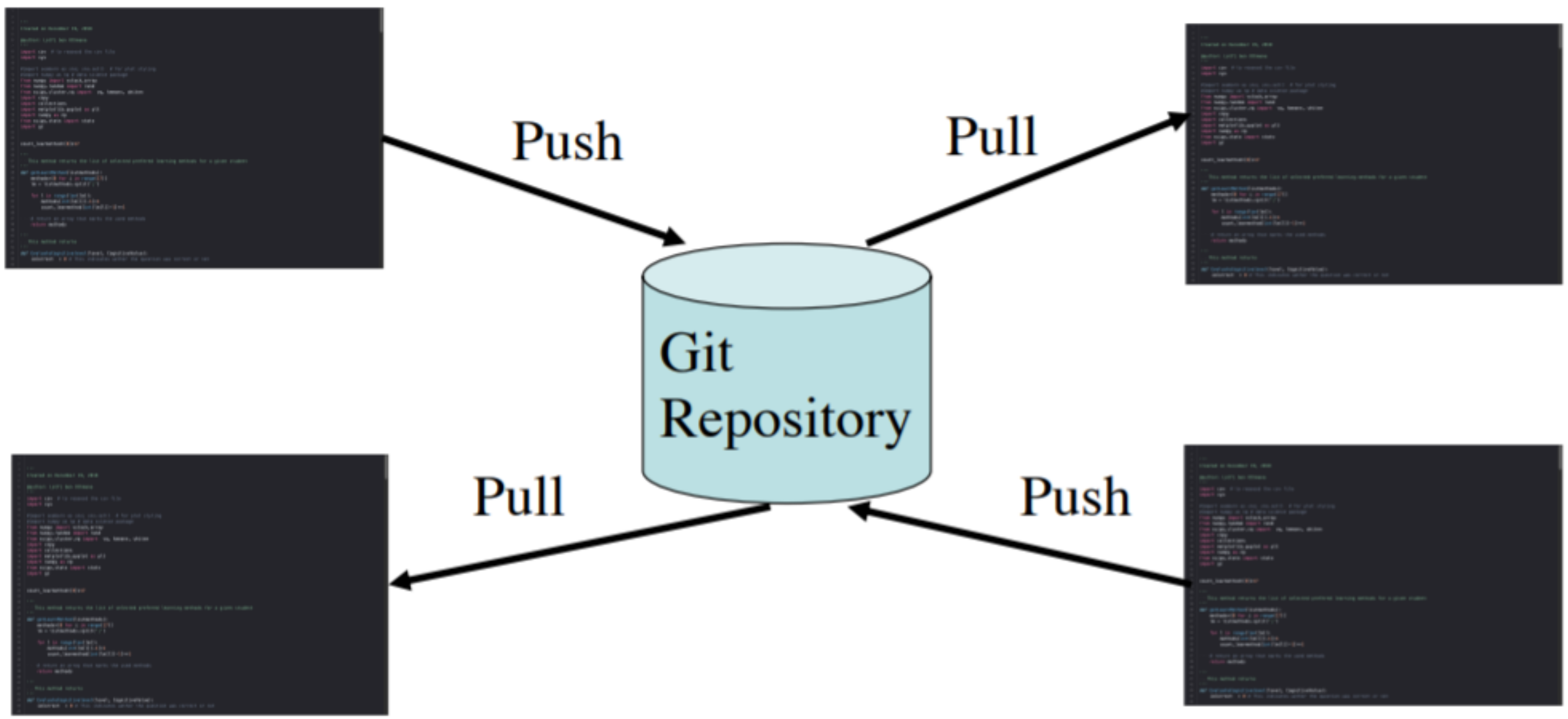
Concurrency Viewpoint

Lotfi ben Othmane

Client-Server Architecture



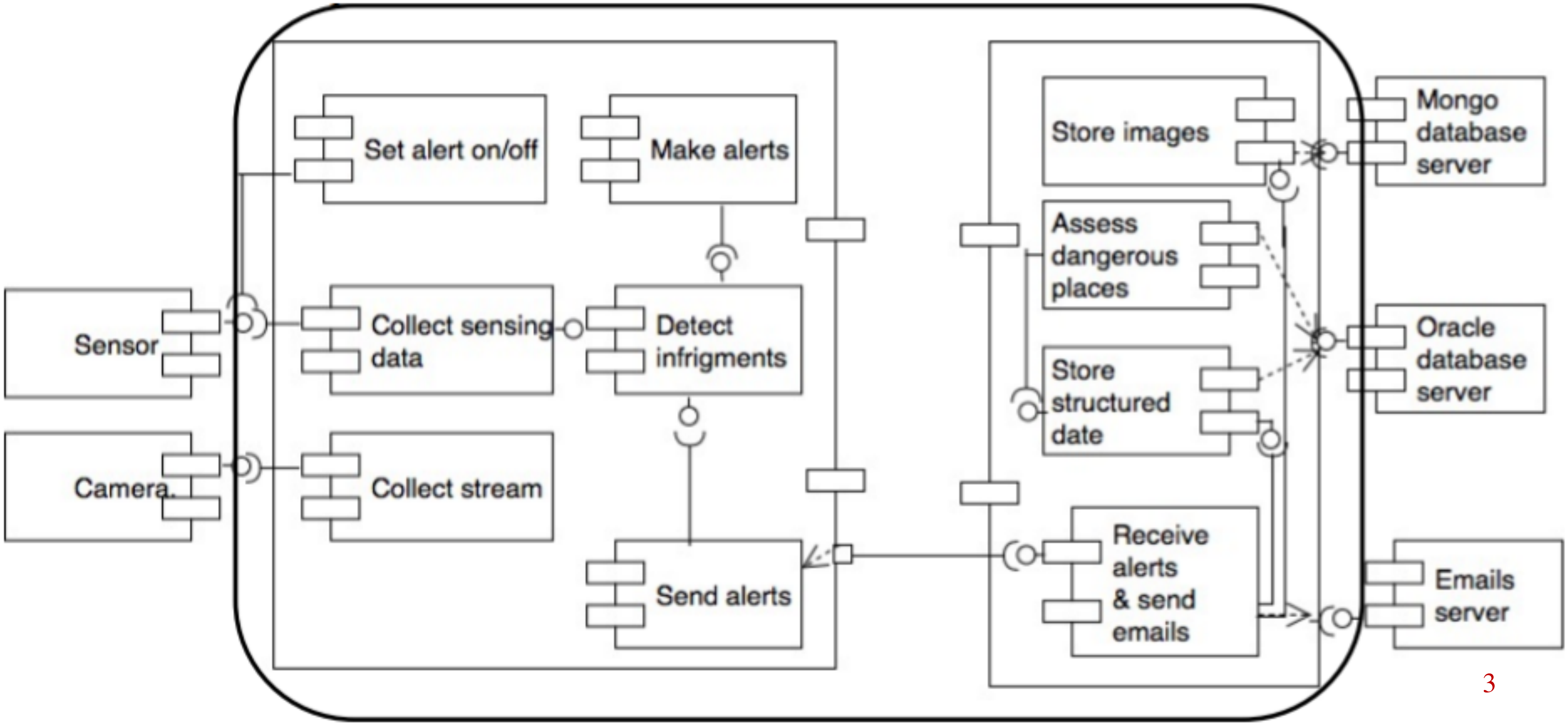
We deploy only one application server.
Each user can install the client on their device.

Motivating Example (1)

How many processes should the software run in at most?

How many processes should the software run in the least?

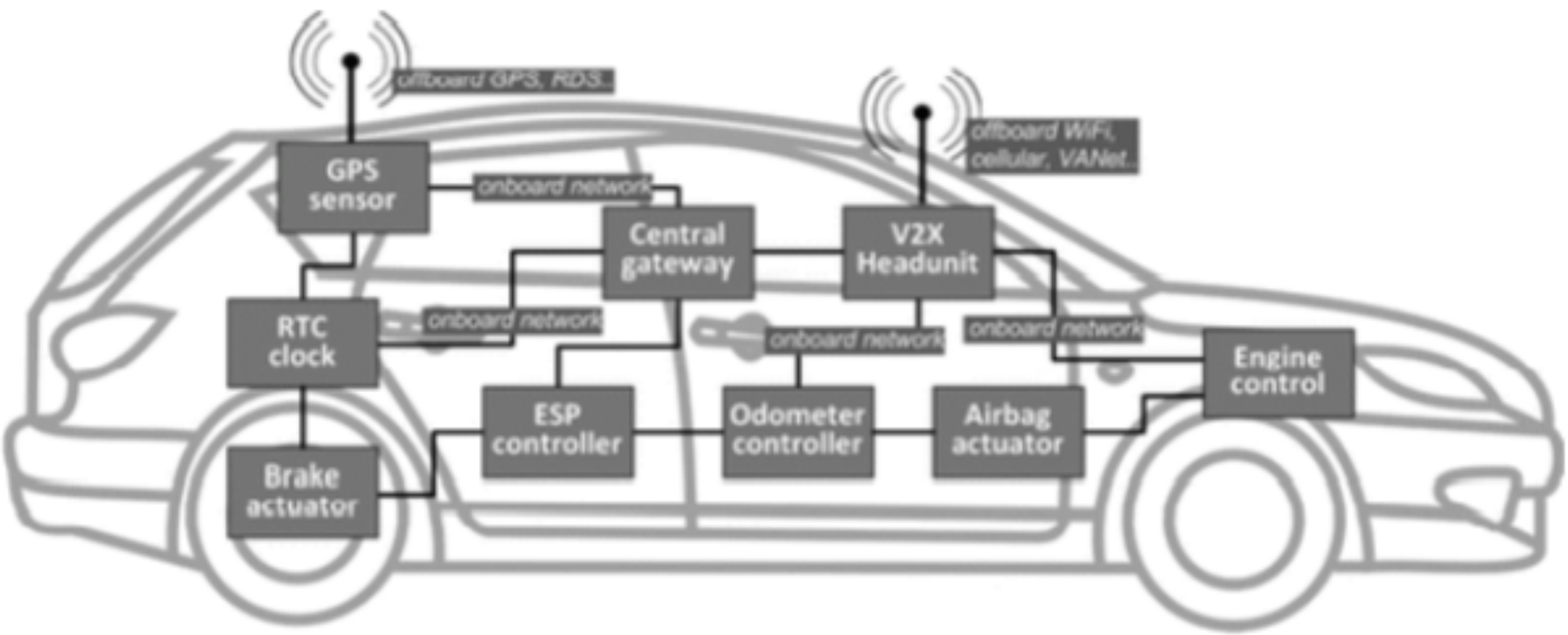
How many processes should the software run in?



Motivating Example (2)

How many processes run in the car?

How do they communicate?



Motivating Example (2)

What happen when there are data in the CAN bus but the program is busy processing previously received data?

```
import can

PID_REQUEST = 0x7DF
PID_RESPONSE = 0x7E8

bus = can.interface.BUS(channel='can0', bustype='socketcan_native')

Message = can.Message(arbitration_id=PID_REQUEST, data=[ID_FIRST_BYTE,
ID_SECOND_BYTE, pid, 0x00, 0x00, 0x00, 0x00, 0x00], extended_id=False)

bus.send(message)

message = bus.recv()
        if message.arbitration_id == PID_RESPONSE:
```

Overview

Concurrency viewpoints involve partitioning the system into components that execute at the same time and setting coordination and control mechanisms for these components.

Overview

Concurrency viewpoint includes:

Process model – the set of processes and threads and inter-process communication mechanisms

State model – the set of states and transitions for some functional elements

Concurrency vs Parallelism

Parallelism: operations are performed on independent or duplicated resources. The results may be merged.

Concurrency: operations are performed on shared resources considering a set of constraints.

Motivating Example

Use threading to solve the problem of data loss

```
import can

PID_REQUEST = 0x7DF
PID_RESPONSE = 0x7E8

bus = can.interface.BUS(channel='can0', bustype='socketcan_native')

Message = can.Message(arbitration_id=PID_REQUEST, data=[ID_FIRST_BYTE, ID_SECOND_BYTE, pid, 0x00, 0x00, 0x00, 0x00, 0x00], extended_id=False)

bus.send(message)

message = bus.recv()
        if message.arbitration_id == PID_RESPONSE:
```

Requirements Types Related to Concurrency

1. Availability – through redundancy
2. Modifiability – better flexibility to extend the system
3. Scalability – support heavy load
4. Security – rights to access resources

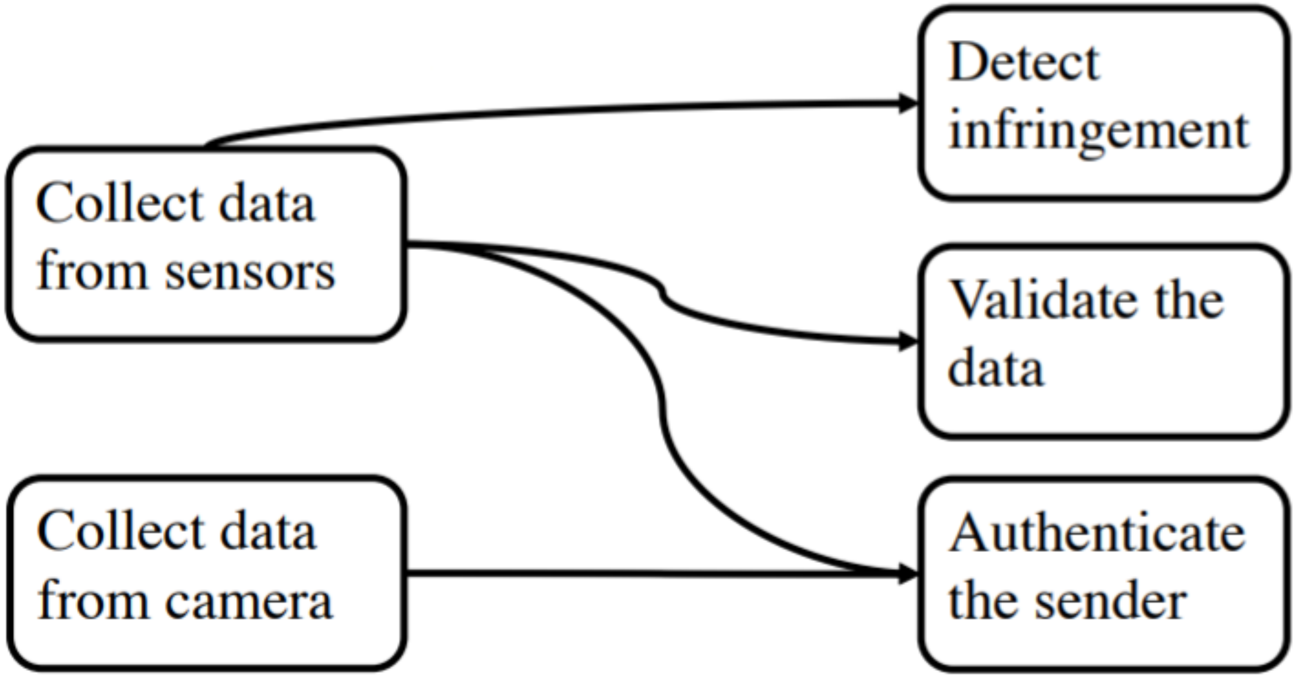
5. Performance – we will revisit this

Principle: Balance Computation and Communication

1. Should we have "validate the data" in a separate process?
2. Should we have "collect data from sensors" and "collect data from cameras" in the same process?

How do you partition the functionalities into processes?

What is the cost of communication?



Concerns 1 – Task Structure

Define the system process structure (set of processes) considering:

- Partitioning the workload into the processes
- Use operating system capabilities in terms of process grouping and threading

Concerns 1 – Task Structure

Concurrency elements:

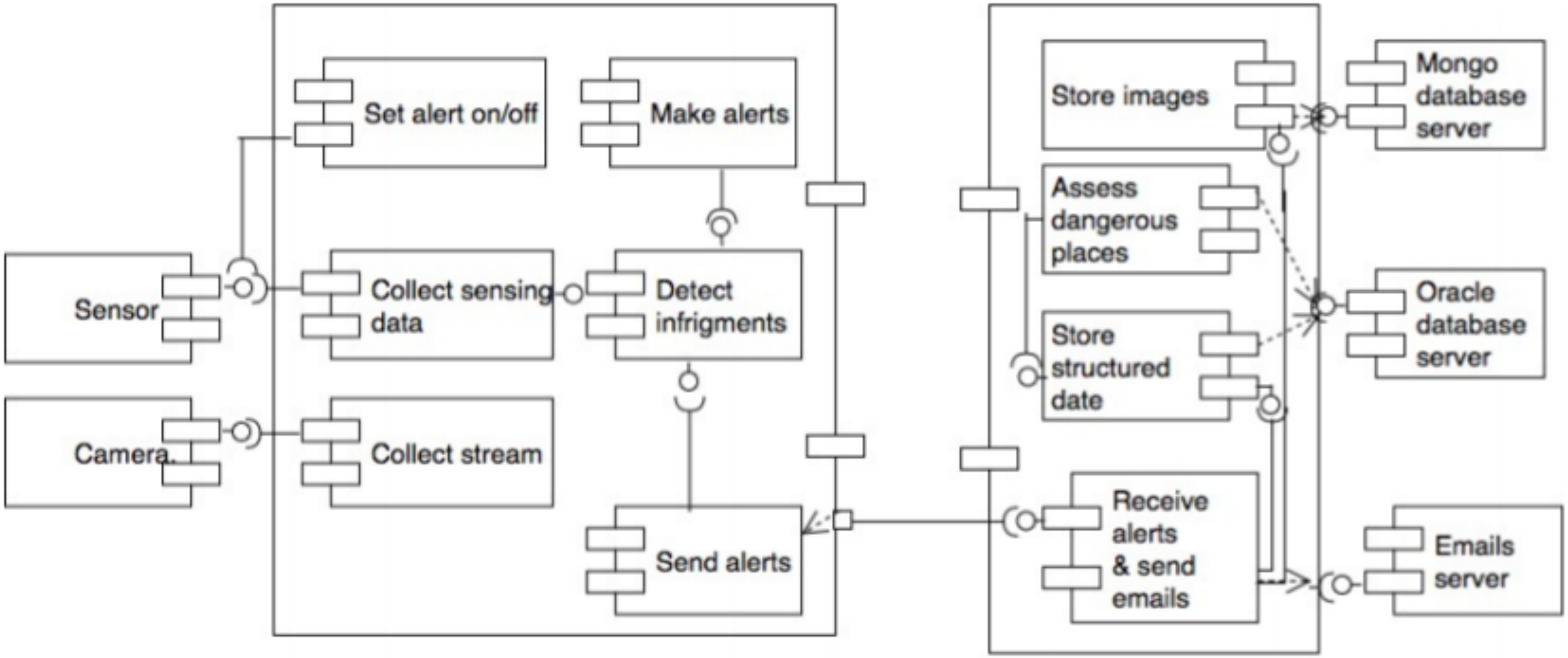
Process – independent isolated execution unit, each uses an OS execution environment.

Thread – execution unit that can be scheduled within processes.

Process group – a set of processes that address a concern.

Concerns 1 – Task Structure

How many processes do we need?



Concerns 2 – Map Functional Elements to Processes

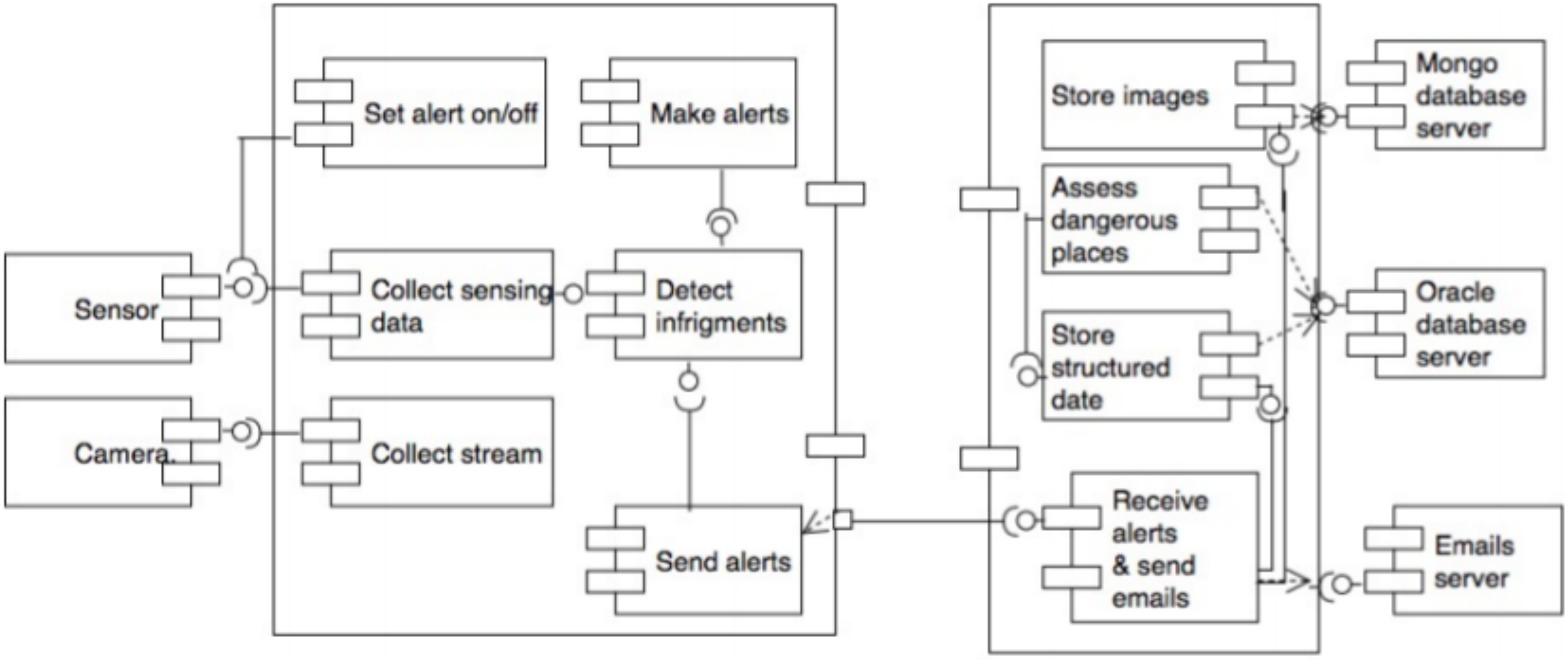
Goal: Map functional elements to tasks

Questions:

1. Which functional elements need to be isolated from each other?
2. Which functional elements need to cooperate closely?

Concern 2 – Map Functional Elements to Processes

- Should we make detect infringements, send alerts, and make alerts in one process or separate processes?
- What about collect sensing data and collect camera stream?



Concern 3 – Inter-process Communication

1. Message passing – send and receive messages through the network
2. Remote procedure calls – a process calls a procedure on another process
3. Application Programming Interface (API), e.g. web services and microservices
4. Data-sharing mechanisms – shared memory, files, and databases
5. Coordination mechanisms – e.g. semaphore and mutex
6. Etc.

Concern 3 – Inter-process Communication

What mechanism is used to communicate with the car?

```
import can

PID_REQUEST = 0x7DF
PID_RESPONSE = 0x7E8

bus = can.interface.BUS(channel='can0', bustype='socketcan_native')

Message = can.Message(arbitration_id=PID_REQUEST, data=[ID_FIRST_BYTE, ID_SECOND_BYTE, pid, 0x00, 0x00, 0x00, 0x00, 0x00], extended_id=False)

bus.send(message)

message = bus.recv()
        if message.arbitration_id == PID_RESPONSE:
```

Concern 3 – Inter-process Communication

```java
import java.rmi.*;
import java.rmi.server.*;
public class AdderRemote extends UnicastRemoteObject implements Adder{
AdderRemote()throws RemoteException{
super();
}
public int add(int x,int y){return x+y;}
}
```

```java
import java.rmi.*;
public class MyClient{
public static void main(String args[]){
try{
Adder stub=(Adder)Naming.lookup("rmi://localhost:5000/sonoo");
System.out.println(stub.add(34,4));
}catch(Exception e){}
}
}
```

Concern 3 – Inter-process Communication

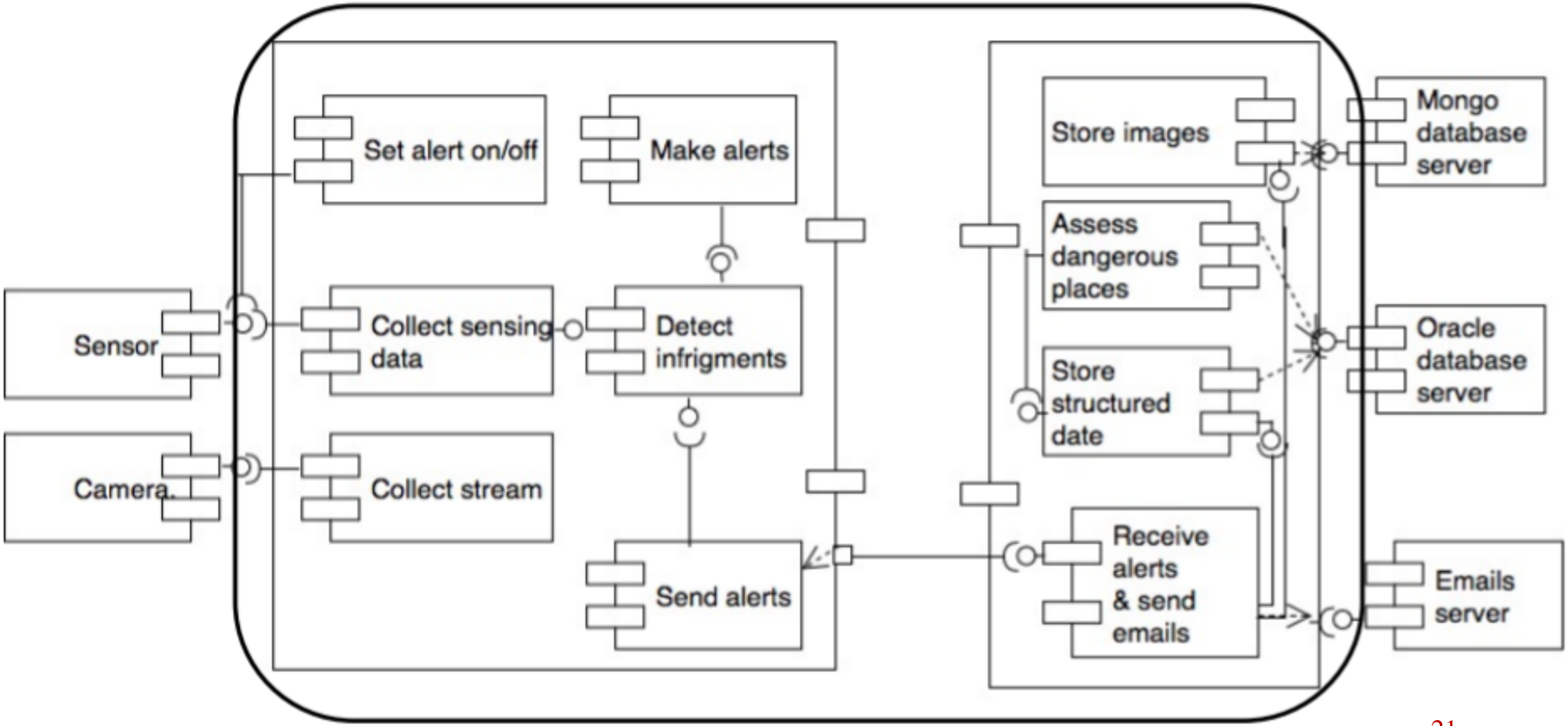
Each inter-process communication mechanism has advantages and disadvantages

Criteria:

1. Synchronous vs asynchronous communication
2. Type of data (Jason, xml)
3. Persistence
4. Location of the communicating process (same node?)
5. Performance – use of transient object provides better performance than use of persistent object
6. Reliability – loss of messages, failure of process
7. Etc.

Concern 3 – Inter-process Communication (IPC)

What mechanism should you use for sensor data collection communication?



Concern 4 – State Management

The runtime state of some system elements is important to correct operation of the system.

- In event-driven systems

The elements would have

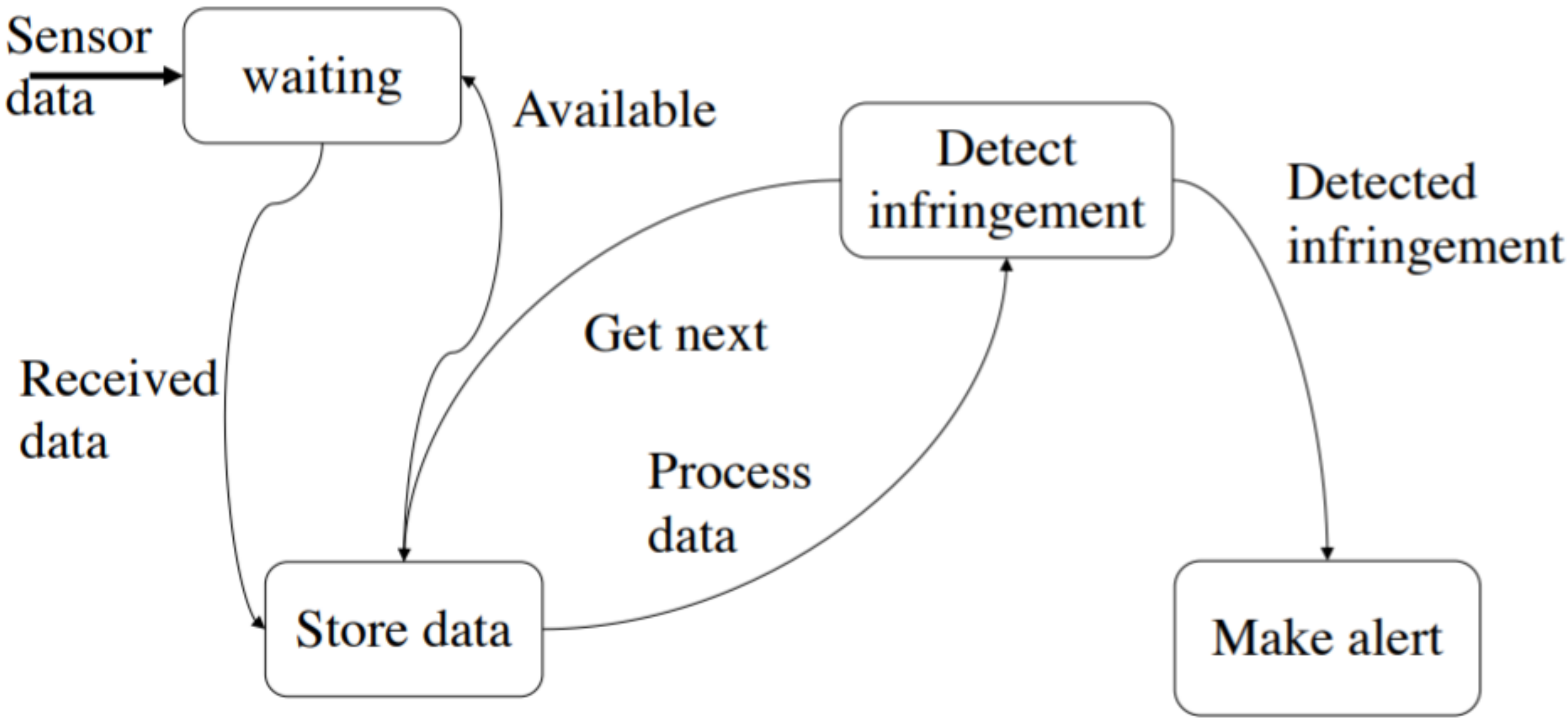
- A set of states
- A set of transitions between the states

Concern 4 – State Management

Would it be ok that two nodes send on the bus at the same time?

```
import can

PID_REQUEST = 0x7DF
PID_RESPONSE = 0x7E8

bus = can.interface.BUS(channel=‘can0’, bustype=‘socketcan_native’)

Message = can.Message(arbitration_id=PID_REQUEST, data=[ID_FIRST_BYTE, ID_SECOND_BYTE, pid, 0x00, 0x00, 0x00, 0x00, 0x00], extended_id=False)

bus.send(message)

message = bus.recv()
        if message.arbitration_id == PID_RESPONSE:
```

Concern 4 – State Management



Concern 5 – Synchronization and Integrity

- Data shared among a set of threads/processes must be coherent and not corrupt.
- Concurrency mechanisms need to coordinate concurrent activities on shared resources.

Concern 6 – Support for Scalability

Concurrency allows to scale systems to address heavy load.

Little concurrency or naïve synchronization mechanisms negatively impact the ability to scale the system

- Functionalities that can run in parallel
- Constraint: the shared resources

Concern 7 – Startup and Shutdown

- Processes may have dependencies.
- Starting the system set of processes should consider these dependencies
- Shutting down the system needs to consider the processes' dependencies – e.g. ensure no data is lost

Concern 8 – Task Failure

- Some of the system's processes could fail, e.g. due to hardware failure or exception.
- Concurrency design should include system-wide monitoring and recovery in case of failure.

Concern 9 - Reentrancy

- Component A needs to correctly operate though it is used by a set of concurrent components B_i.

- E.g. the data analytics component needs to be reentrant to allow data storage queries.

Concurrency Model

Concurrency view maps functional elements to runtime execution entities via concurrency model.

Concurrency model includes:

1. Processes
2. Process groups
3. Threads
4. Inter-process communication

Example - Representation of Concurrency

Activities for Concurrency Model

1. Map elements to processes – define the number of needed processes and assign functional elements to processes.
2. Determine threading design – number of threads for each process.
3. Determine mechanisms for resource sharing – identify shared resources and protocols to use them.
4. Determine the IPC (inter-process communication) mechanisms – decide on IPC for each communicating process.
5. Analyze contention – checks for processes that require shared resources concurrently.
6. Analyze deadlocks – check if Process A waits for a resource used by process B and B waits for A for another resource.

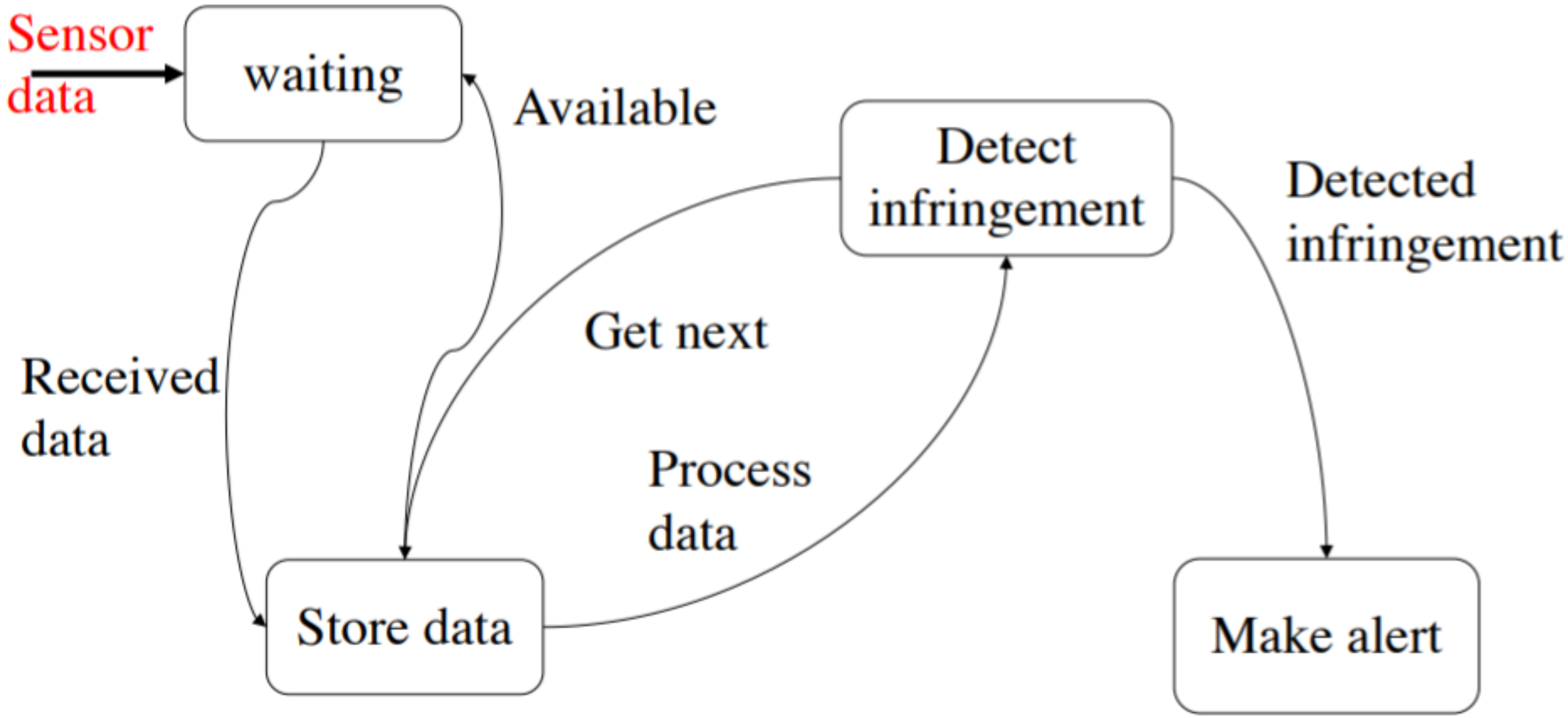
State Models

A state model describes the set of states that system runtime elements can be in and valid transition between them.

Entities:

- State – condition of the element
- Event – something of interest has happened
- Transition – change of state due to event
- Action – piece of processing associated with transaction

State Models



Self-Check

1. What are the two models that are used to represent concurrency?
2. Is concurrency model a UML diagram?
3. What factors would you use to allocate functionalities to processes?

Thank You