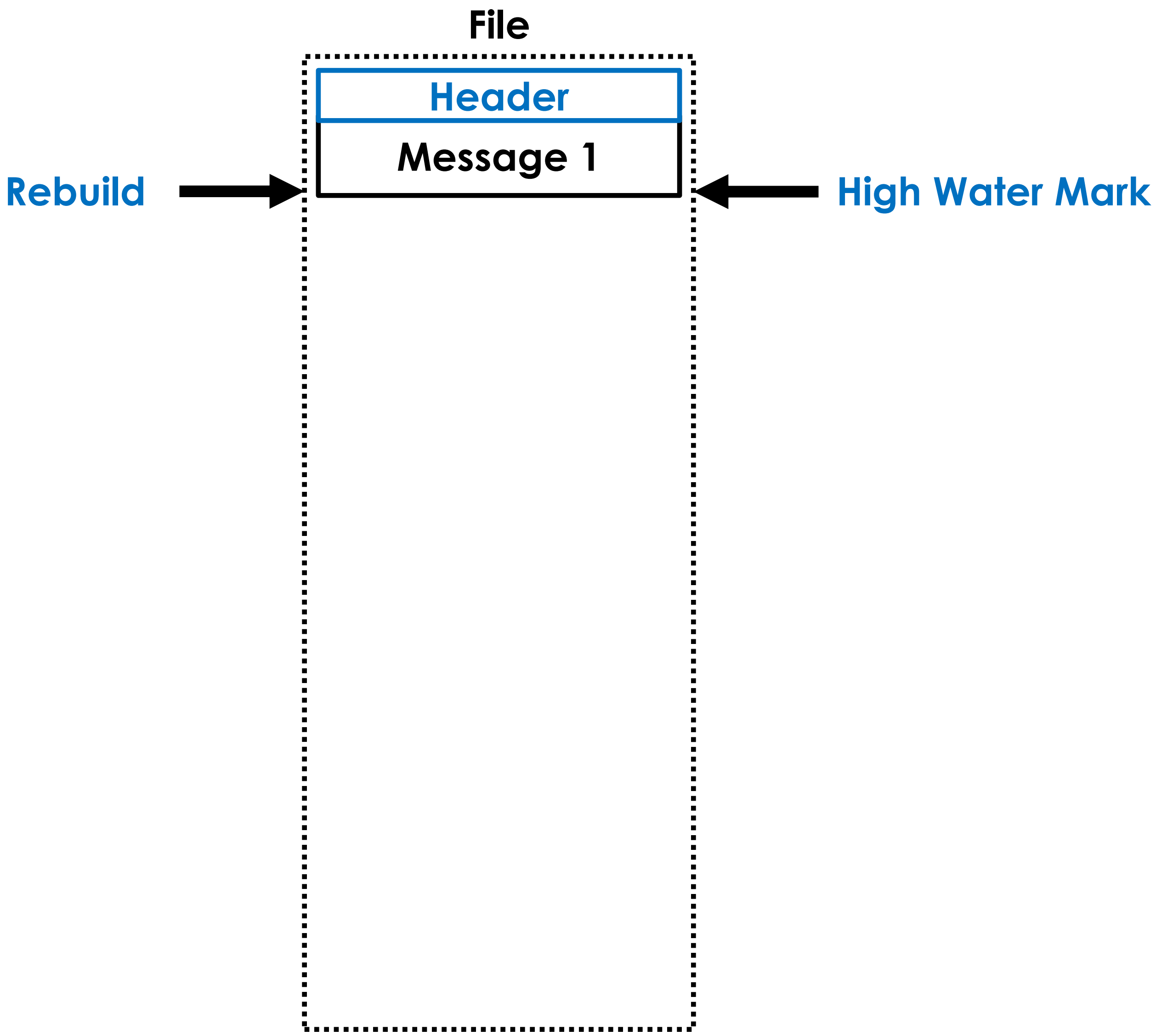

Sender
Heartbeat
Data
Data
Receiver

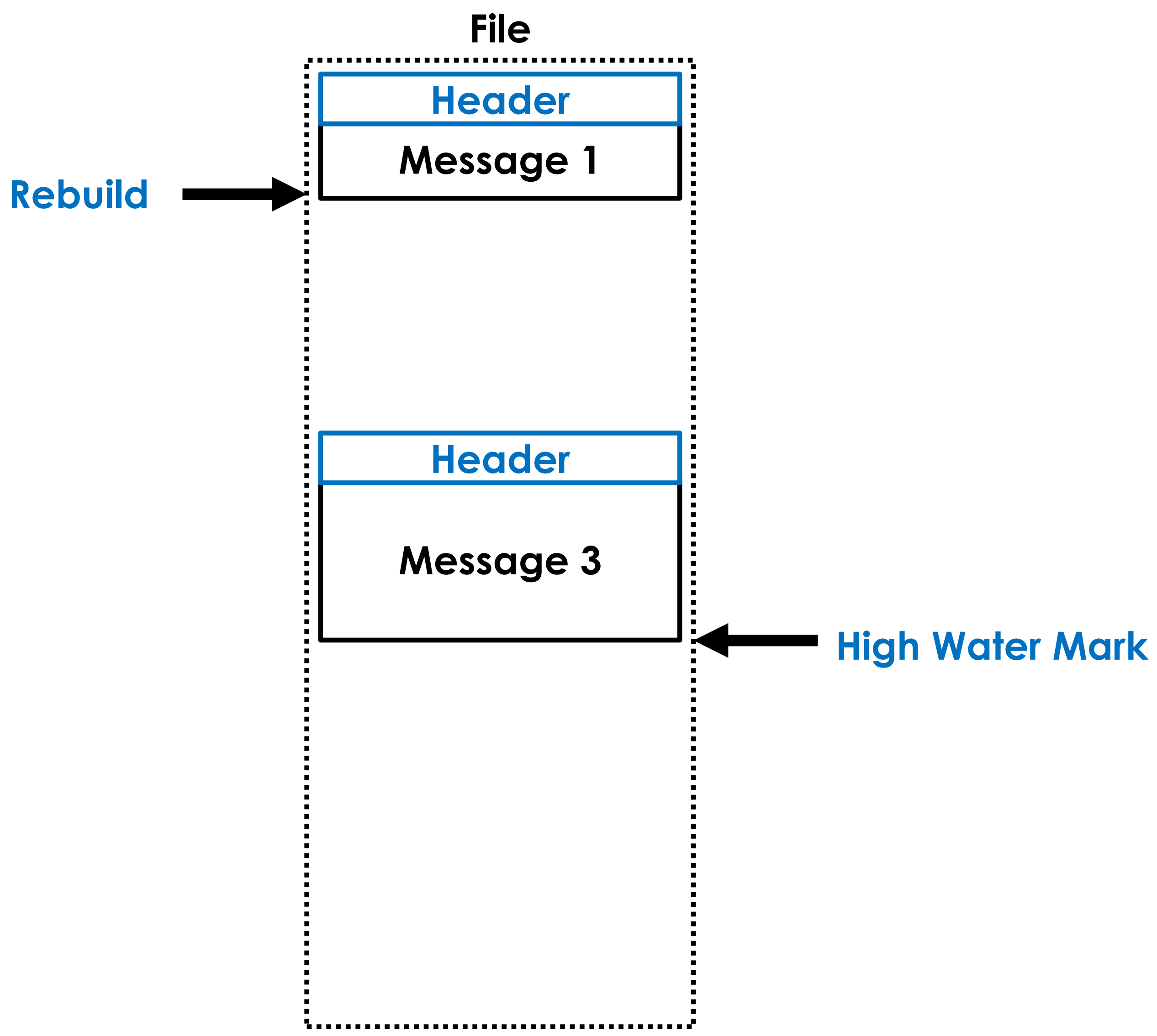

Sender
Data
Data
Receiver
NAK

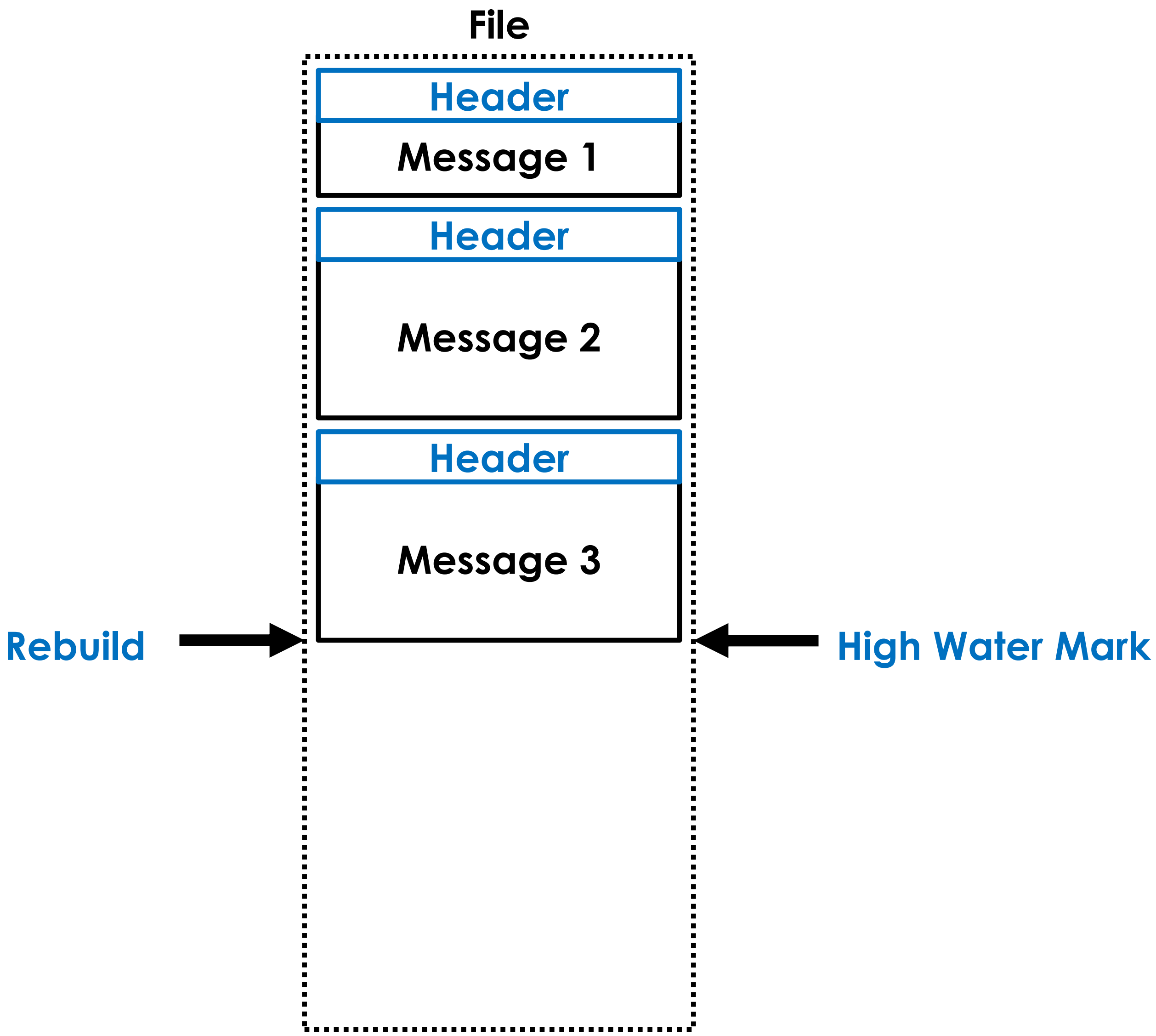

Sender
Data
Receiver

How to rebuild a message stream?


File
Rebuild
High Water Mark


File
Header
Message 1
Rebuild
High Water Mark


File
Header
Message 1
Rebuild
Header
Message 3
High Water Mark


File
Header
Message 1
Header
Message 2
Header
Message 3
Rebuild
High Water Mark

What if a gap is not filled?

How do we know what is consumed?

Publishers, Senders, Receivers, and Subscribers all keep position counters

Counters are the key to flow control and monitoring

Protocols can be more subtle than you think…

What about “Self similar behaviour”?

4. What did we learn on the way?

Monitoring and Debugging

Loss, throughput, and buffer size are all strongly related!!!

Pro Tip: Know your OS network parameters and how to tune them

Some parts of Java really suck!

Some parts of Java really suck!

Unsigned Types?

Some parts of Java really suck!

Unsigned Types?

NIO (most of) - Locks

Some parts of Java really suck!

Unsigned Types?

NIO (most of) - Locks

Off-heap, PAUSE, Signals, etc.

Some parts of Java really suck!

Unsigned Types?

NIO (most of) - Locks

Off-heap, PAUSE, Signals, etc.

String Encoding

Some parts of Java really suck!

Unsigned Types?

NIO (most of) - Locks

Off-heap, PAUSE, Signals, etc.

String Encoding

Managing External Resources

Some parts of Java really suck!

Unsigned Types?

NIO (most of) - Locks

Off-heap, PAUSE, Signals, etc.

String Encoding

Managing External Resources

Selectors - GC

Bytes!!!

```
public void main(final String[] args)
{
    byte a = 0b0000_0001;
    byte b = 0b0000_0010;

    byte flags = a | b;

    System.out.printf(
        "flags=%s\n",
        Integer.toBinaryString(flags));
}
```

Bytes!!!

```
public void main(final String[] args)
{
    byte a = 0b0000_0001;
    byte b = 0b0000_0010;

    byte flags = a | b;

    System.out.printf(
        "flags=%s\n",
        Integer.toBinaryString(flags));
}
```

Bytes!!!

```
public void main(final String[] args)
{
    byte a = 0b0000_0001;
    byte b = 0b0000_0010;

    byte flags = a | b;

    System.out.printf(
        "flags=%s\n",
        Integer.toBinaryString(flags));
}
```

Error:(8, 24) java: incompatible types: possible lossy conversion from int to byte

Some parts of Java are really nice!

Some parts of Java are really nice!

***Tooling** – IDEs, Gradle, HdrHistogram*

Some parts of Java are really nice!

***Tooling** – IDEs, Gradle, HdrHistogram*

Profiling – Flight Recorder

Some parts of Java are really nice!

***Tooling** – IDEs, Gradle, HdrHistogram*

Profiling – Flight Recorder

Bytecode Instrumentation

Some parts of Java are really nice!

***Tooling** – IDEs, Gradle, HdrHistogram*

Profiling – Flight Recorder

Bytecode Instrumentation

Unsafe!!! + Java 8

Some parts of Java are really nice!

***Tooling** – IDEs, Gradle, HdrHistogram*

Profiling – Flight Recorder

Bytecode Instrumentation

Unsafe!!! + Java 8

The Optimiser

Some parts of Java are really nice!

***Tooling** – IDEs, Gradle, HdrHistogram*

Profiling – Flight Recorder

Bytecode Instrumentation

Unsafe!!! + Java 8

The Optimiser – Love/Hate

Some parts of Java are really nice!

Tooling – IDEs, Gradle, HdrHistogram

Profiling – Flight Recorder

Bytecode Instrumentation

Unsafe!!! + Java 8

The Optimiser – Love/Hate

Garbage Collection!!!

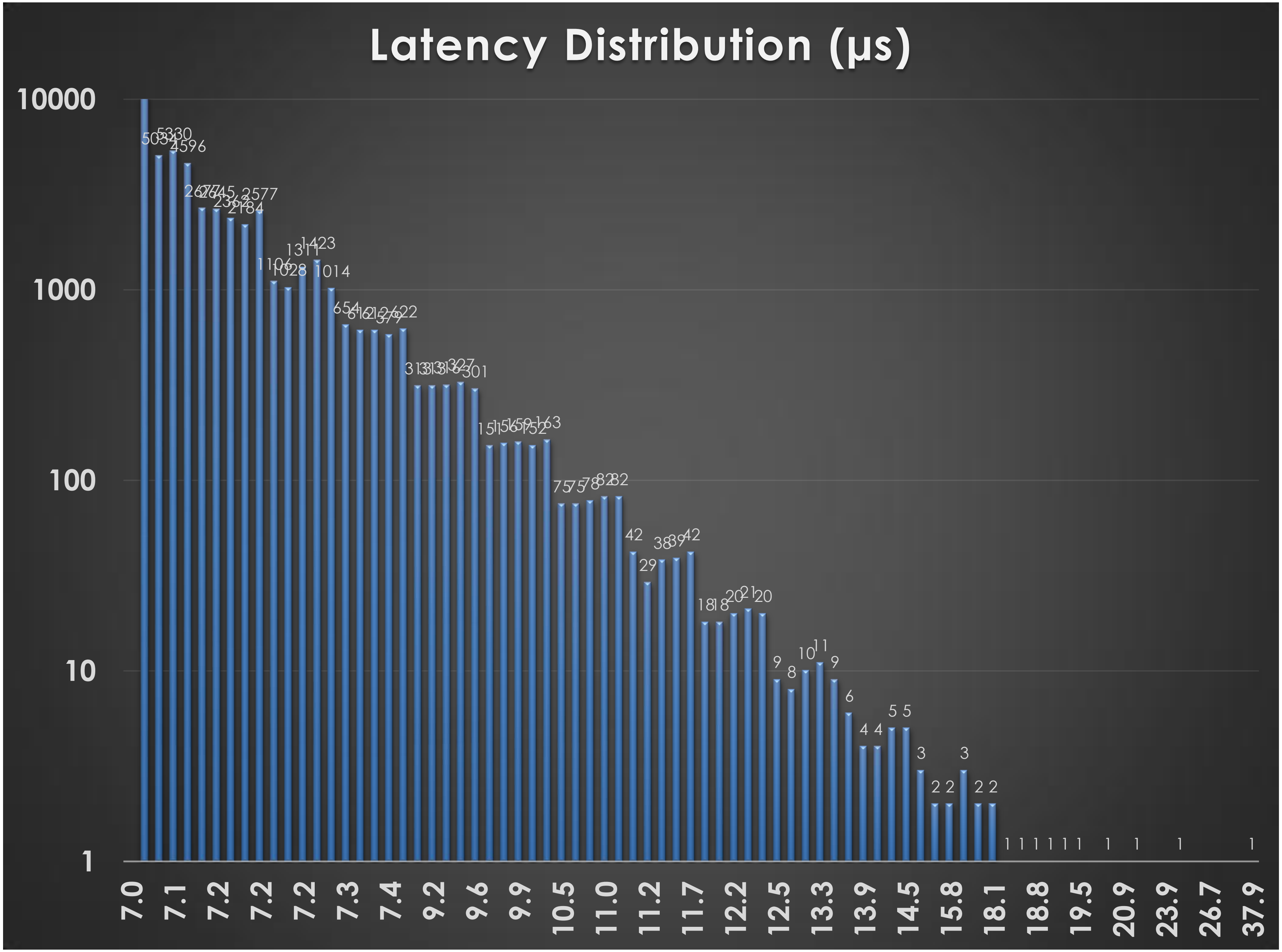
5. What's the Roadmap?

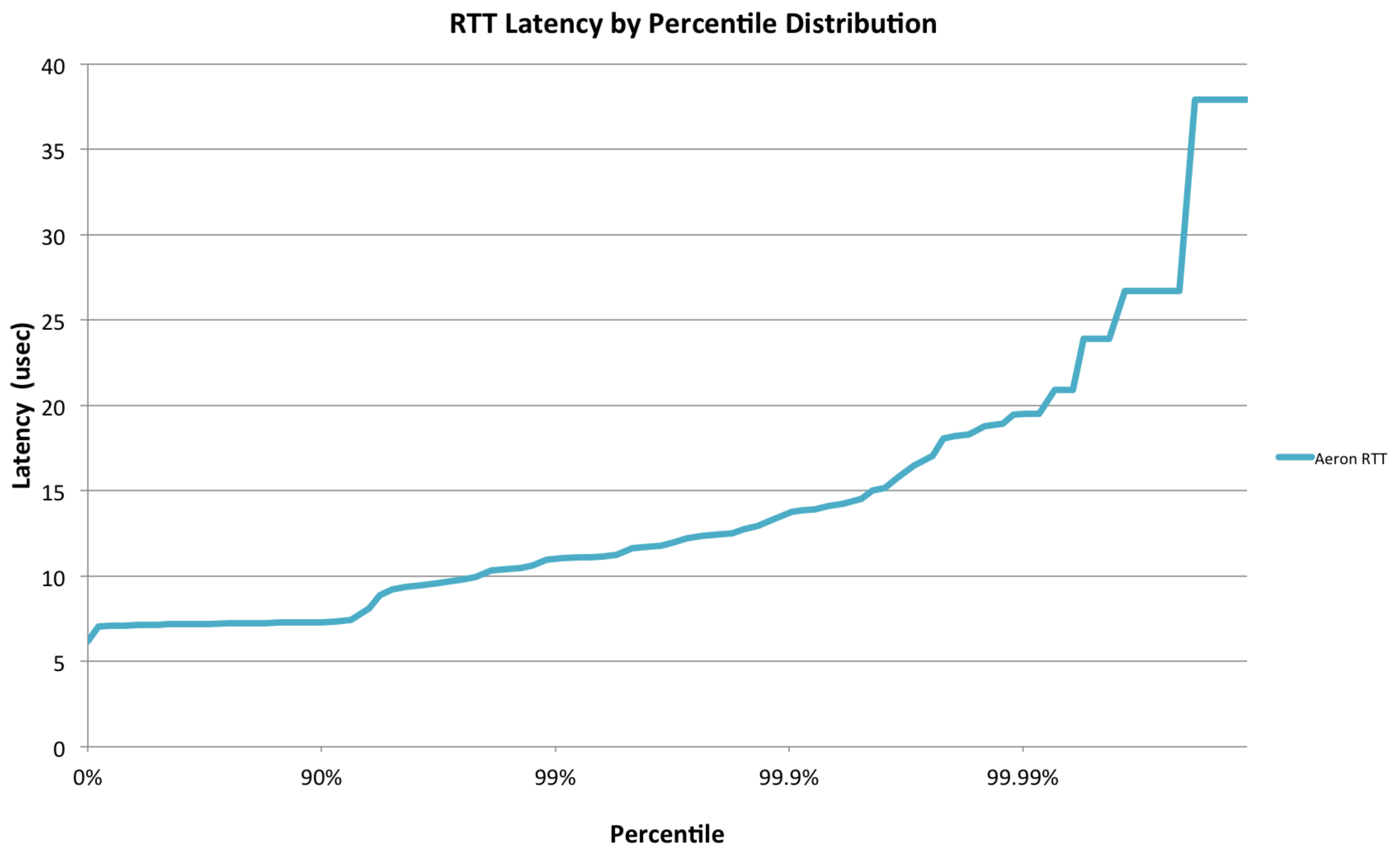
We’ve done a few passes of Profiling and Tuning

Things are looking **very** *good*

20+ Million 40 byte messages per second with Java client!

28+ Million 40 byte messages per second with C++ client!!!


Latency Distribution (μs)
10000
1000
100
10
1
5034
5330
4596
2677
2645
2362
2184
2577
1106
1028
1311
1423
1014
654
616
612
579
622
313
313
316
327
301
151
156
159
152
163
75
75
78
82
82
42
29
38
39
42
18
18
20
21
20
9
8
10
11
9
6
4
4
5
5
3
2
2
3
2
2
1
1
1
1
1
1
1
1
1
1
7.0
7.1
7.2
7.2
7.2
7.3
7.4
9.2
9.6
9.9
10.5
11.0
11.2
11.7
12.2
12.5
13.3
13.9
14.5
15.8
18.1
18.8
19.5
20.9
23.9
26.7
37.9


RTT Latency by Percentile Distribution
Latency (usec)
40
35
30
25
20
15
10
5
0
0%
90%
99%
99.9%
99.99%
Percentile
Aeron RTT

A number of clients have tried Aeron and are staggered by the latency improvements

Persistence

Stream Query

Replication

Queueing

Services

Aeron Core

Batch Send/Recv

Performance

IPC

Monitoring

Infiniband

Multi Unicast Send

In closing…

Do epic shit,
or die trying.

Where can I find it?

https://github.com/real-logic/Aeron

Questions?

Blog: http://mechanical-sympathy.blogspot.com/

Twitter: @mjpt777

"Any intelligent fool can make things bigger, more complex, and more violent.

It takes a touch of genius, and a lot of courage, to move in the opposite direction."

- Albert Einstein