# COS3043
# System Fundamentals

Lecture 11

# Topics

| | |
|---|---|
| 1. | Abstractions<br>1.1 Hardware Resources<br>1.2 OS Functionality<br>1.3 Managing the CPU and Memory |
| 2. | OS Structure<br>2.1 SPIN Approach<br>2.2 Exokernel Approach<br>2.3 L3/L4 Micro-Kernel Approach |
| 3. | Virtualization<br>3.1 Intro to Virtualization<br>3.2 Memory Virtualization<br>3.3 CPU and Device Virtualization |
| 4. | Parallelism<br>4.1 Shared Memory Machines<br>4.2 Synchronization<br>4.3 Communication<br>4.4 Scheduling |
| 5. | Distributed Systems<br>5.1 Definitions<br>5.2 Lamport Clocks<br>5.3 Latency Limit |

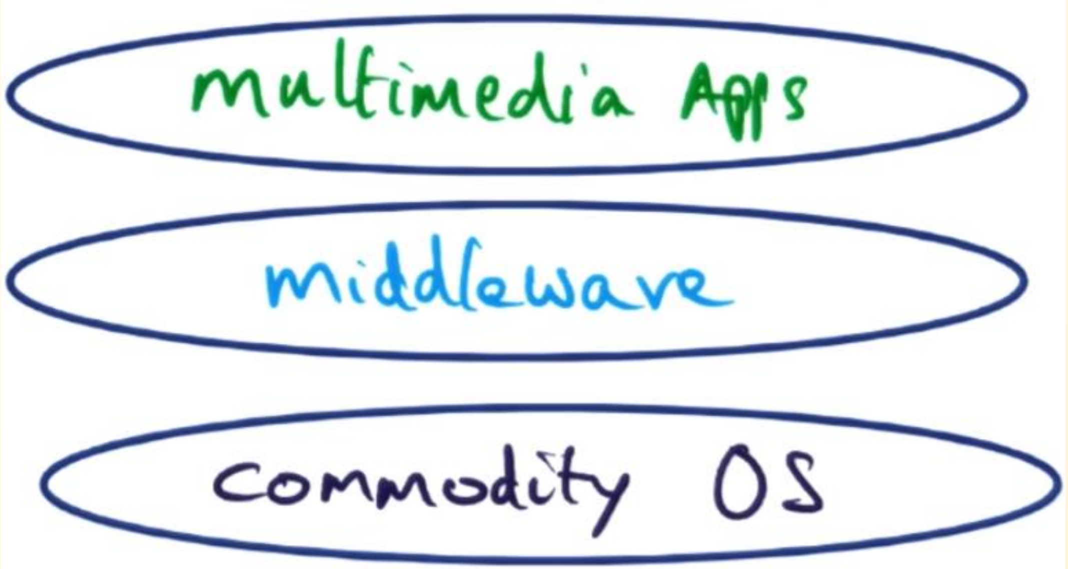
| | |
|---|---|
| 6. | Distributed Object Technology<br>6.1 Spring Operating System<br>6.2 Java RMI<br>6.3 Enterprise Java Beans |
| 7. | Design and Implementation of Distributed Services<br>7.1 Global Memory System<br>7.2 Distributed Shared Memory<br>7.3 Distributed File System |
| 8. | System Recovery<br>8.1 Lightweight Recoverable Virtual Memory<br>8.2 Rio Vista<br>8.3 Quicksilver |
| 9. | Internet Scale Computing<br>9.1 GiantScale Services<br>9.2 Content Delivery Networks<br>9.3 MapReduce |
| 10. | Real-Time and Multimedia<br>10.1 Persistent Temporal Streams |

# List of Discussion

- Persistent Temporal Streams

# Introduction

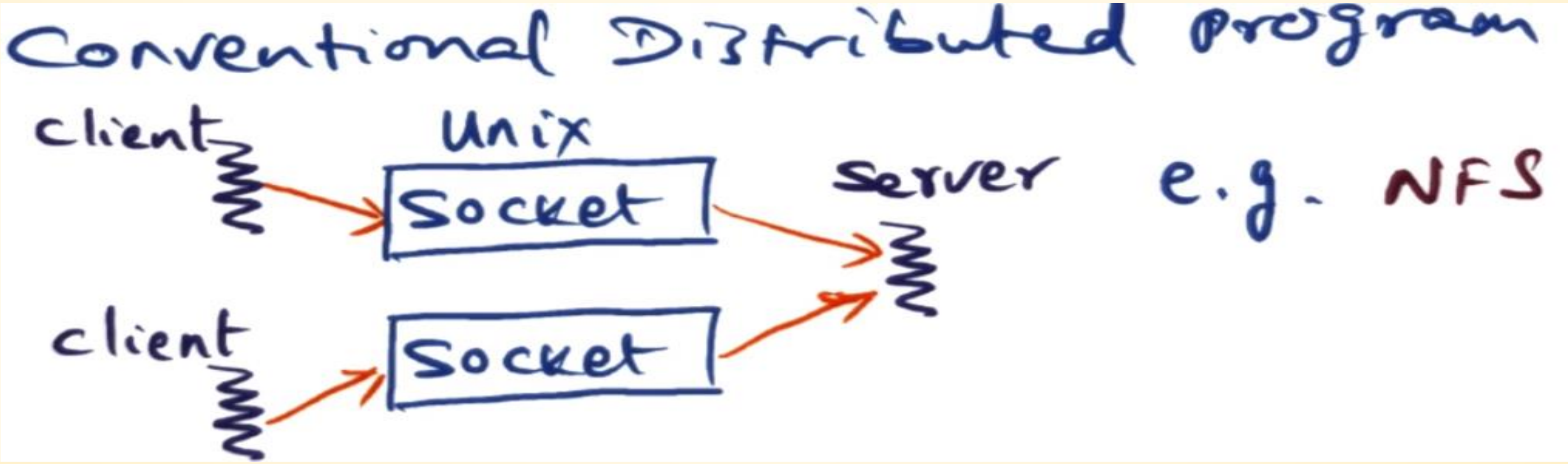
- So far, we haven’t talked about how OS handles real-time and multimedia application such as live streaming application.
- Persistent Temporal Streams (PTS) that supports a higher-level, domain-targeted programming abstraction for such applications.



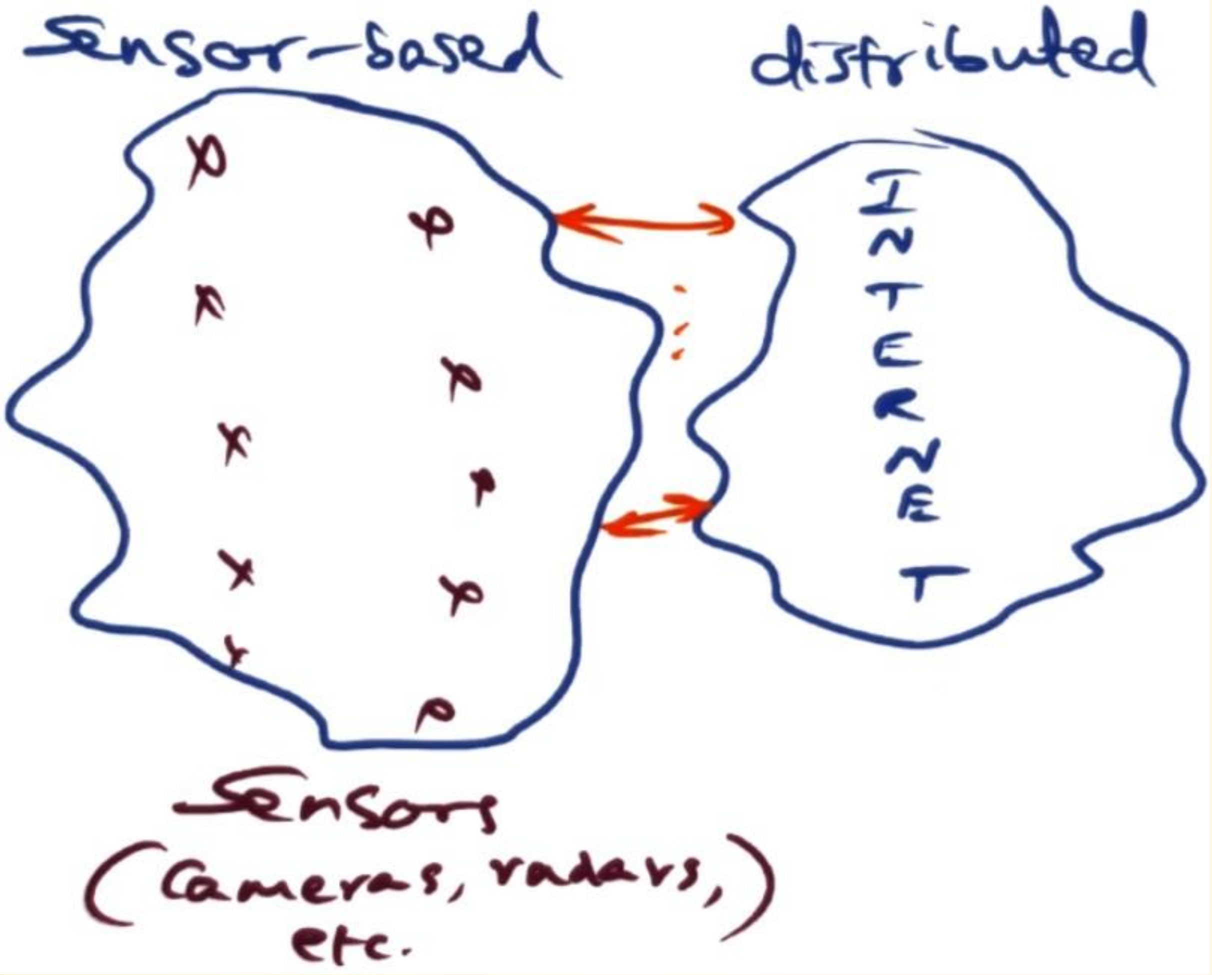
# Persistent Temporal Streams (PTS)

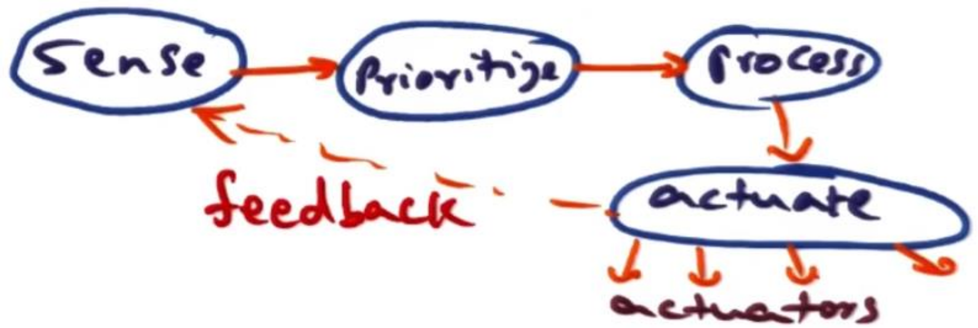
# Programming Paradigms

- Parallel Programs
  - Pthreads: API for parallel programs
- Distributed Programs
  - Sockets: API for distributed programs
  - But unfortunately, sockets API is too low level that is insufficient semantics for emerging multimedia applications.

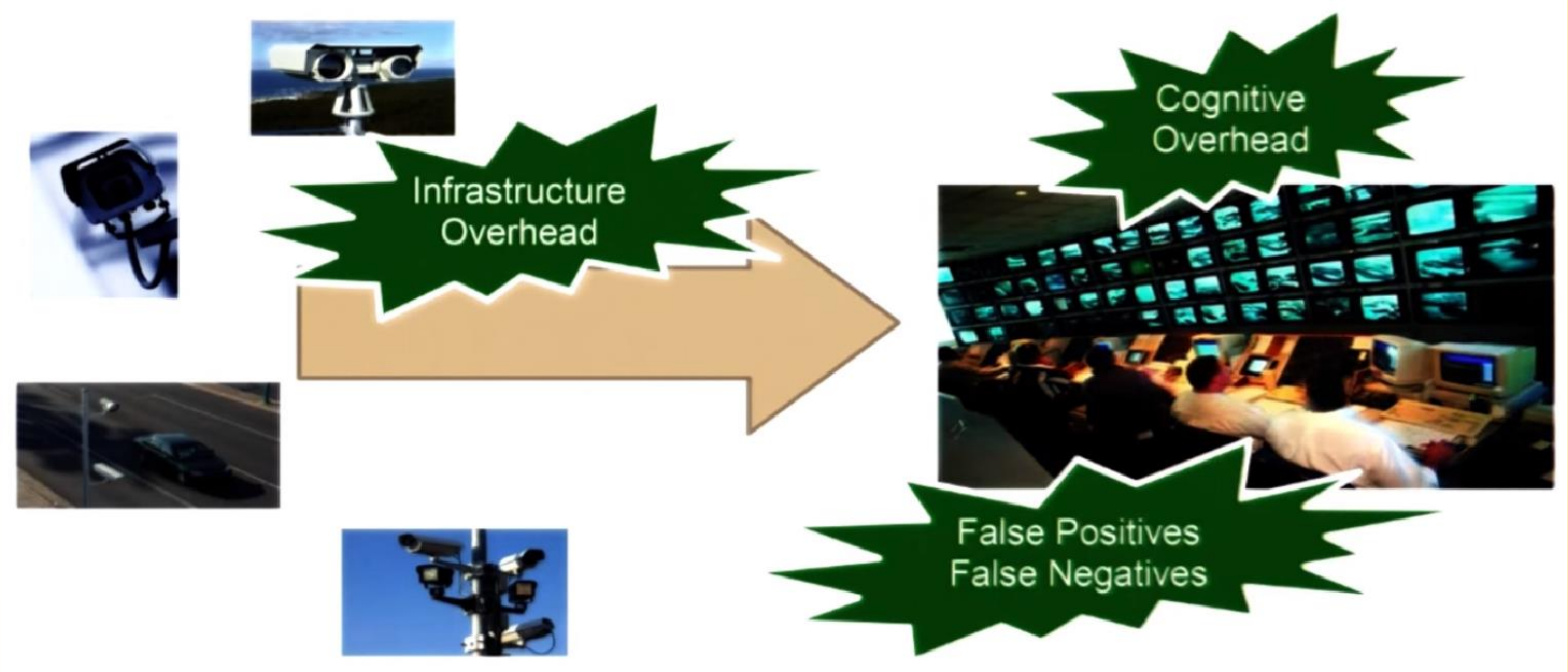


# Novel Distributed Multimedia Apps

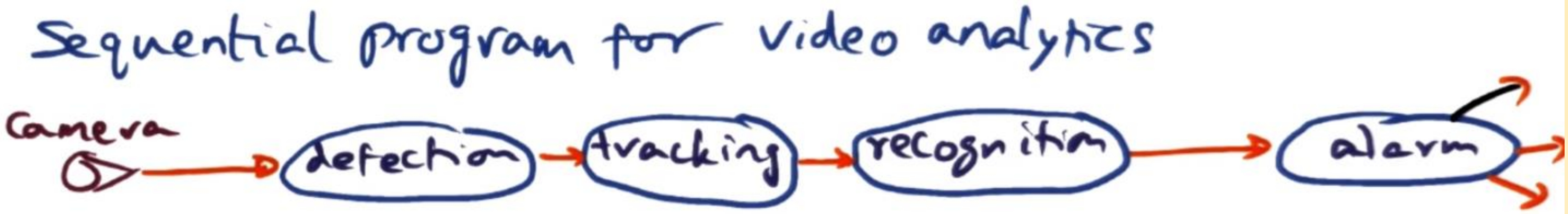




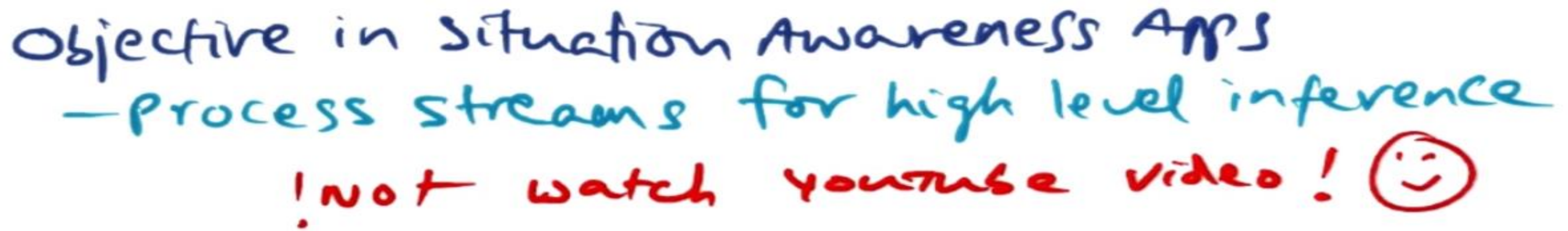
# Example: Large Scale Situation Awareness



# Programming Model for Situation Awareness



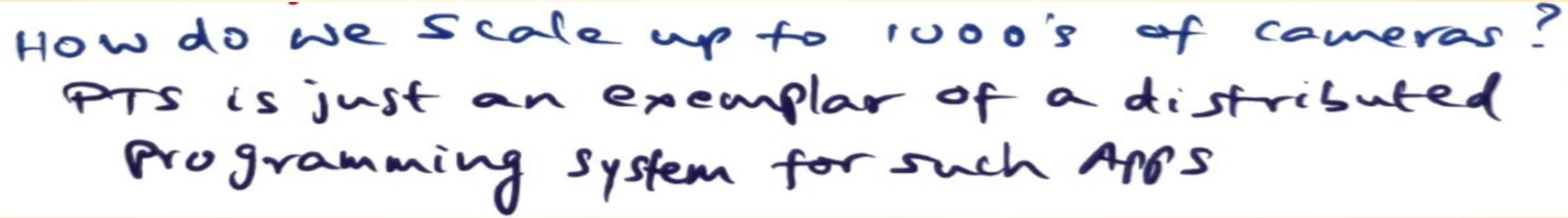
Sequential program for video analytics



Objective in Situation Awareness Apps

- process streams for high level inference

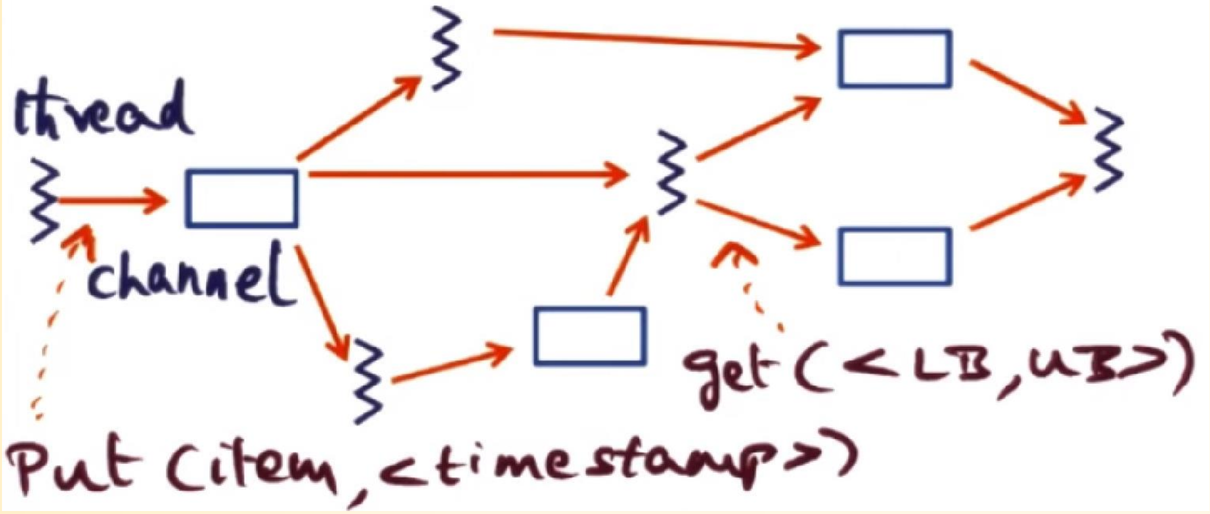
!not watch youtube video! ☺

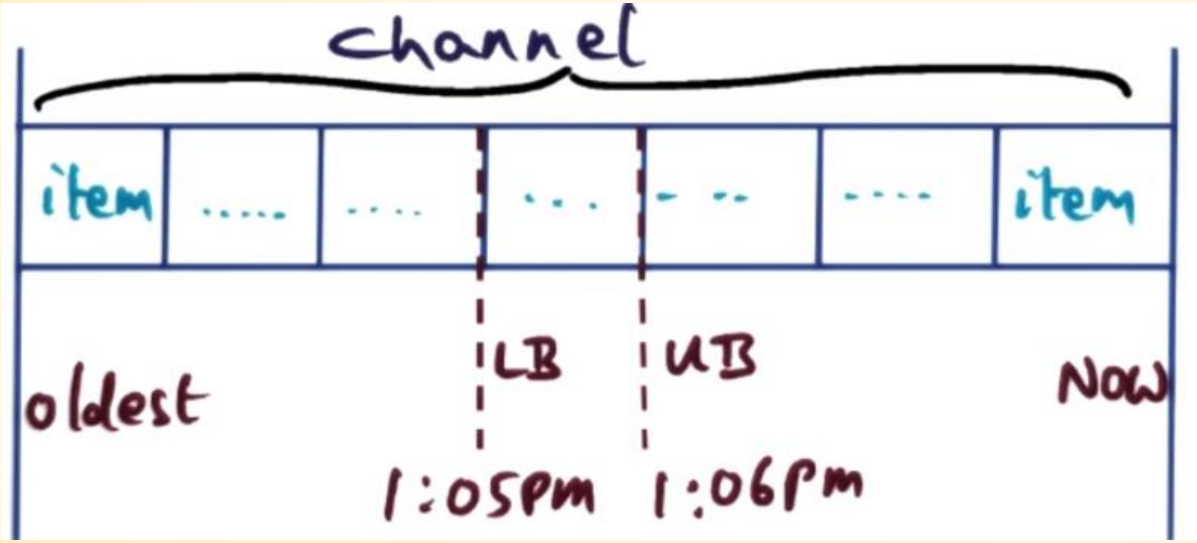


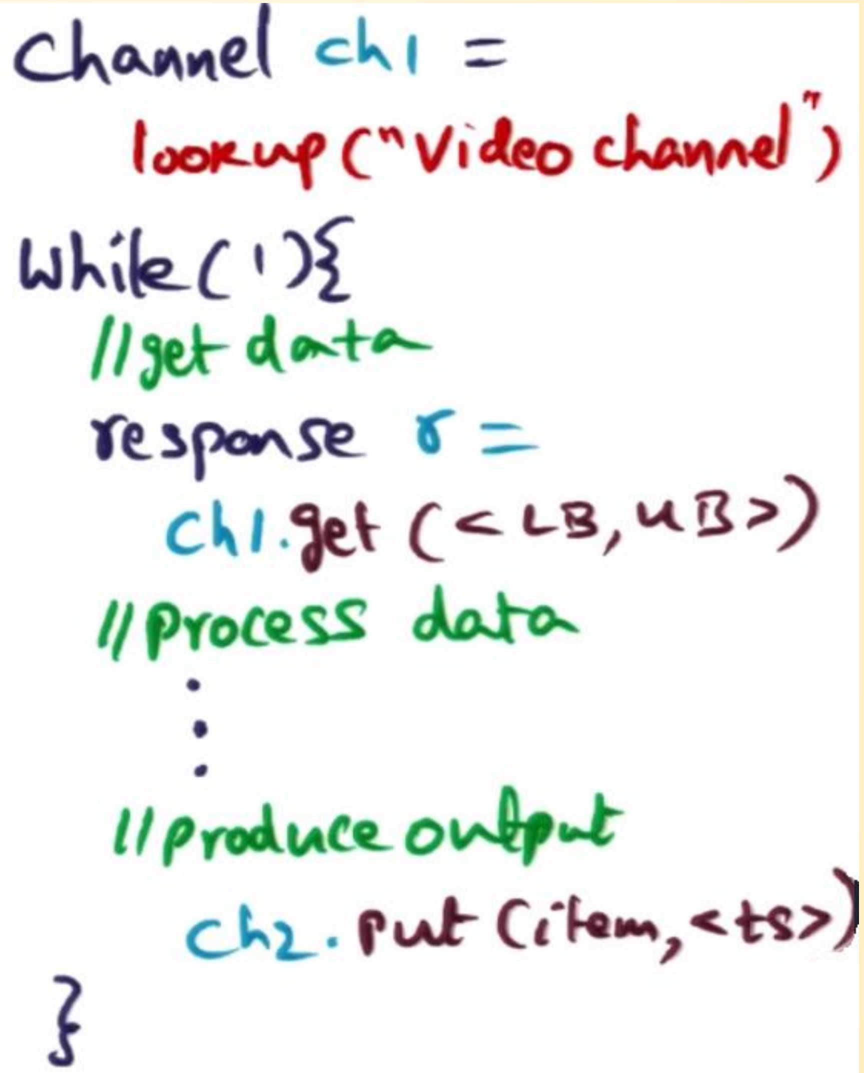
How do we scale up to 1000's of cameras?

PTS is just an exemplar of a distributed programming system for such Apps

# PTS Programming Model

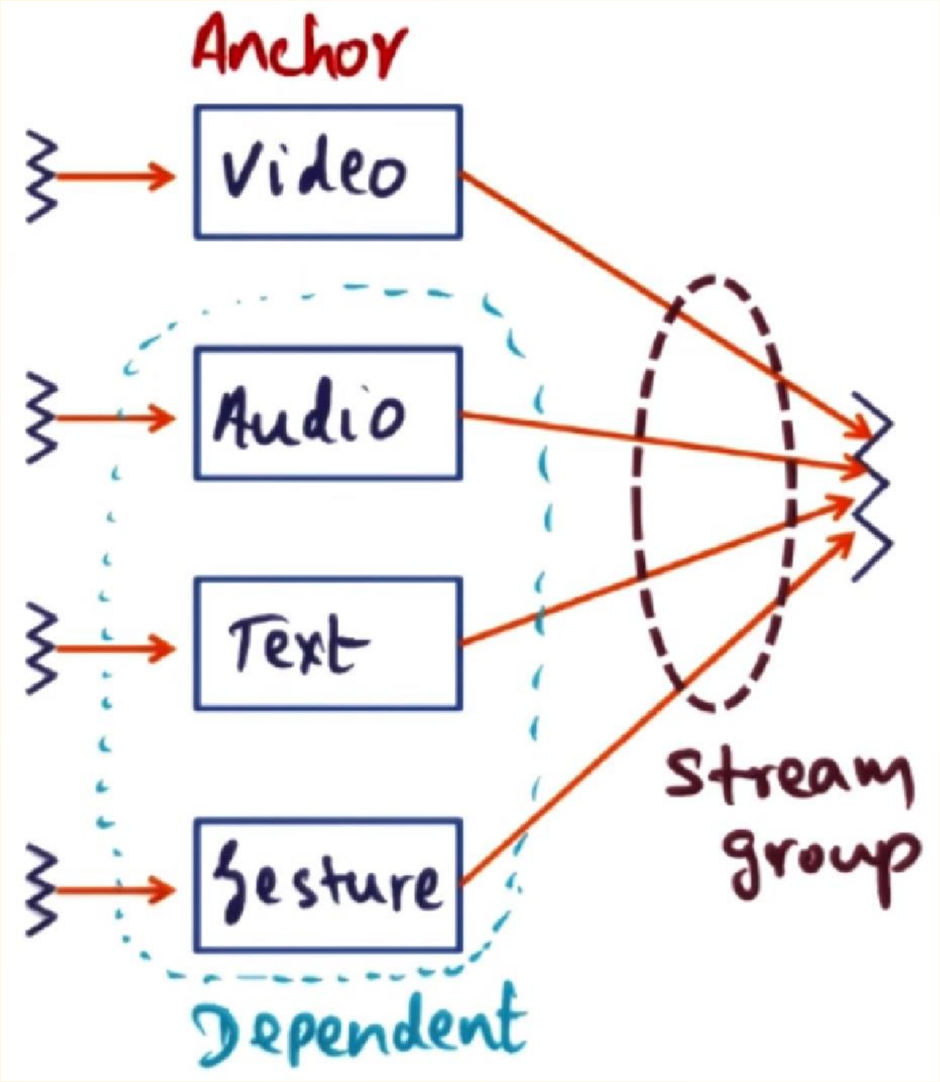


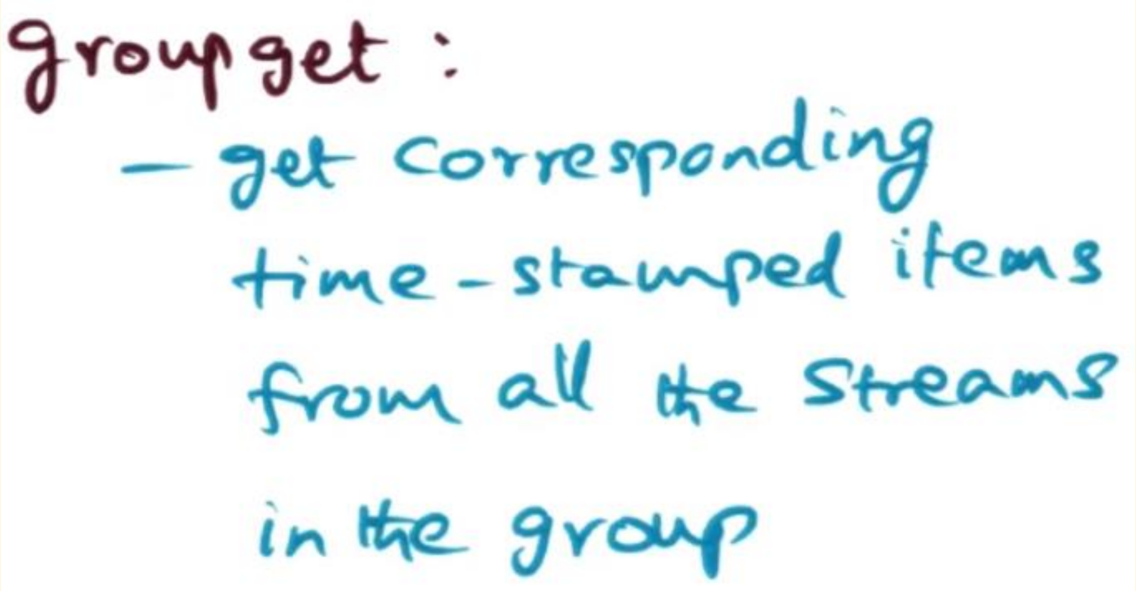




```
Channel ch1 =
    lookup("video channel")
While(1){
  //get data
  response r =
     Ch1.get(<LB,UB>)
  //process data
     :
  //produce output
     Ch2.put(item,<ts>)
}
```

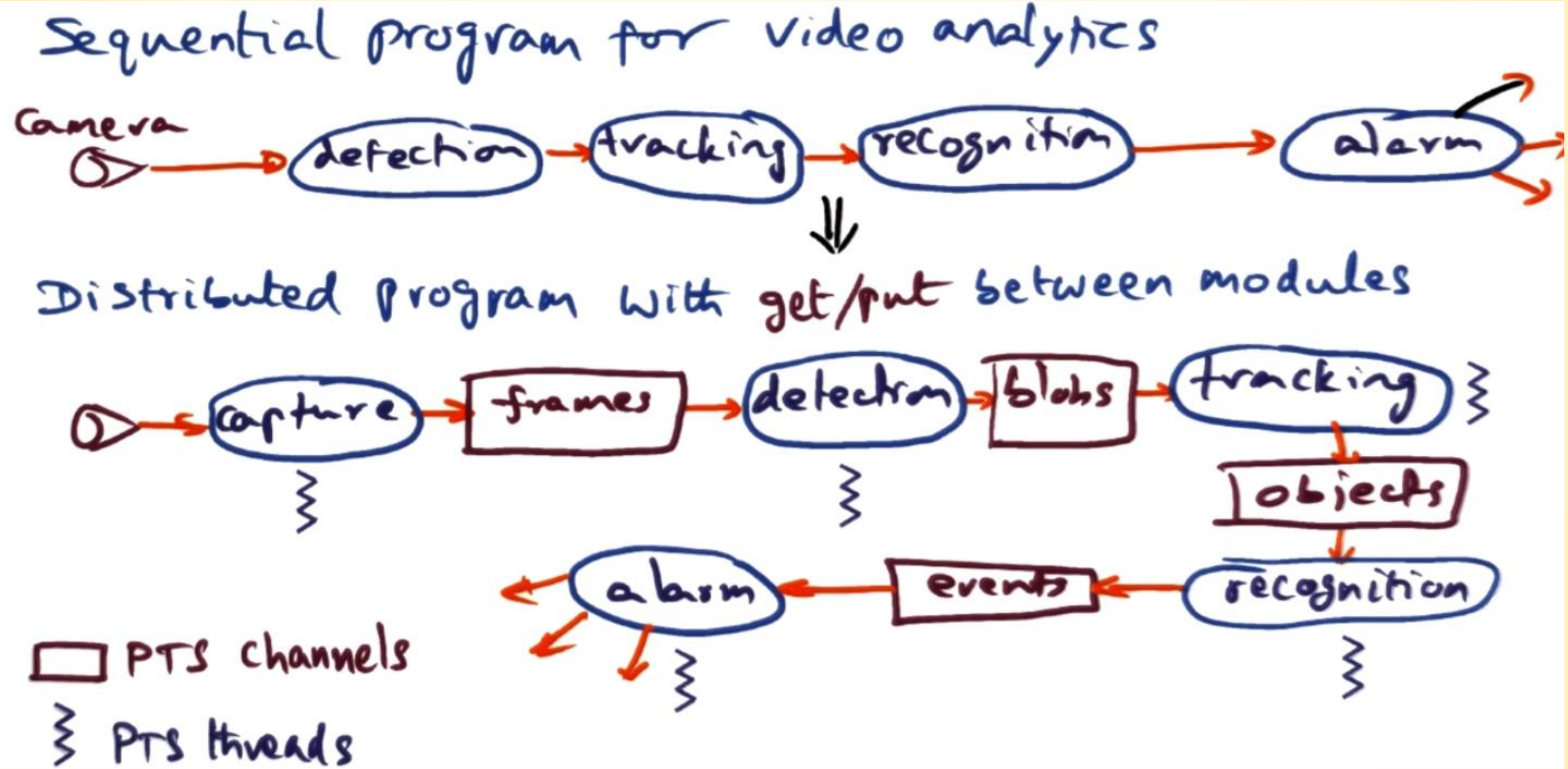
# Bundling Streams



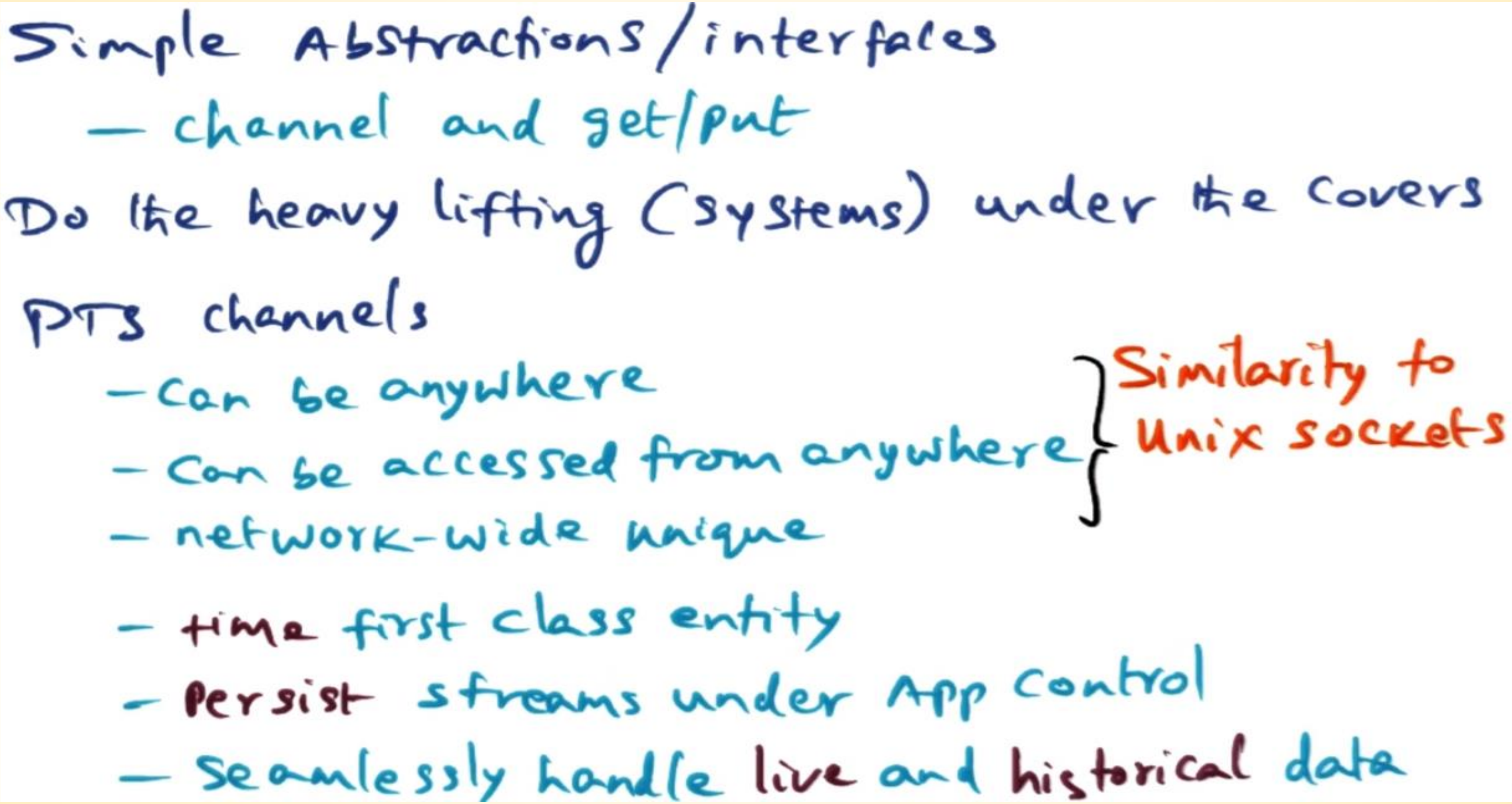

groupget :

- get corresponding time-stamped items from all the streams in the group

# Power of Simplicity



# PTS Design Principles



Simple Abstractions/interfaces

- channel and get/put

Do the heavy lifting (systems) under the covers

PTS channels

- Can be anywhere
- Can be accessed from anywhere

} Similarity to Unix sockets

- network-wide unique
- time first class entity
- Persist streams under App control
- Seamlessly handle live and historical data