Edge-Stream

a Stream Processing Approach for Distributed Applications on a Hierarchical Edge-computing System

Xiaoyang Wang, Zhe Zhou, Ping Han, Tong Meng, Guangyu Sun, Jidong Zhai

aws

25 Regions with 80 Availability Zones

5 Local Zones and 13 Wavelength Zones

6 Regions and 12 Local Zones announced

230+ Edge Network Locations

108 AWS Direct Connect locations

Private network backbone between all AWS Regions, CloudFront PoPs. and Direct Connect locations

Redundant 100 Gbps links & encrypted network traffic

* Stats current as of Q2 2021

25 Regions with 80 Availability Zones

5 Local Zones and 13 Wavelength Zones

6 Regions and 12 Local Zones announced

230+ Edge Network Locations

108 AWS Direct Connect locations

Private network backbone between all AWS Regions, CloudFront PoPs. and Direct Connect locations

Redundant 100 Gbps links & encrypted network traffic

* Stats current as of Q2 2021

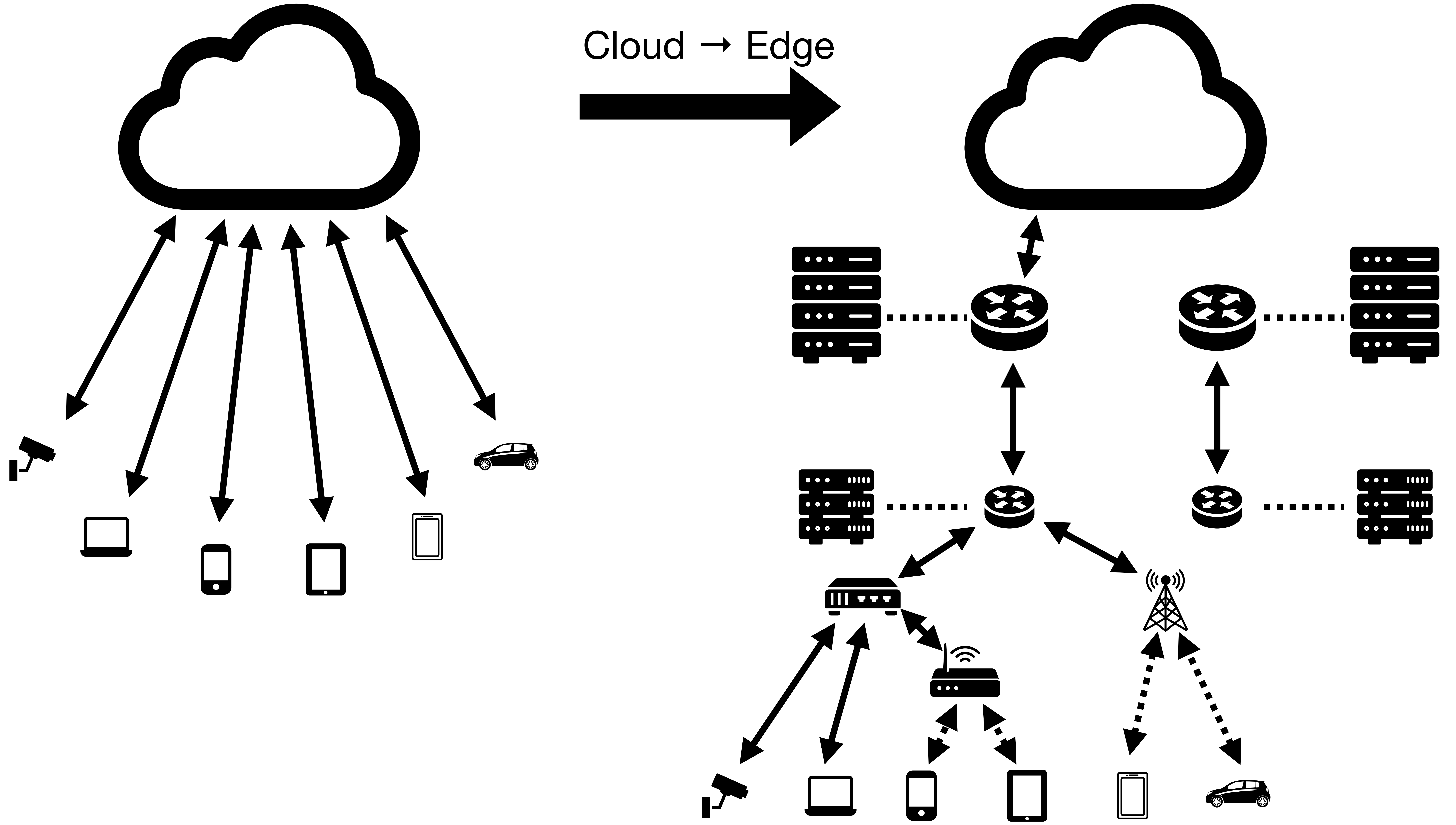
REGION
AZ
AZ
AZ
On-premises
Internet of Things
Rugged edge
The next frontier

Edge-Stream

a Stream Processing Approach for Distributed Applications on a Hierarchical Edge-computing System

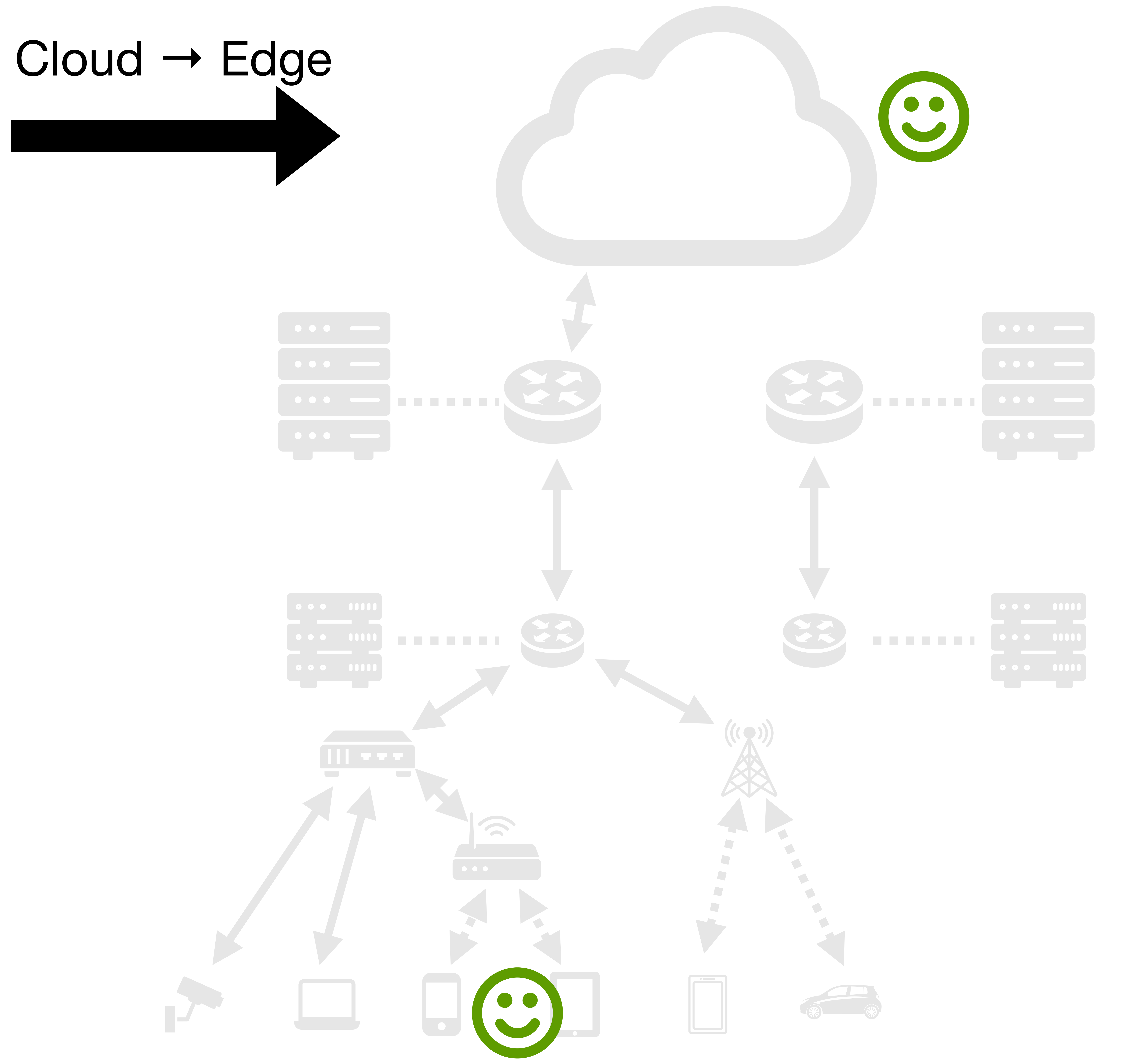
Xiaoyang Wang, Zhe Zhou, Ping Han, Tong Meng, Guangyu Sun, Jidong Zhai

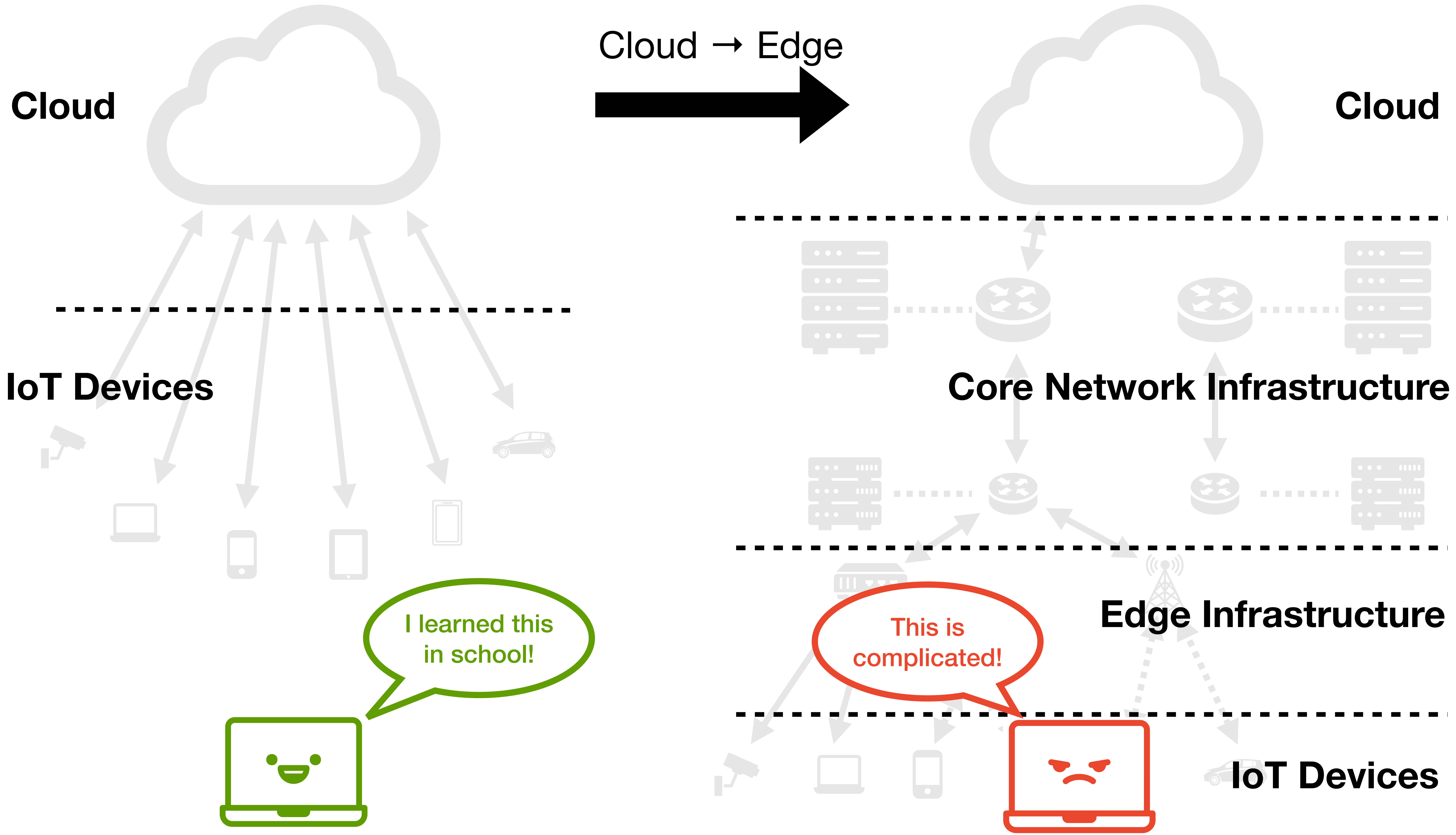
Why Edge Computing?


Cloud → Edge

verizon
Another traffic jam!
This is slow!
@#!*

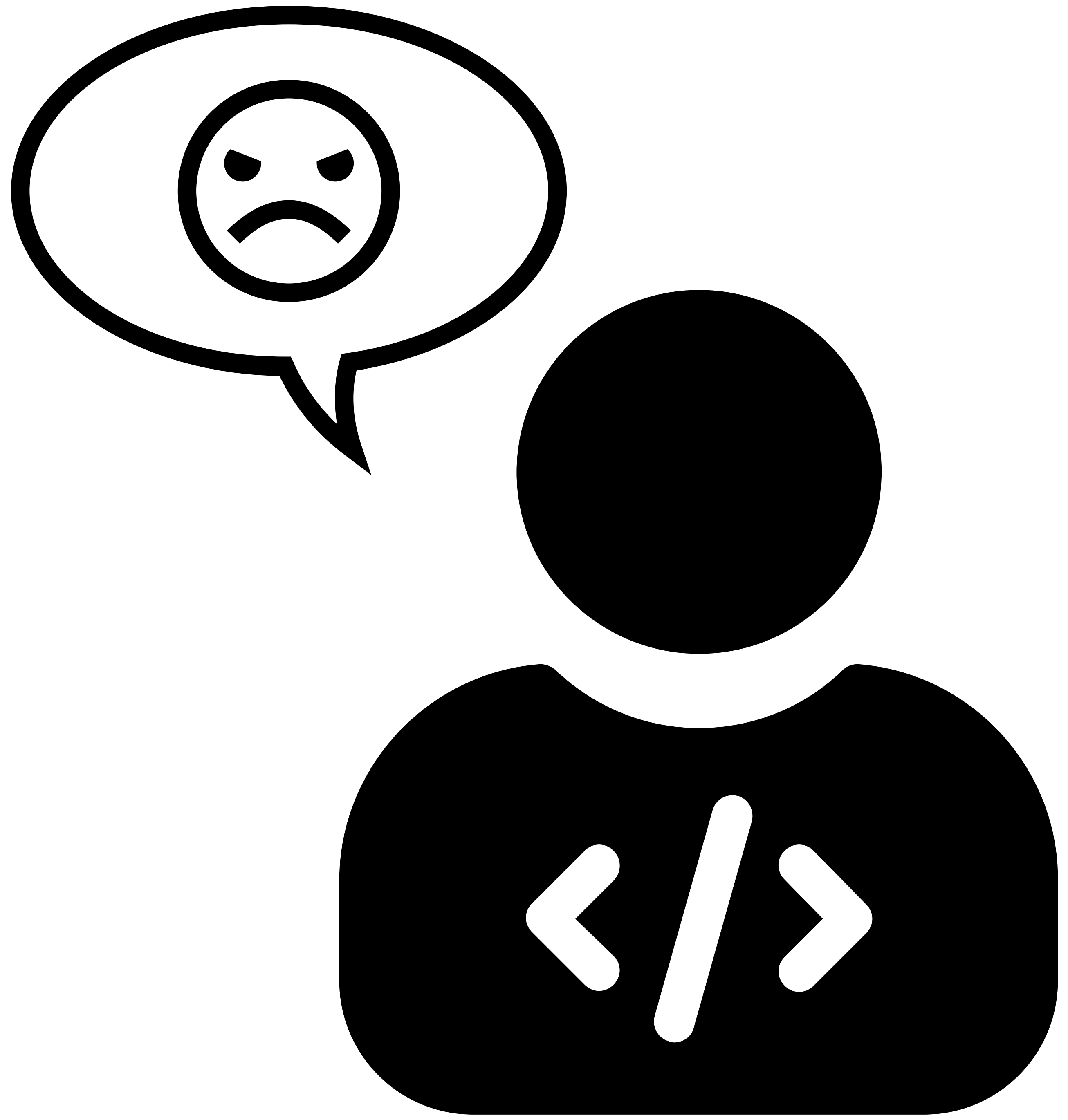
Cloud → Edge




Cloud
IoT Devices
I learned this in school!
Cloud → Edge
Cloud
Core Network Infrastructure
Edge Infrastructure
This is complicated!
IoT Devices

The Challenge

Developing for the Edge is inefficient because of a lack of high-level abstractions

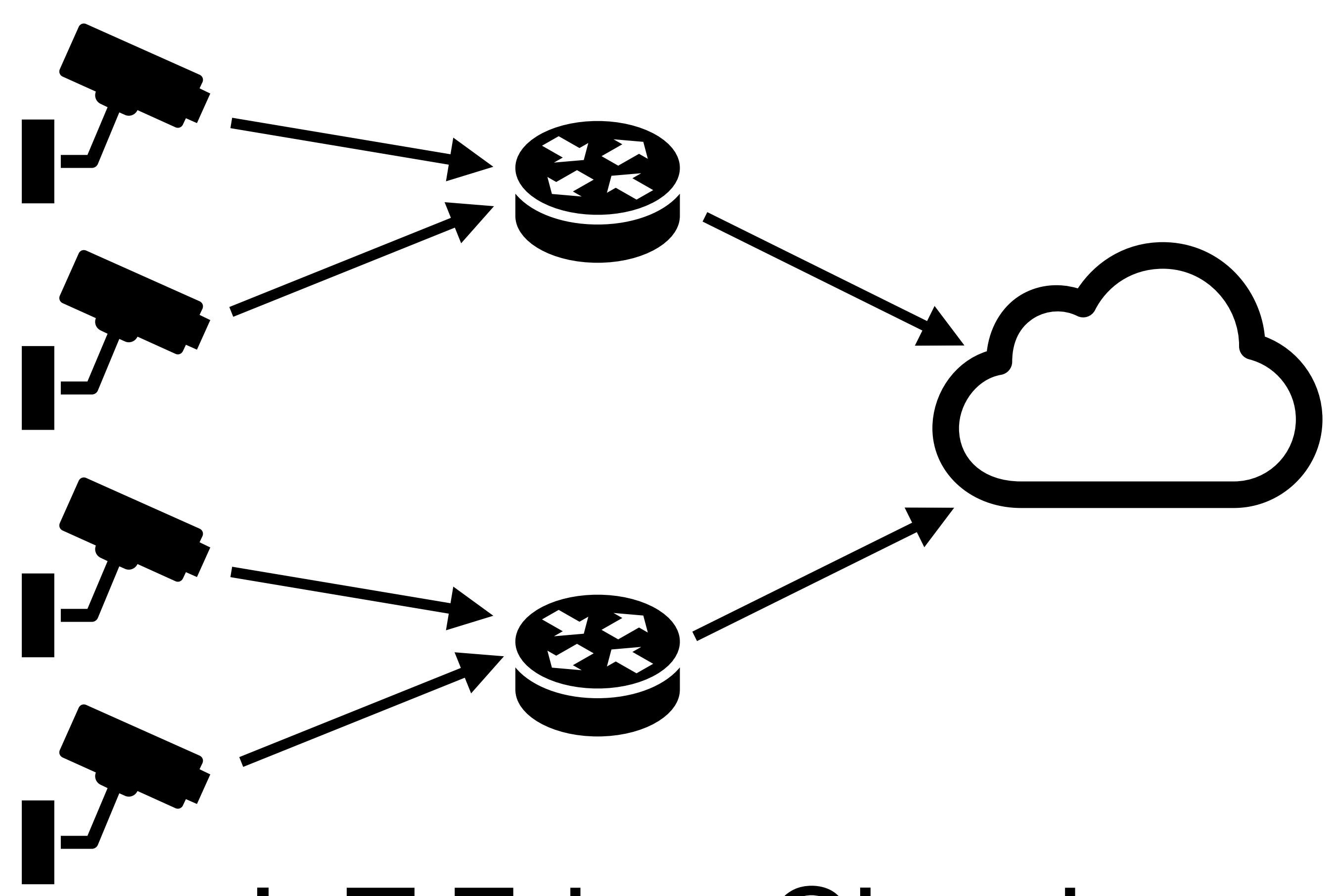


The Solution

Introduce high level abstractions for developing for the Edge

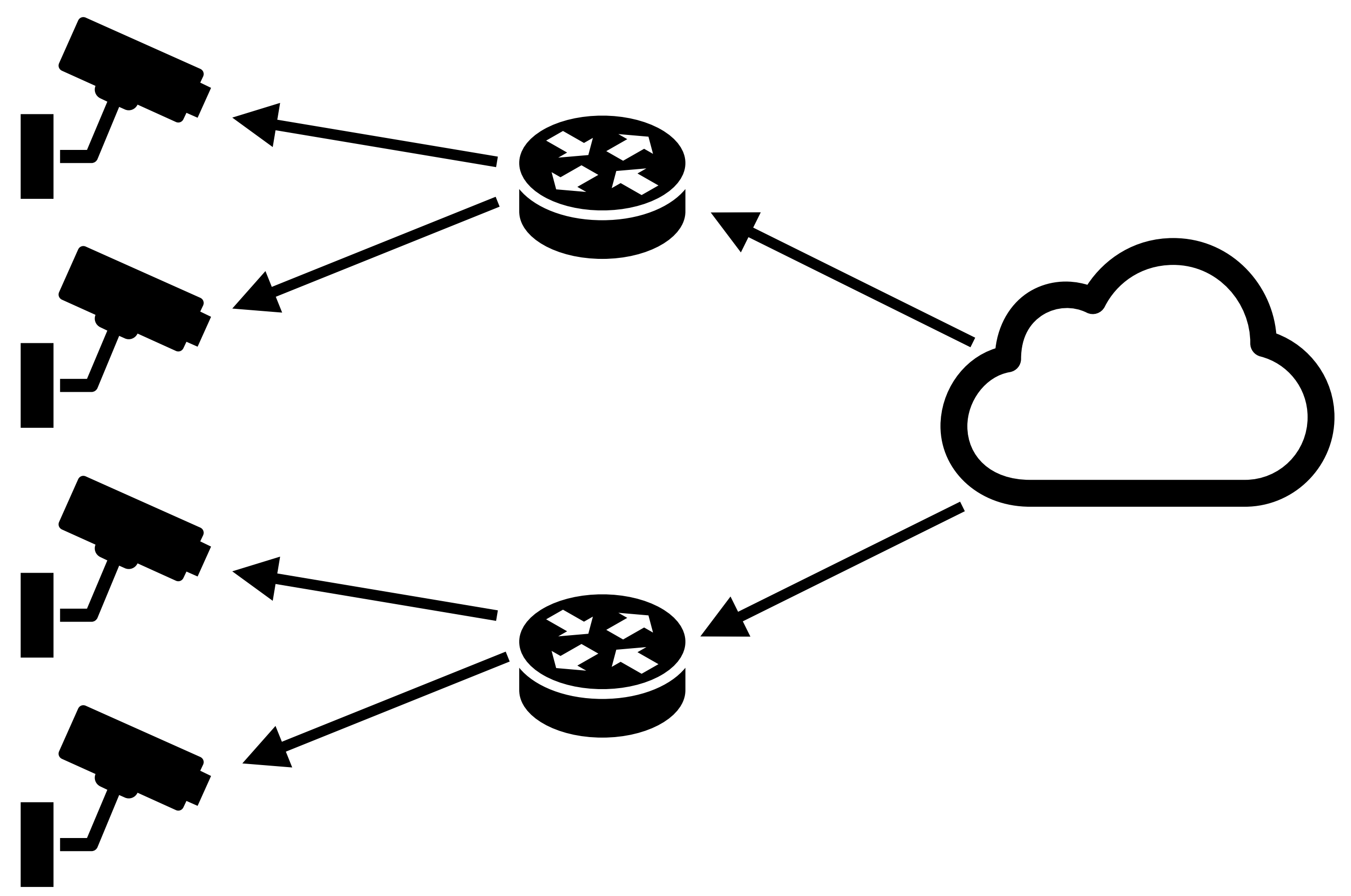
Scenarios

1.



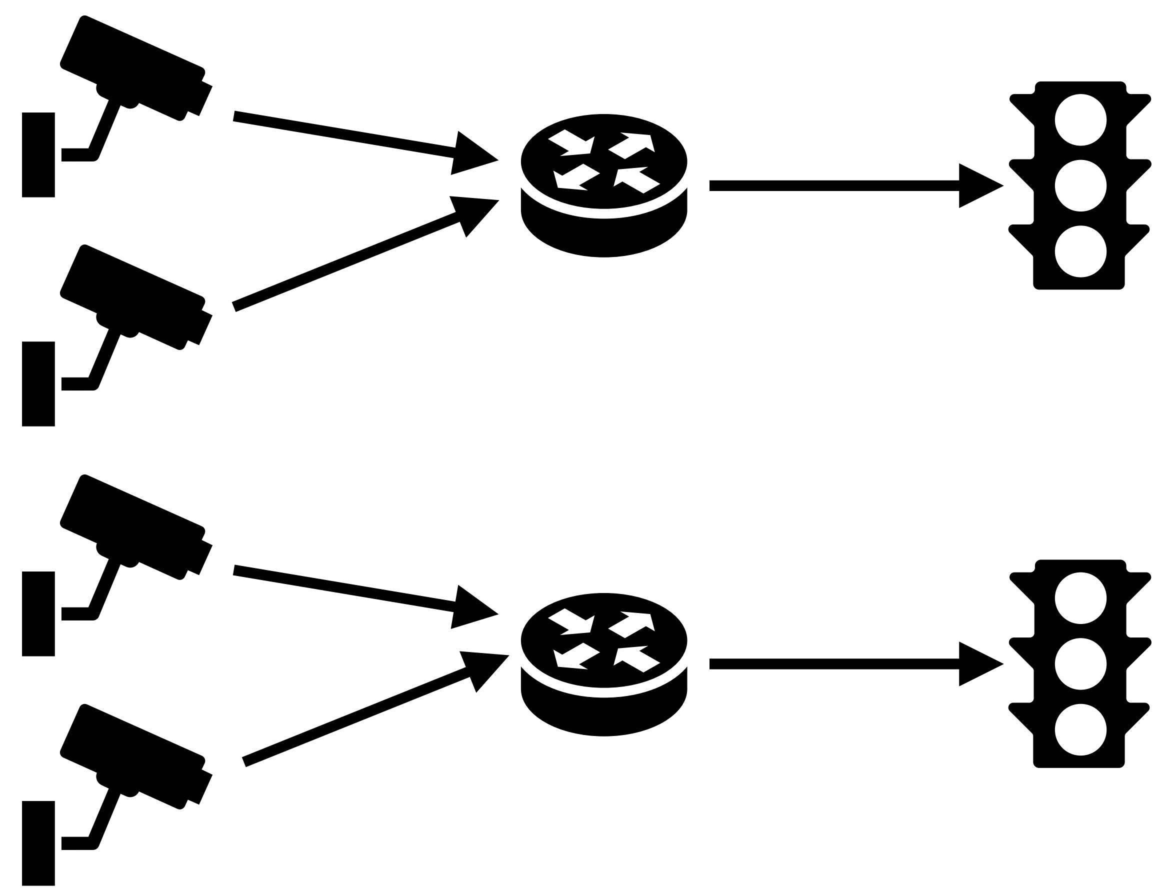
IoT-Edge-Cloud

2.



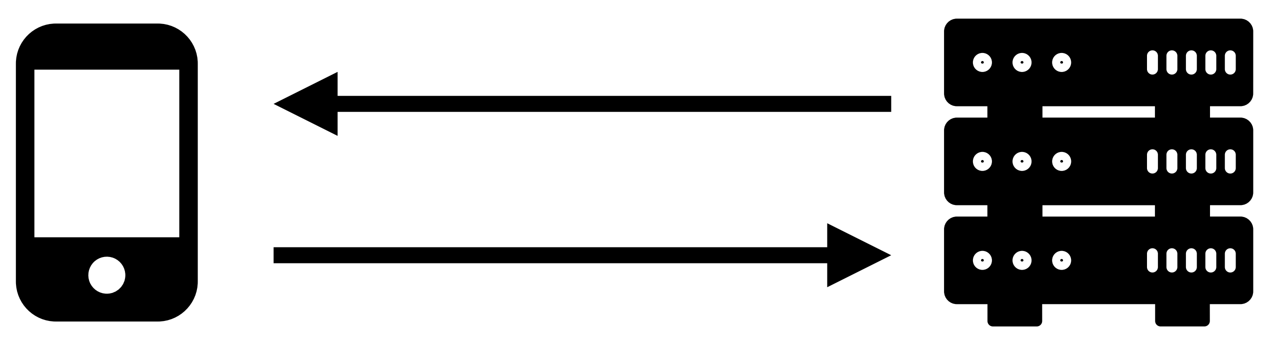
Cloud-Edge-IoT

3.



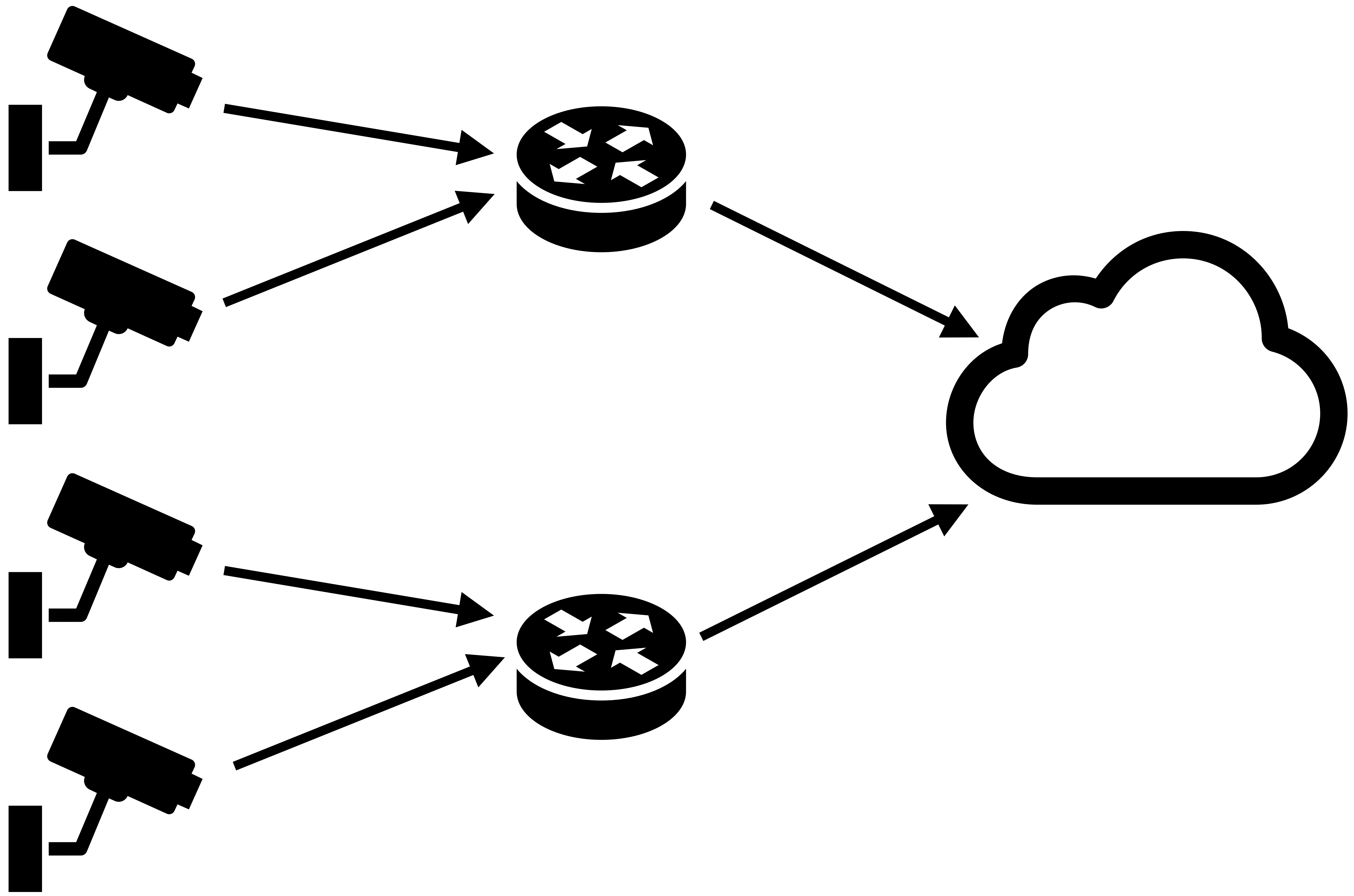
IoT-Edge-IoT

4.

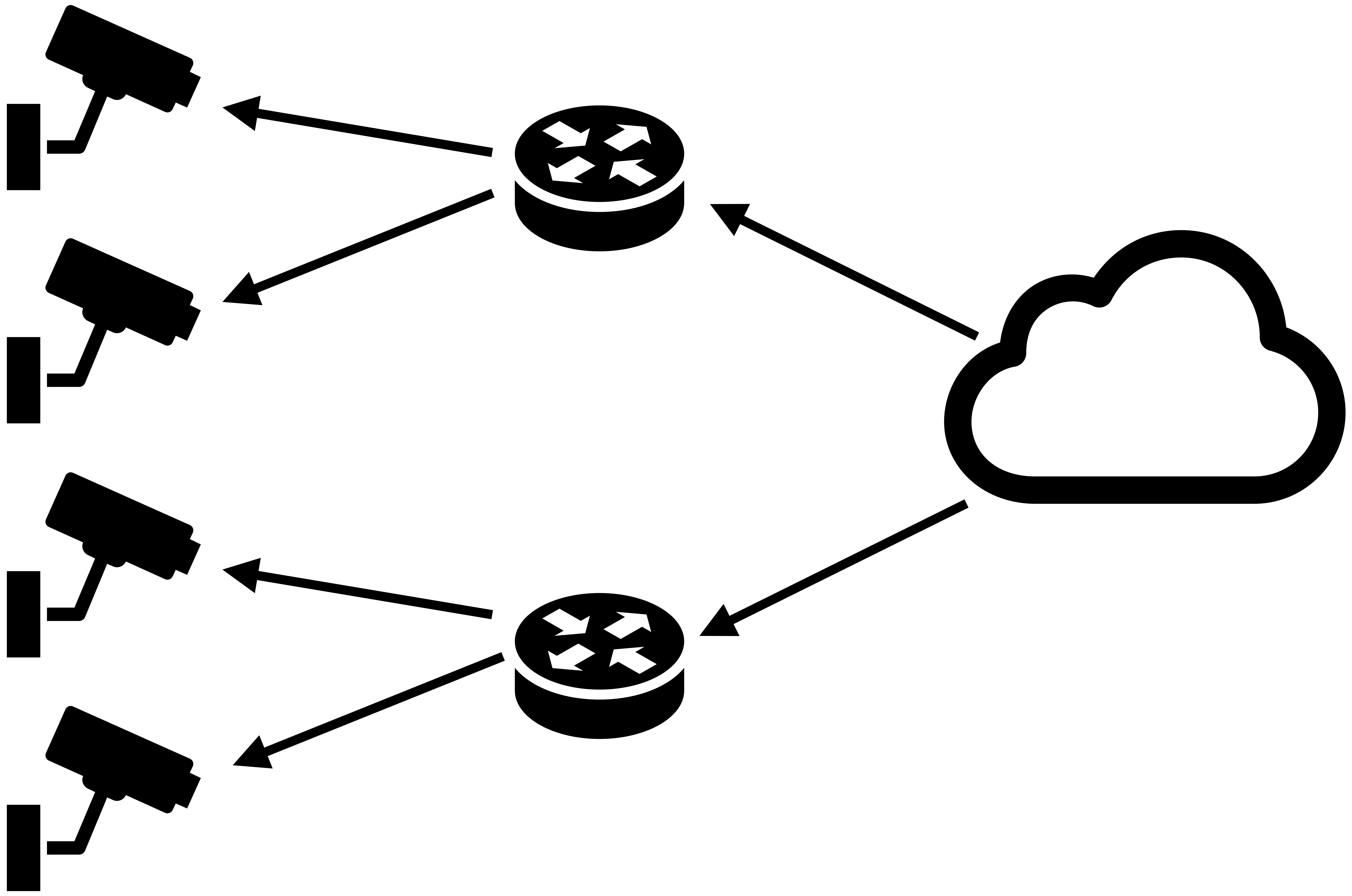


IoT-Edge

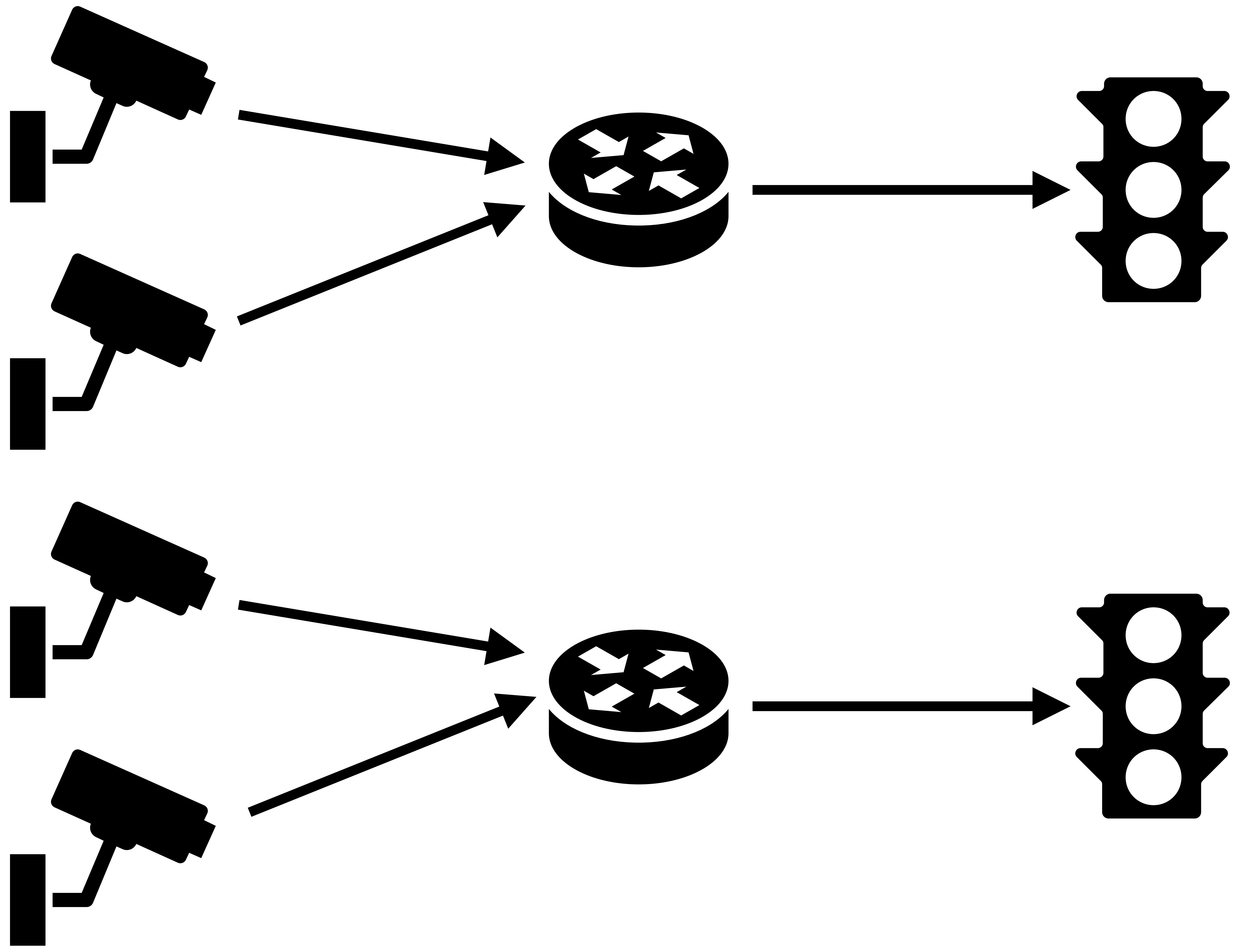
1 IoT-Edge-Cloud



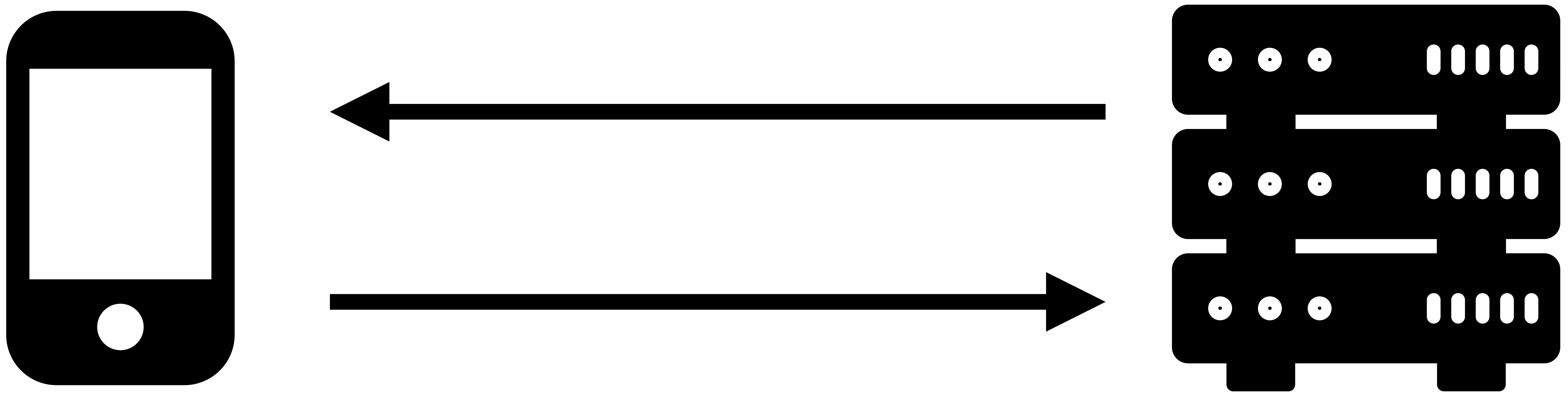
2 Cloud-Edge-IoT



3 IoT-Edge-IoT



4 IoT-Edge



Create an abstraction that handles **all four** of these scenarios

Agenda

Agenda

Agenda

How to develop on Linux?

Files

Users Manage and Share **Data Blocks** by **File**

Commands

Commands abstract "format" from the specific "content" of input **Files**.

Scripts

Scripts are composed of pre-defined **Commands**.

Files

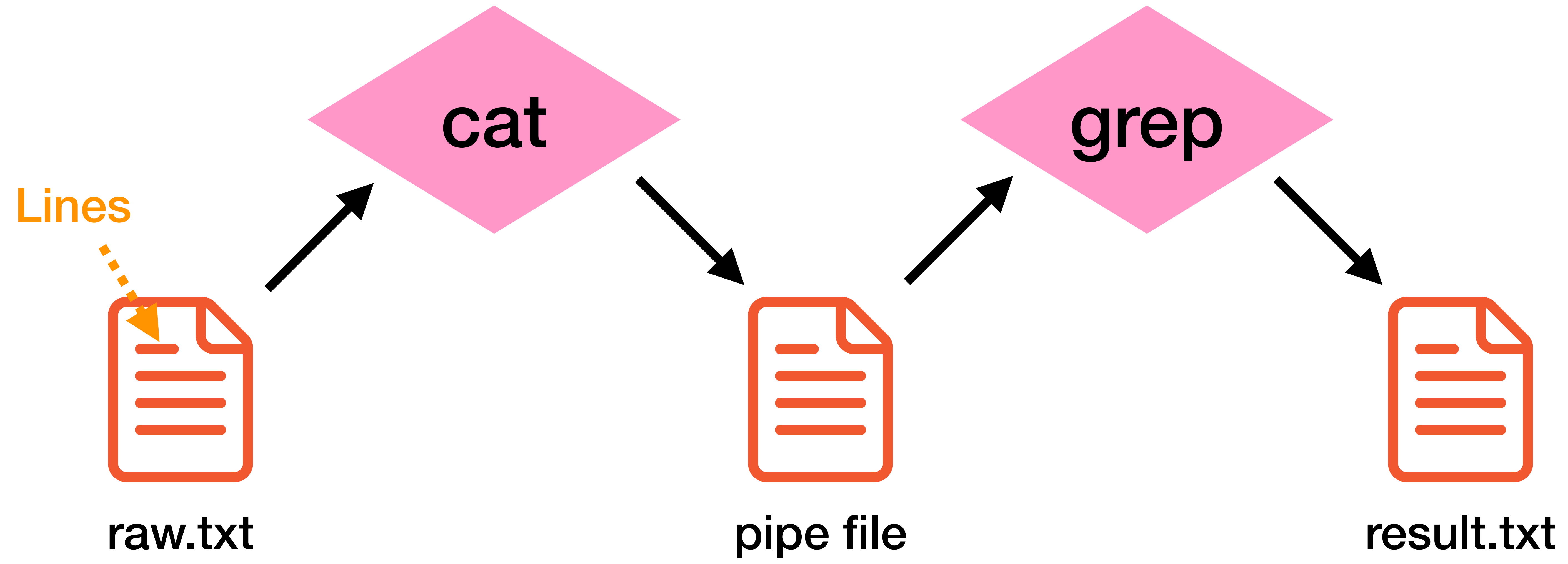
Users Manage and Share **Data Blocks** by **File**

Commands

Commands abstract “format” from the specific “content” of input **Files**.

Scripts

Scripts are composed of pre-defined **Commands**.



```
cat raw.txt | grep “hello” > result.txt
```

How to develop on the Edge?

Streams

Users Manage and Share **Data Sequences** by **Stream**

Operators

Operators abstract “format” from the specific “content” of input **Streams**.

Applications

Applications are composed of pre-defined **Operators**.

Streams

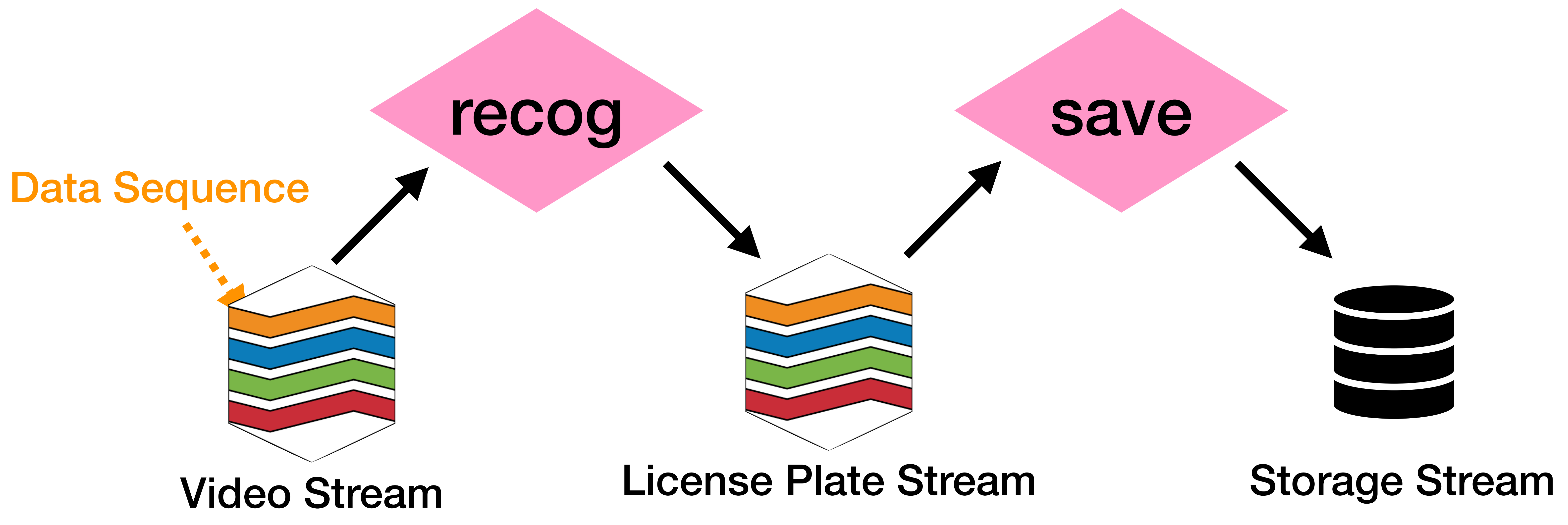
Users Manage and Share **Data Sequences** by **Stream**

Operators

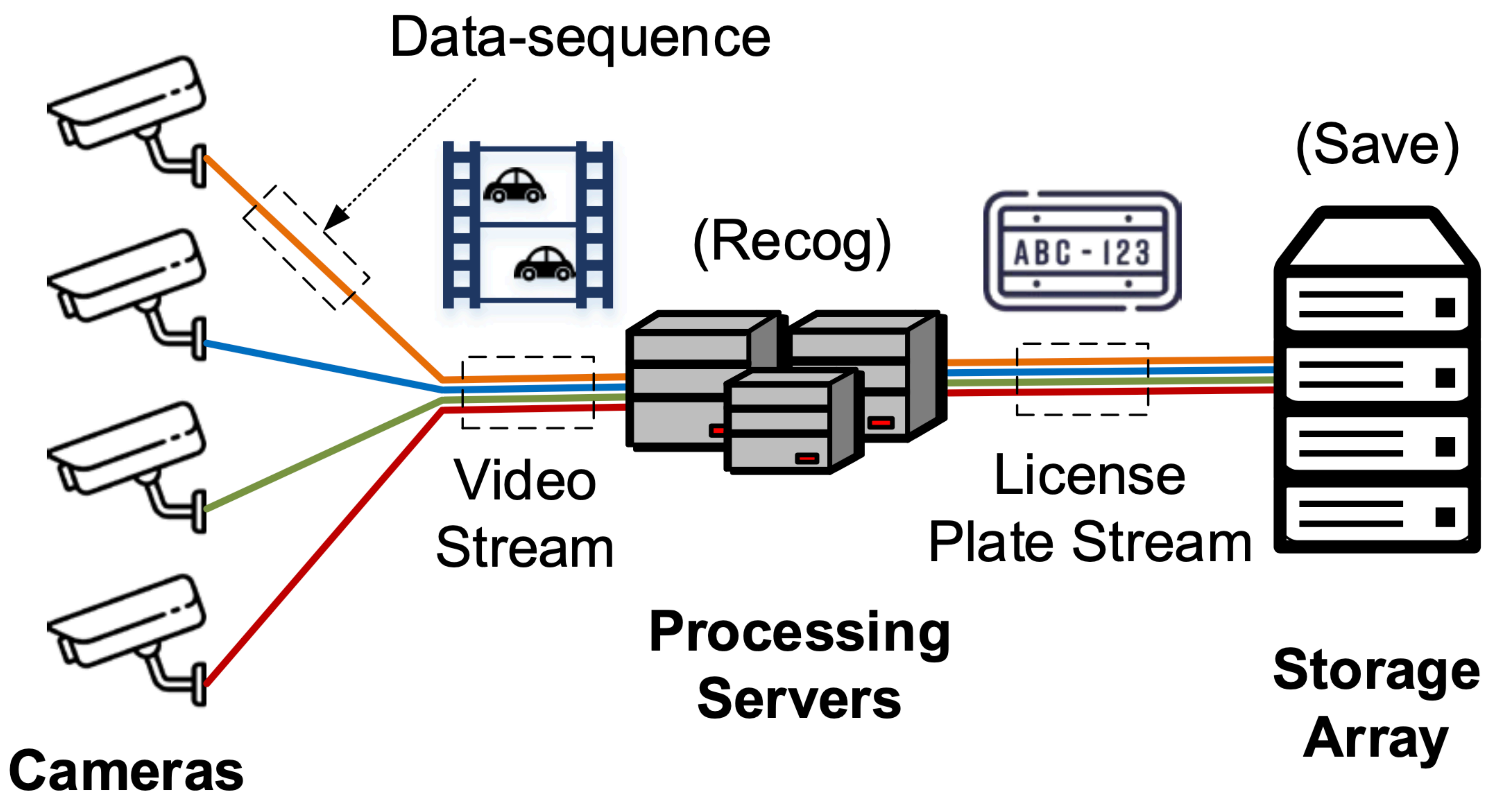
Operators abstract “format” from the specific “content” of input **Streams**.

Applications

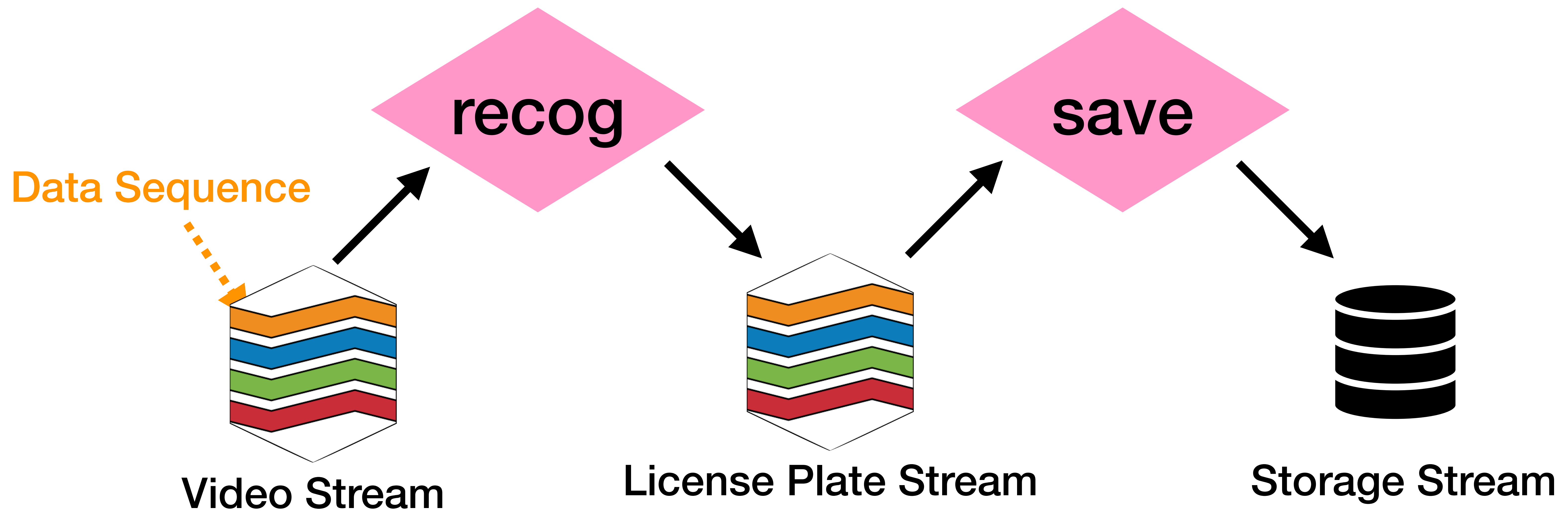
Applications are composed of pre-defined **Operators**.



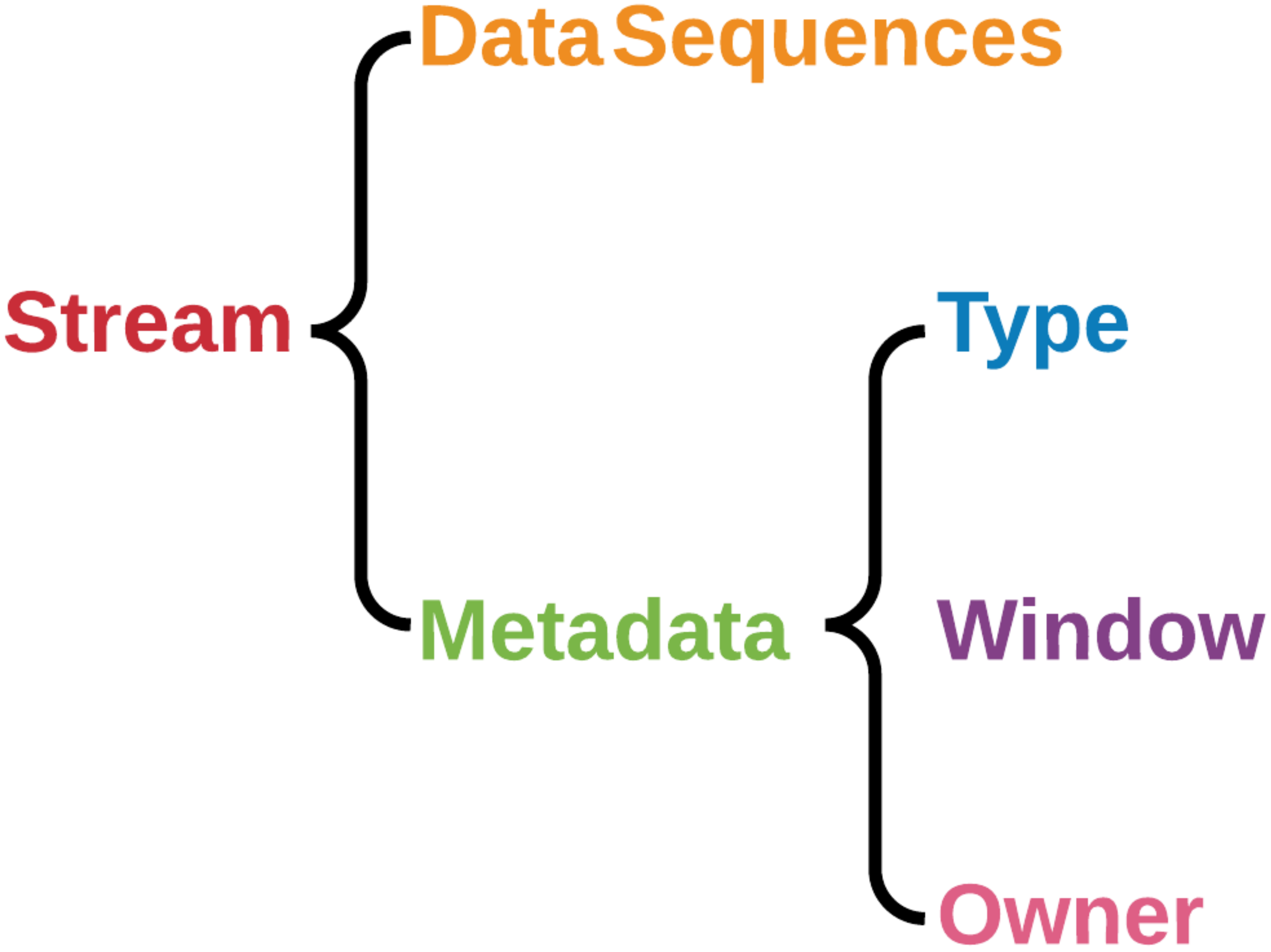
Physical System



Abstract View

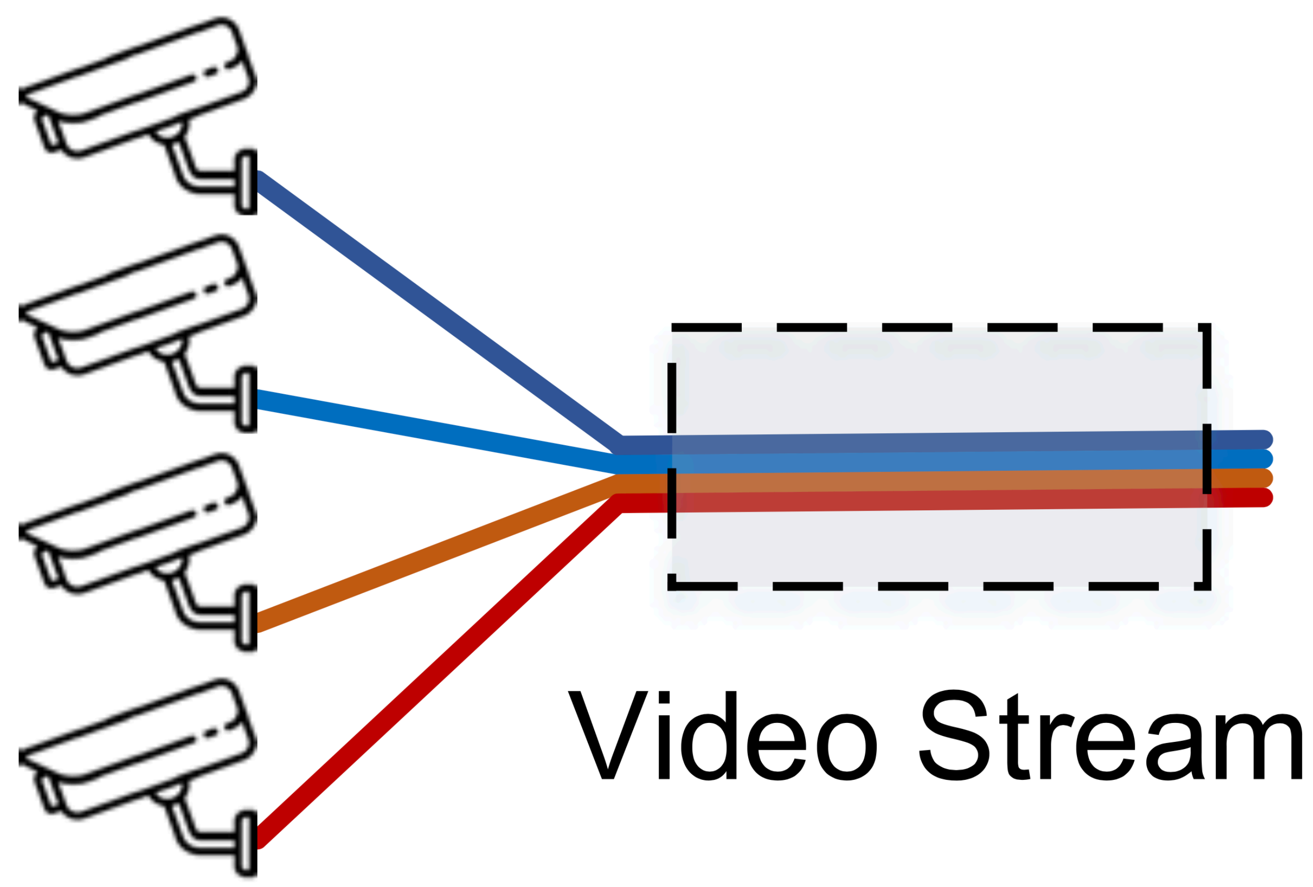


What is a Stream?


Stream
Data Sequences
Metadata
Type
Window
Owner

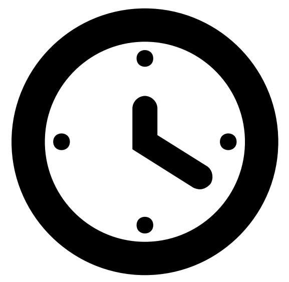
Stream Type

Primitive



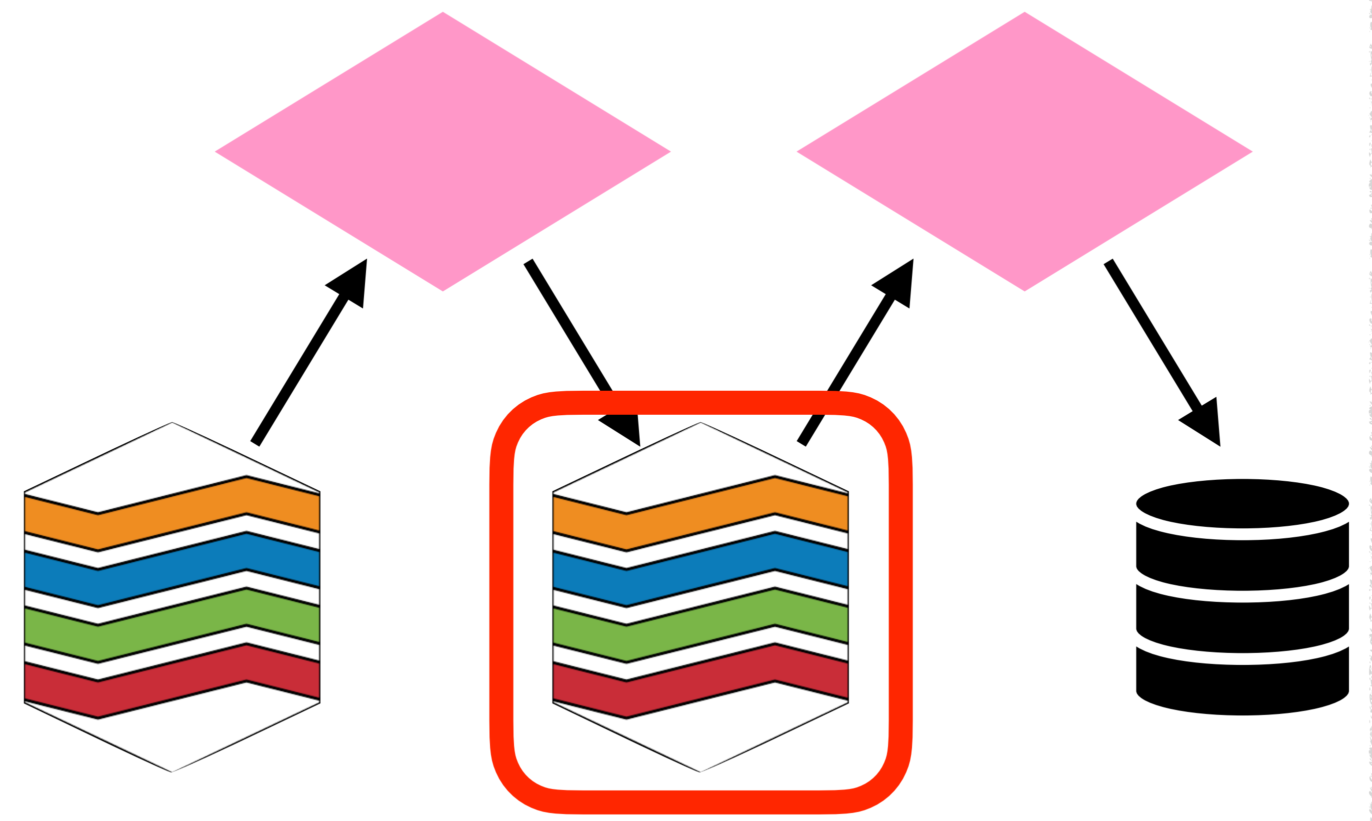
Virtual

Timer Stream

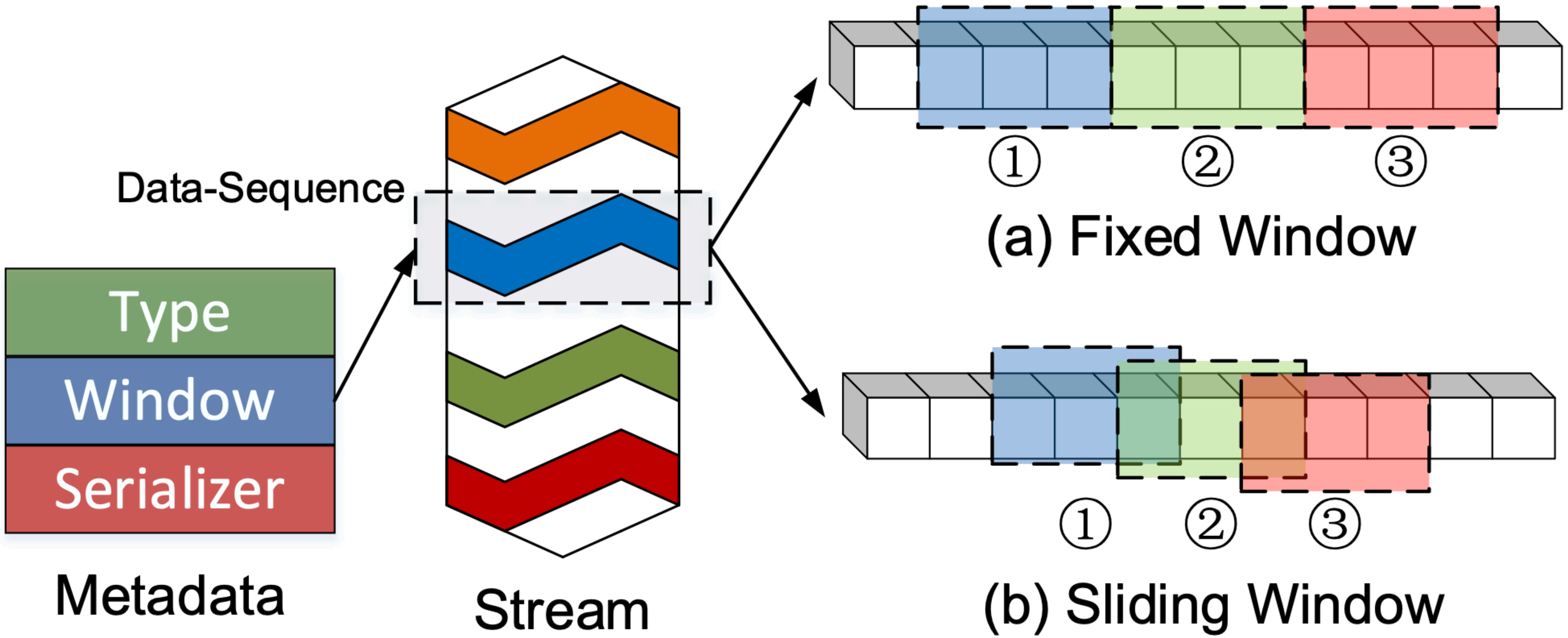


[..., 1:01, 1:02, 1:03...]

Generated



Window


Data-Sequence
Type
Window
Serializer
Metadata
Stream
①
②
③
(a) Fixed Window
①
②
③
(b) Sliding Window

What is an Operator?

Three Operator Types

Reshape

Define how to organize existing data-sequences, without changing the data inside.

Compute

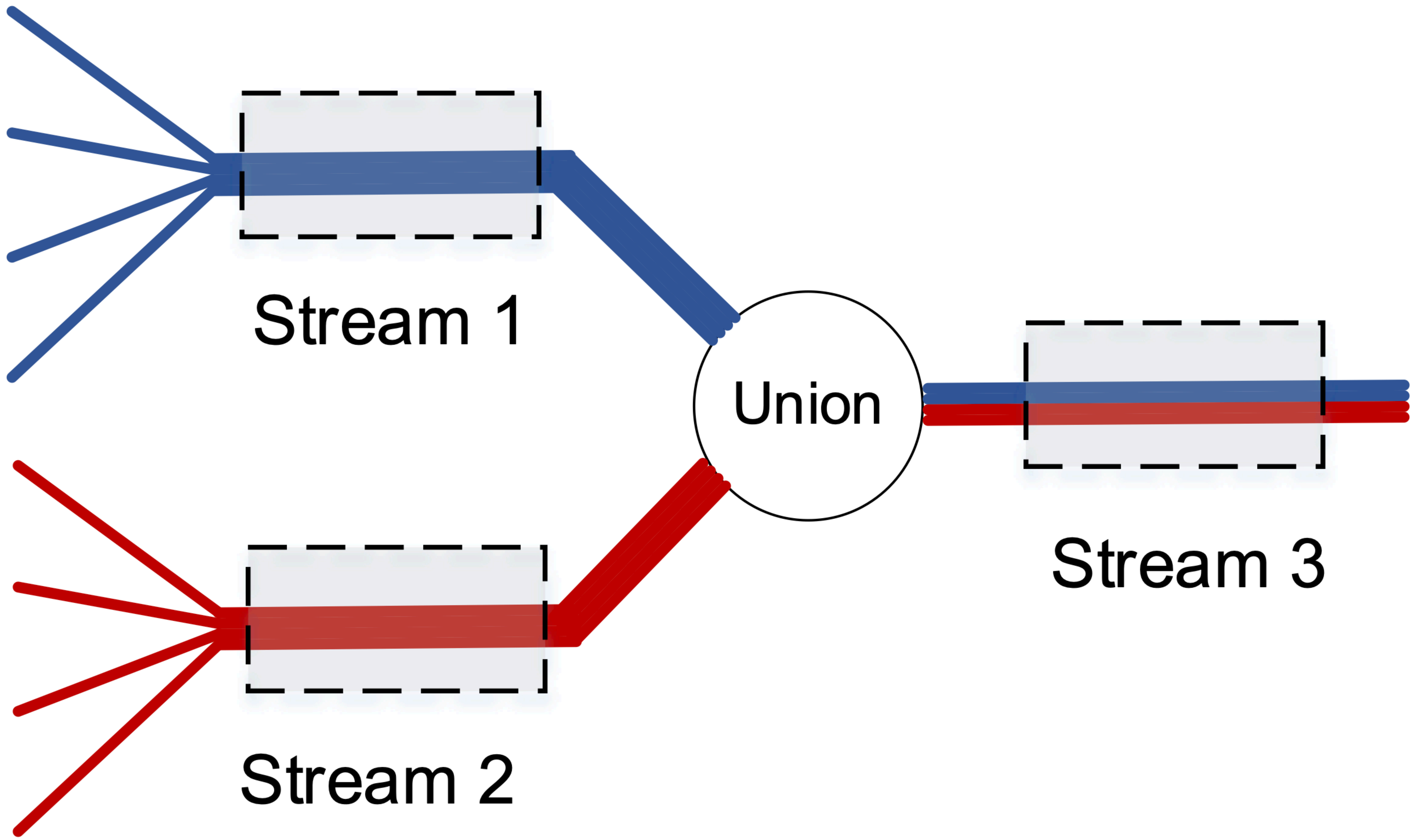
Generate new data from input streams with functions.

Group

Reorganize data-sequences

Reshape

Define how to organize existing data-sequences, without changing the data inside.



Compute

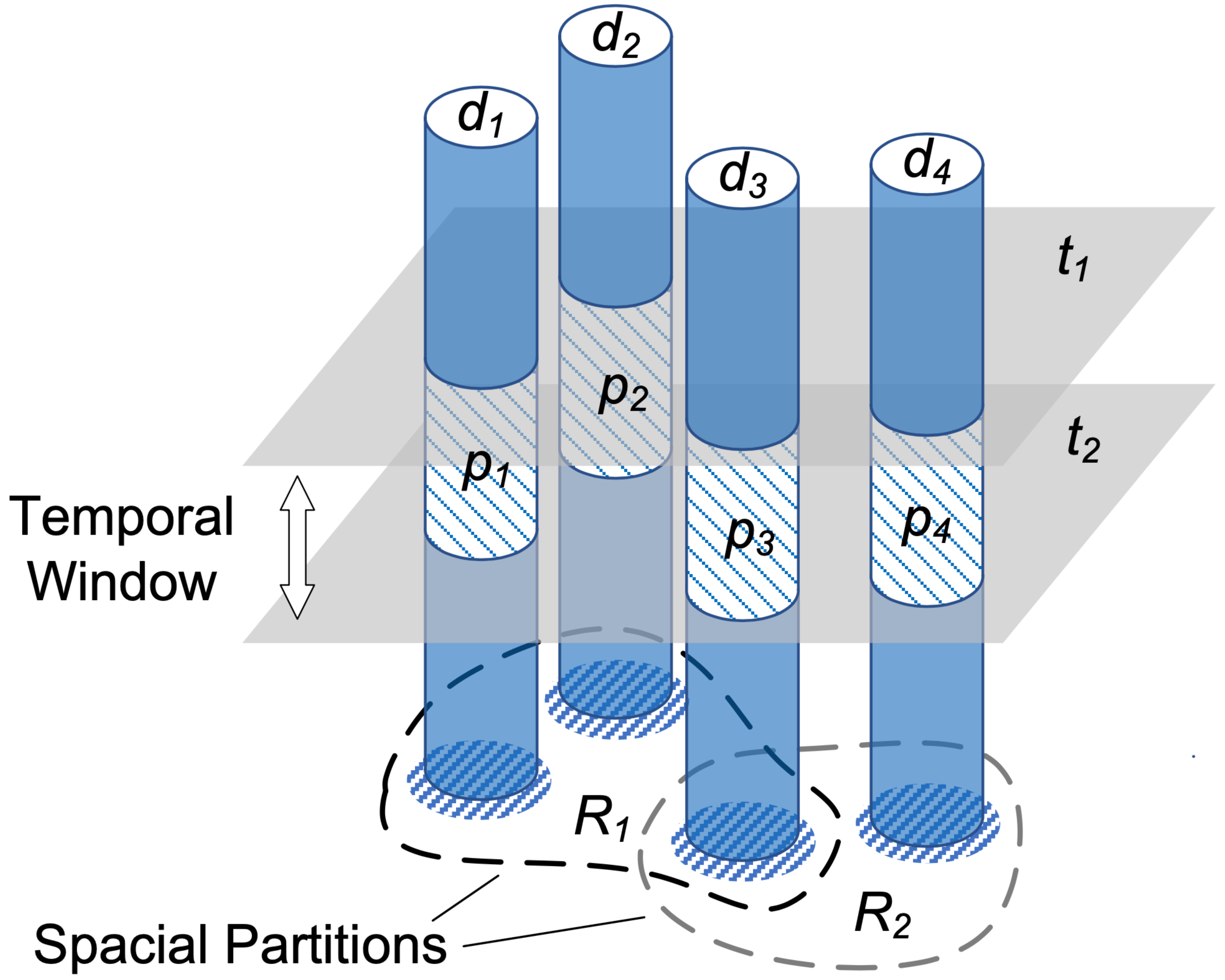
Generate new data from input streams with functions.
Functions access data through a standard set of APIs (map/reduce)

```
#include <string>
#include "MyRecogLib"
%%
%in S_video<Picture, null, File>
%out S_plate<std::string, null, JSON>
%%
%{
  auto inPicture = S_video.getNext();
  auto outPlate = PlateRecog(inPicture);
  S_plate.pushItem(outPlate);
%}
```

```
#include <string>
%%
%in S_plate<std::string, fixed, JSON>
%out S_result<int, null, JSON>
%%
%{
  int counter = 0;
  auto plates = S_plate.getWindow();
  for (plate : plates) {
    counter ++;
  }
  S_result.pushItem(counter);
%}
```

Group

Grouping provides spacial partitions (Windows generate temporal slices).

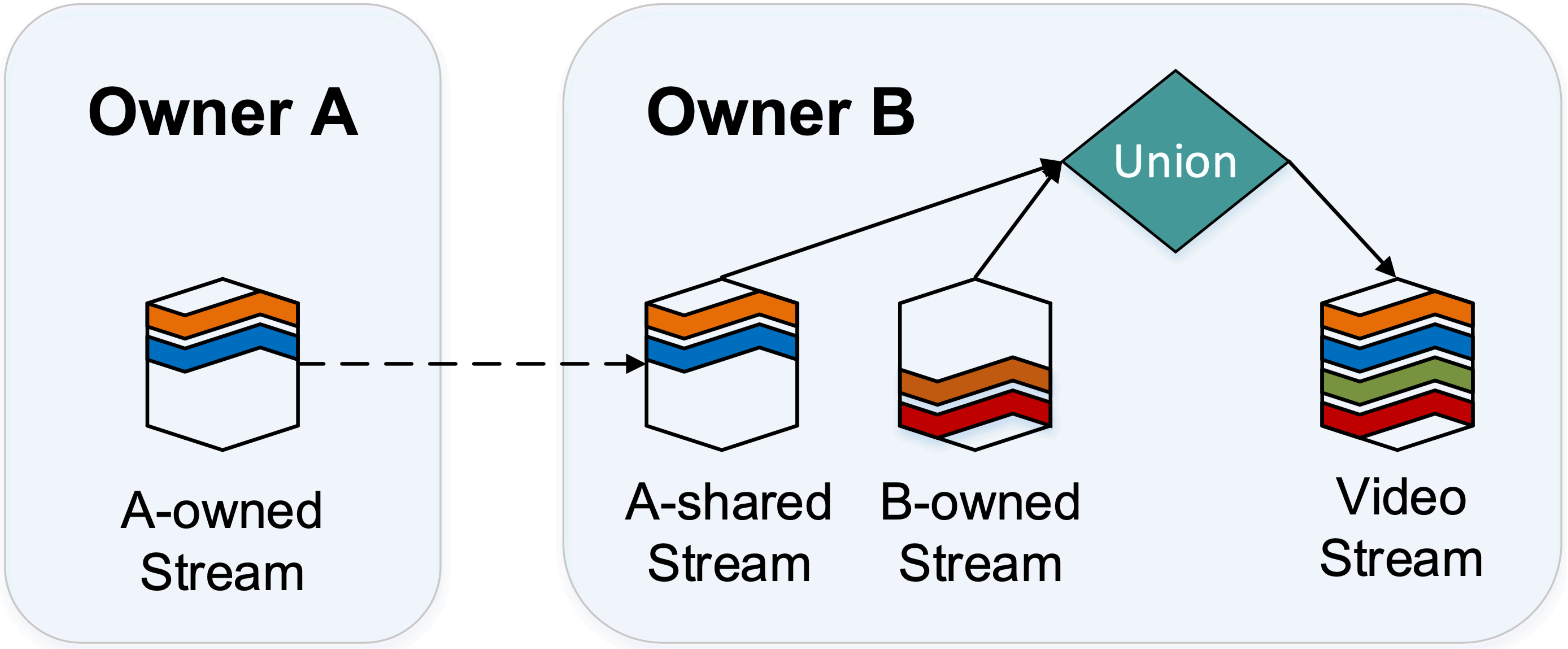


How do I share a Stream?

Each stream has a unique owner.

The owner is able to share the stream to other users.

Those users are allowed to build new streams from it, but cannot modify or delete the original stream.



Agenda

Agenda

EStream

A prototype realization of Edge-Stream

- **Help to verify the benefit from the new model**
- **Provide a practical scheduling method**

Agenda

Discussion

- **Familiar Interface.** Don’t reinvent the wheel.
- **Useful.** A “file system” for the edge fills a legitimate need.

- **Conceptual.** Needs additional work before adoption.
- **Testability.** Developer productivity is difficult to measure.

Any Questions?

Personally, do you find the Edge Stream model intuitive

What work would have to be done if Edge Stream were to become commercially viable.