RADICALBIT

Deep, Different, Disruptive.

Introduction to Apache Flink

Jug Milano, 16/03/16

Business Operations Technology

Agenda

1. Apache Flink
2. Flink Applications @Radicalbit

First Part

Apache Flink

Part 1 Agenda

1. What is Apache Flink?
2. Dataflow Programming Model
3. Architectural concept
4. Programming Model and Example
5. Fault tolerance mechanism
6. Memory Management system

1. What is Apache Flink?

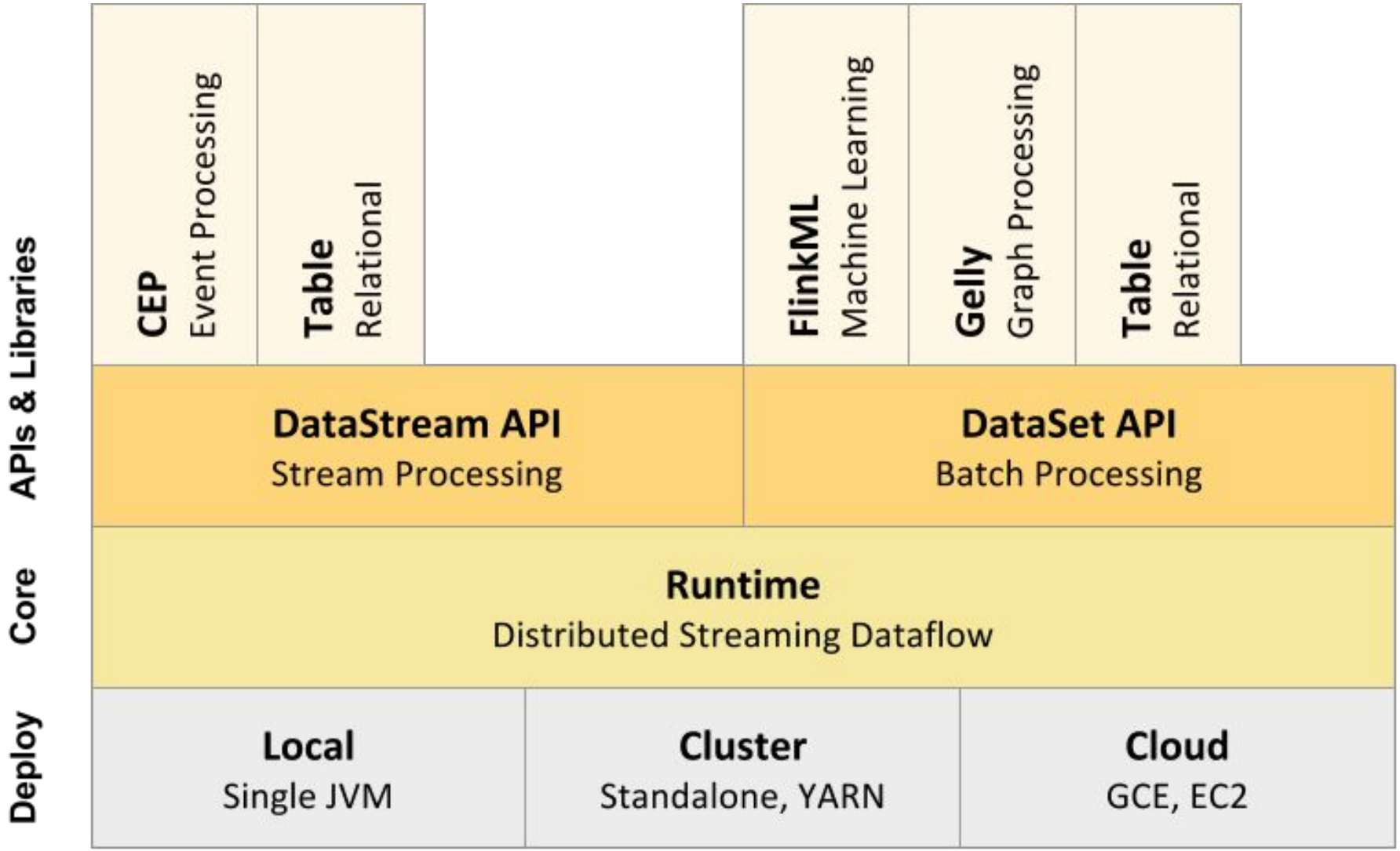
Apache Flink

- An open source platform for distributed stream and batch data processing;
- Its streaming dataflow engine provides:
  - a **fine-grained** control of execution
  - optimization, parallelization and scheduling
  - communication and data distribution
  - **fault tolerance** via asynchronous distributed snapshots
  - a sophisticated **memory management** system
- It builds batch processing on top of the streaming engine.

Project history

- 2010: born as a research project called "*Stratosphere: Information Management on the Cloud*" funded by German Research Foundation (DFG) and in collaboration with TU Berlin.
- Flink committers are currently employed by **data Artisans**: company that was founded by the original creators of Apache Flink
- March 2014: became an **Apache Incubator project**
- December 2014: was accepted as an **Apache top-level project**
- February 2017: stable release Apache Flink 1.2

Apache Flink Stack



RADICALBIT
Deep, Different, Disruptive.

2. Dataflow Programming Model

From Flink code to Streaming Dataflow

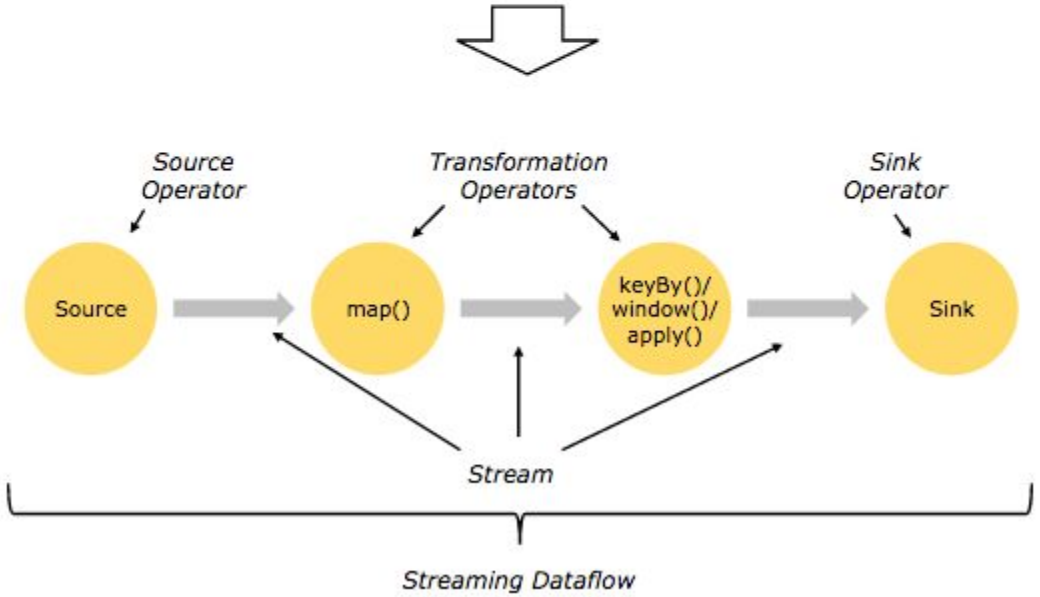
```
DataStream<String> lines = env.addSource(
                            new FlinkKafkaConsumer<>(…));

DataStream<Event> events = lines.map((line) -> parse(line));

DataStream<Statistics> stats = events
        .keyBy("id")
        .timeWindow(Time.seconds(10))
        .apply(new MyWindowAggregationFunction());

stats.addSink(new RollingSink(path));
```

Source

Transformation

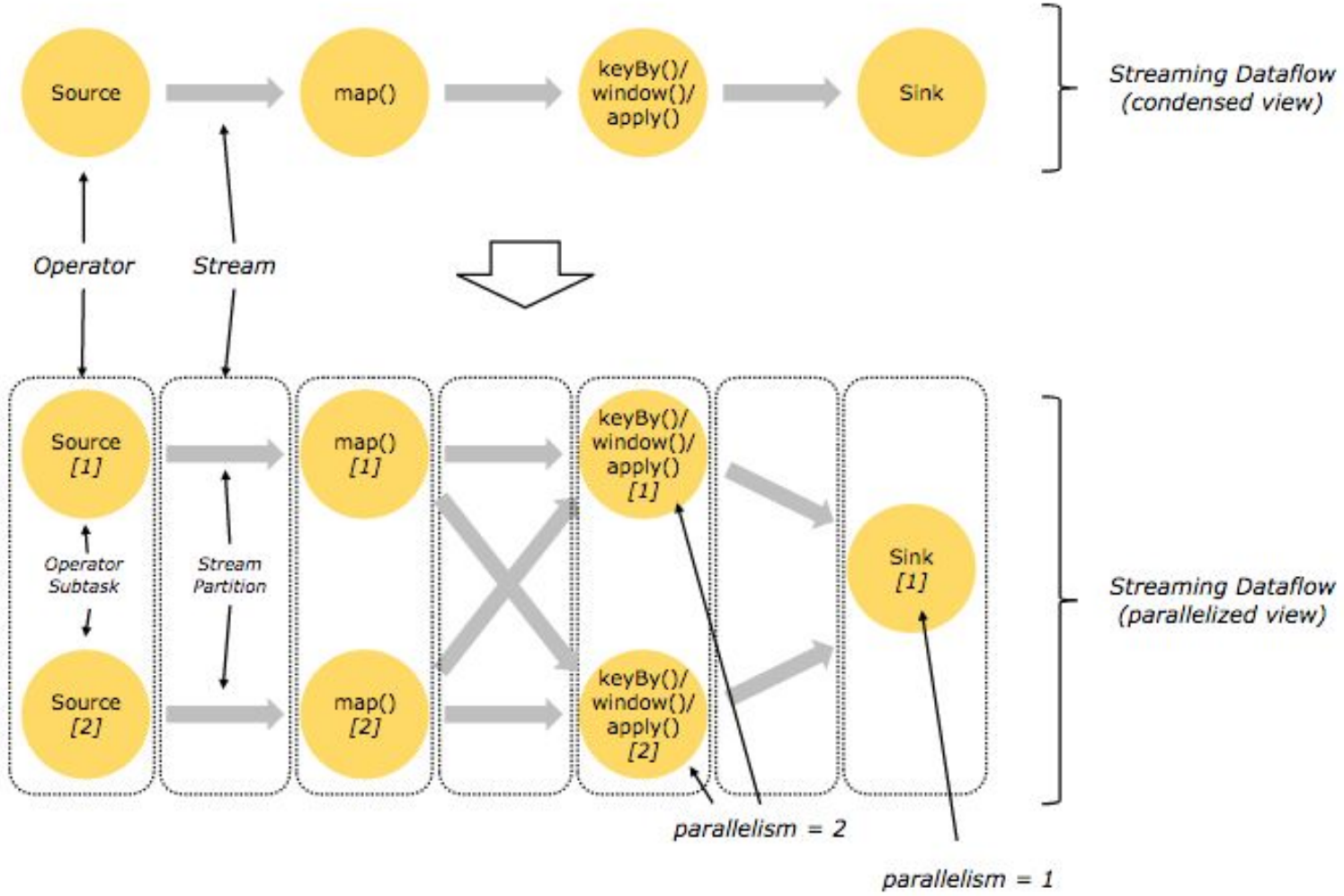
Transformation

Sink



RADICALBIT
Deep, Different, Disruptive.

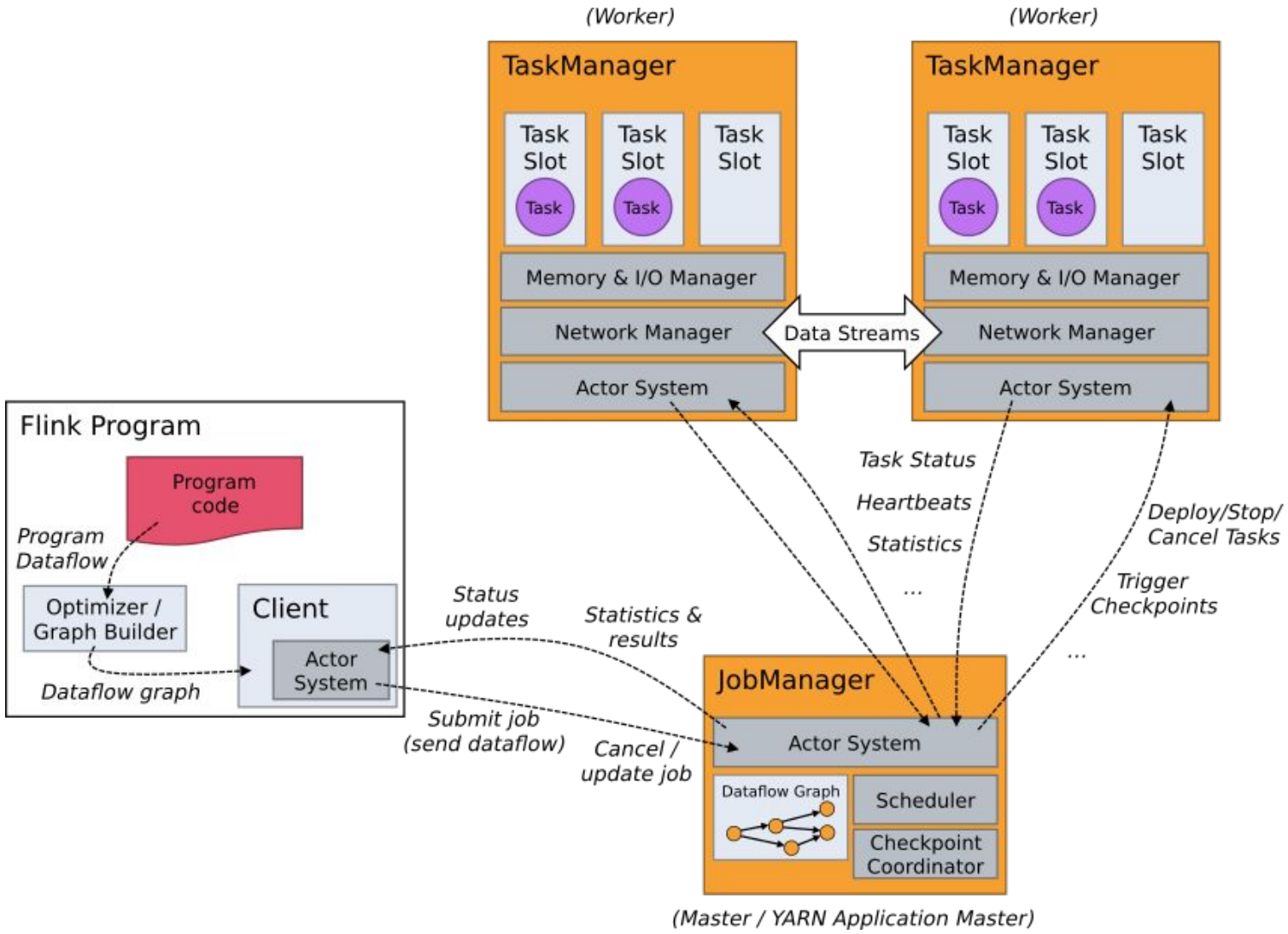
Parallel Streaming Dataflow



RADICALBIT
Deep, Different, Disruptive.

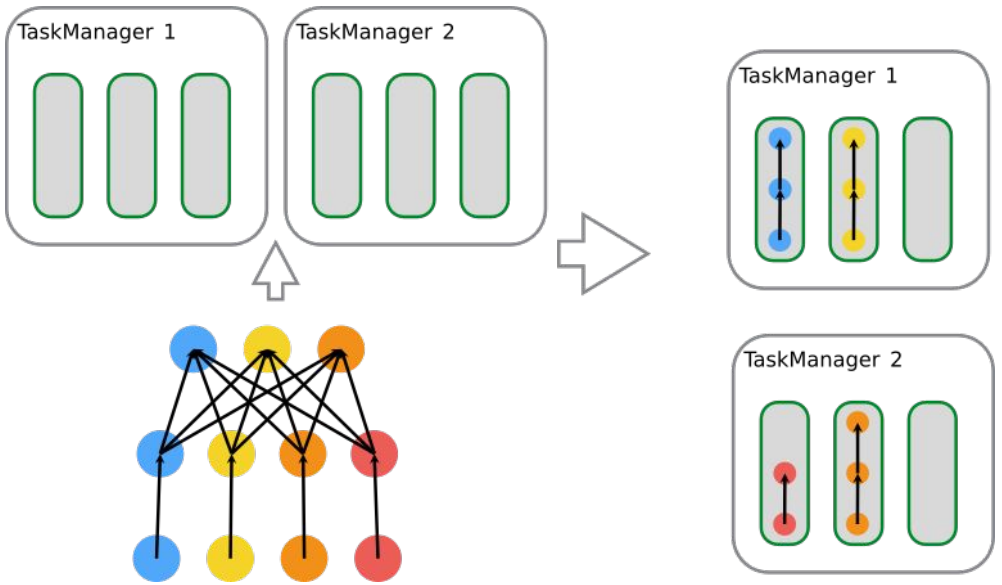
3. Architectural Concept

The process model



RADICALBIT
Deep, Different, Disruptive.

Scheduling



- Task slot: execution resource owned by a task manager
- Each slot executes a pipeline (i.e. multiple successive tasks)
- Tasks in a pipeline can be ran concurrently in a non-blocking fashion.

3. Programming Model

Anatomy of a Flink program

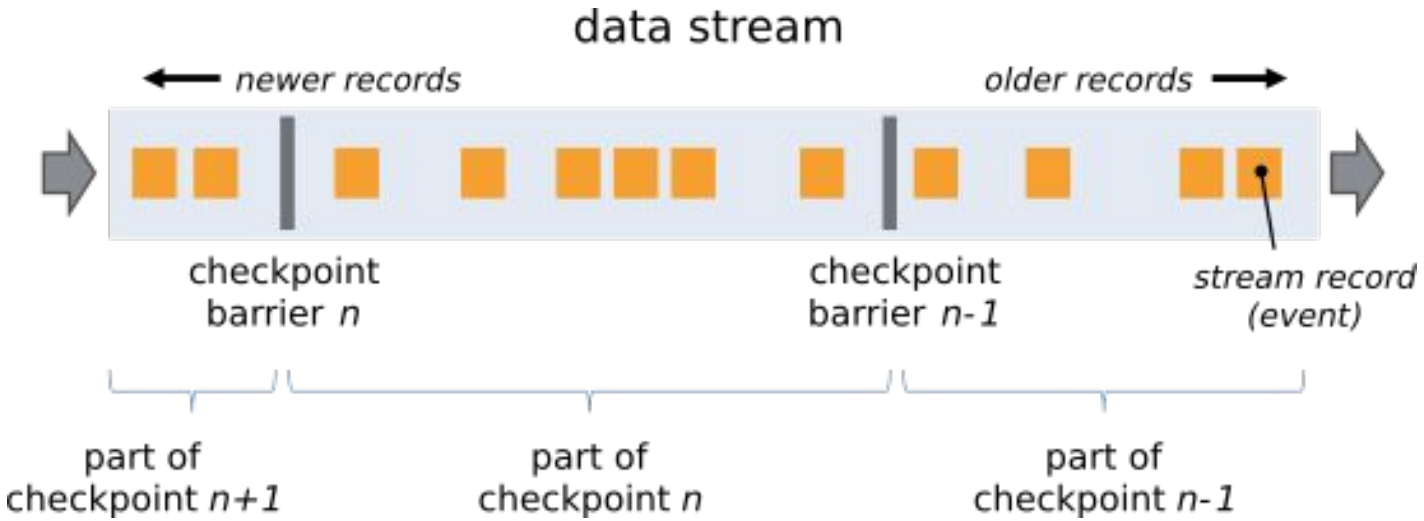
1. Obtain an *execution environment*
2. Load/create the initial data
3. Specify transformations on this data
4. Specify where to put the results of your computations
5. Trigger the program execution

Small Streaming Flink Example

...

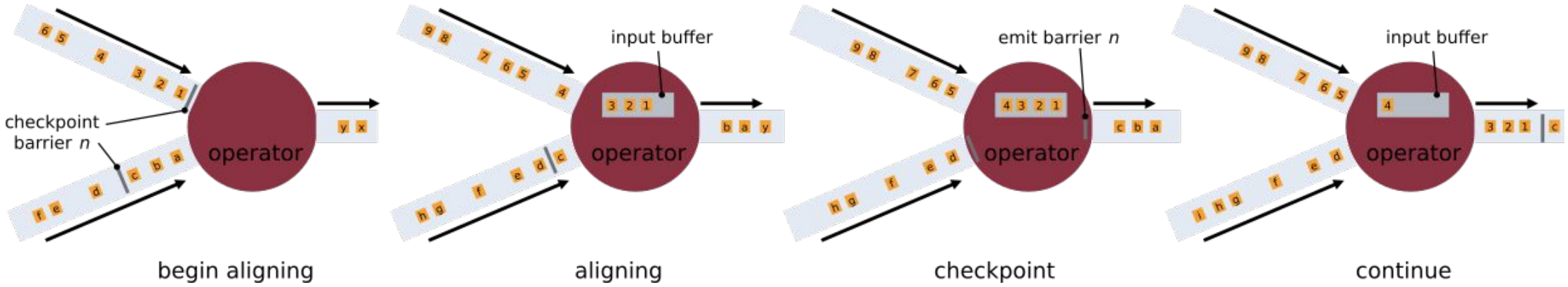
4. Fault Tolerance

Barriers



- Barriers are *injected* from the source *dividing* the stream in checkpoints;
- **Operators forward barriers** as they ingest inputs and produce outputs;
- After all sinks acknowledged a snapshot, it is considered **completed**.

Stream alignment



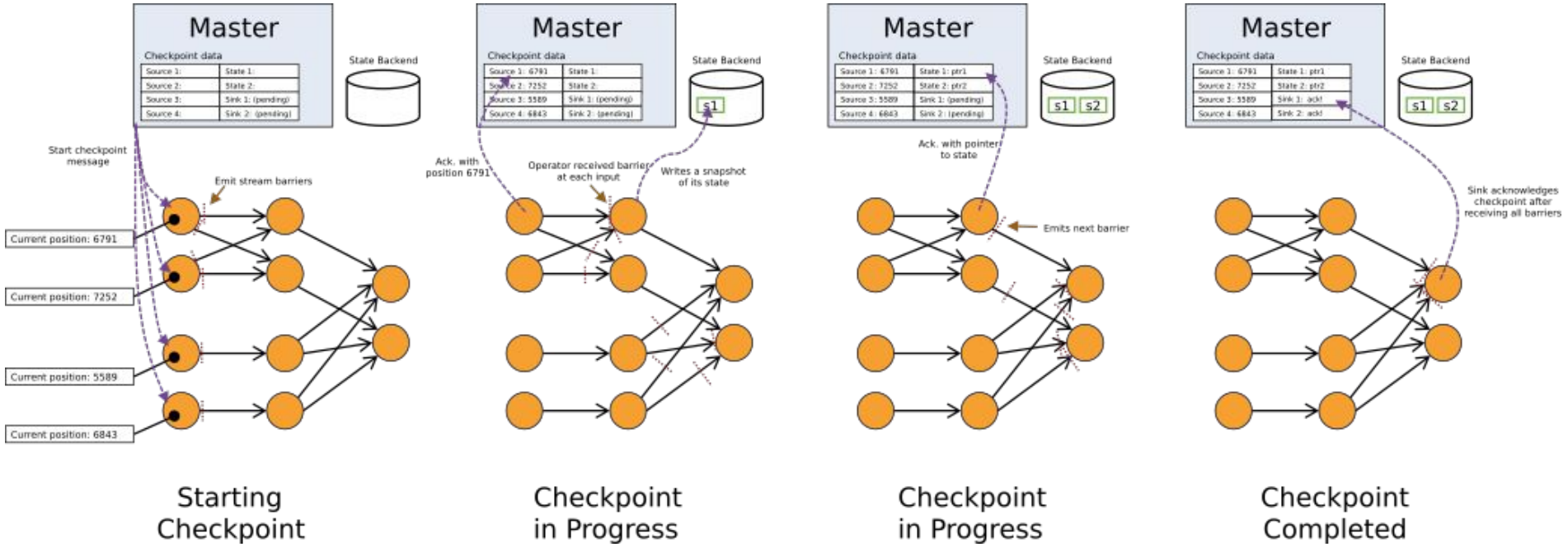
- Operators with *more than one input* need to align on barriers;
- As an input arrives, processing **can't go on** until the counterparts arrive;
- Partial inputs are *buffered* until all the inputs of the checkpoint arrive.

RADICALBIT
Deep, Different, Disruptive.

Stateful operators

- When operators contain any form of state it must be **checkpointed;**
- State can be part of the definition of the operator provided by the user;
- Flink itself has to keep state for certain operations. (e.g. *windows*)

State backend



- Operators save their state when they reach an **aligned state**
- As the state can be large, it can be persisted on a *state backend*
- The state backend should provide a *distributed reliable storage* service

Delivery guarantees

- This mechanisms provide a basis to offer **exactly-once** semantics
- Guarantees **can be configured** and downgraded to at-least-once
  - You have to take care about making your operators idempotent
- This basically means that **alignment is skipped**, all the rest stays valid
  - As alignment is not performed, you can gain in latency on **complex graphs**
  - No changes for massively parallel jobs (e.g. no joins)

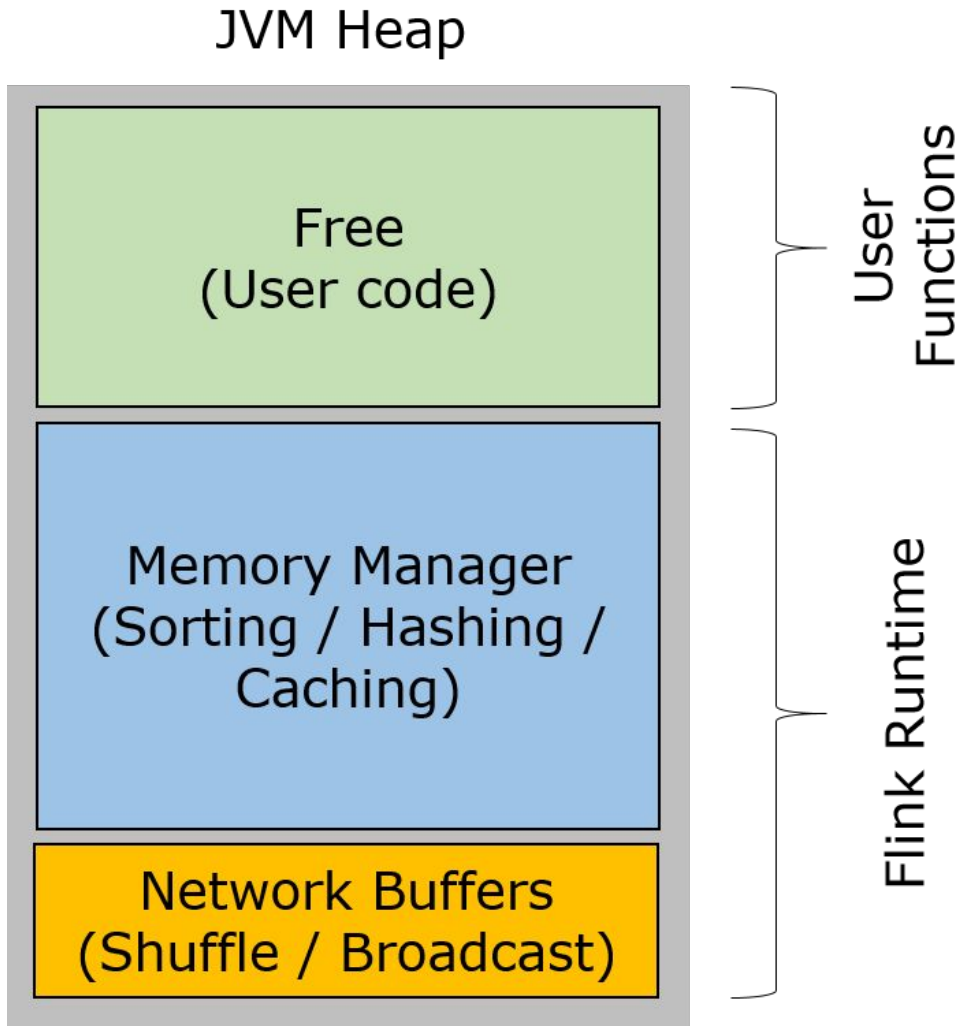
5. Memory Management

Motivation and basic features

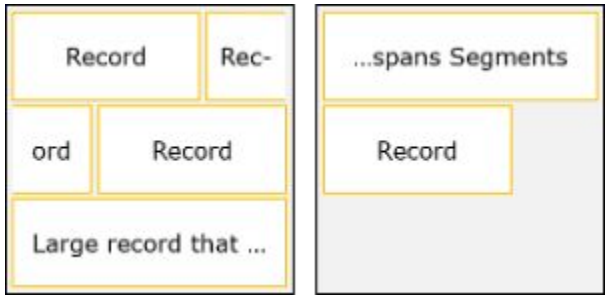
- Computations on big data sets tend to consume **a lot of memory**;
- A *memory manager* has been developed to use it efficiently;
- **Spills in disk** if the heap occupation is reaching a critical point;
- Can be configured to store data **off-heap;**
- *Note: only applied to batch operators*.

Memory heap layout

- The **Free heap** is for user code and task manager data structures
- The **Memory Manager** holds a large pool of pre-allocated memory used by the runtime to store data in serialized form
- **Network buffers** is used to handle data transfer between nodes



Memory segments



- Managed objects are kept **serialized** in buffers called *memory segments*(by default 32 KiBytes)
- Flink keeps track of types and how to compare them
- ***Computation can be performed directly on the serialized form***(i.e. sorting)

Second Part

Flink Applications @Radicalbit

RADICALBIT
Deep, Different, Disruptive.

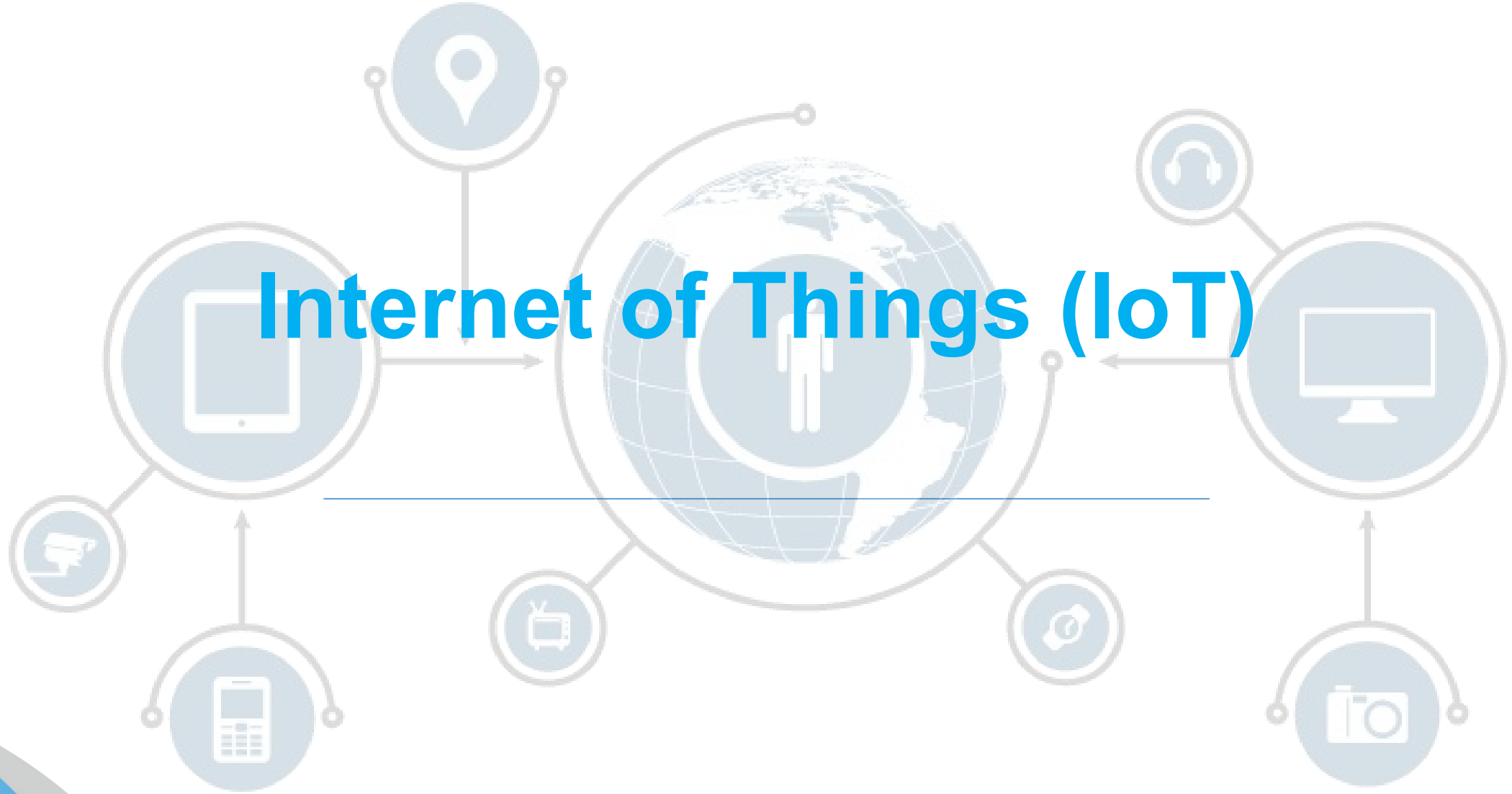
Some of the cool stuff we do @Radicalbit

Internet Of Things

Machine Learning

RADICALBIT
Deep, Different, Disruptive.

Internet of Things (IoT)

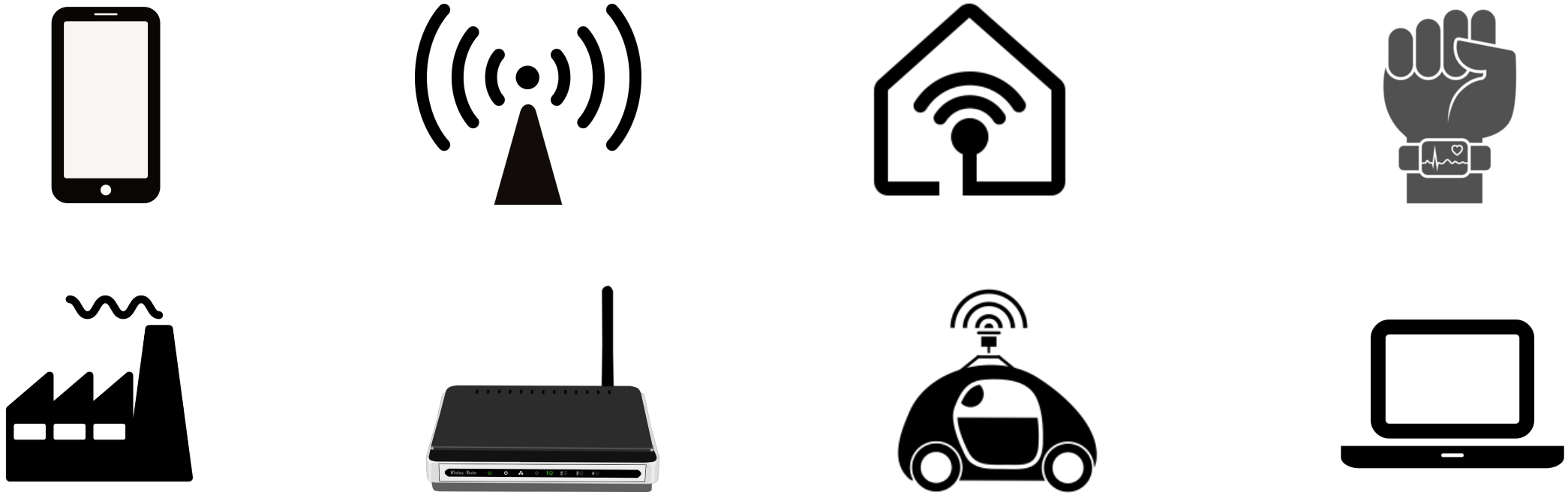


RADICALBIT
Deep, Different, Disruptive.

Recap: what Flink does on streaming-purposes

- Continuous processing of data (on data which is continuously generated)
  - Thus, almost all **Big Data**
- Two things which matter here
  - **State**
  - **Time**
- Flink handles **both**

No more IoT but *even more*: *Internet Of Everything*



*"Internet of Everything will generate **$ 19 Trillion** of Value, all by 2020."*

Forbes, https://goo.gl/WFJ6jR

RADICALBIT
Deep, Different, Disruptive.

Interesting Considerations About IoT

WHAT WE KNOW

Data is continuously generated → **Stream Processing**

Event-Time based processing → events have **timestamps**

WHAT WE WANT

Time Windows analysis

WHAT BECOMES A *PROBLEM*

Events are likely over huge delays / **out-of-order**

What is Event-Time Processing abstraction

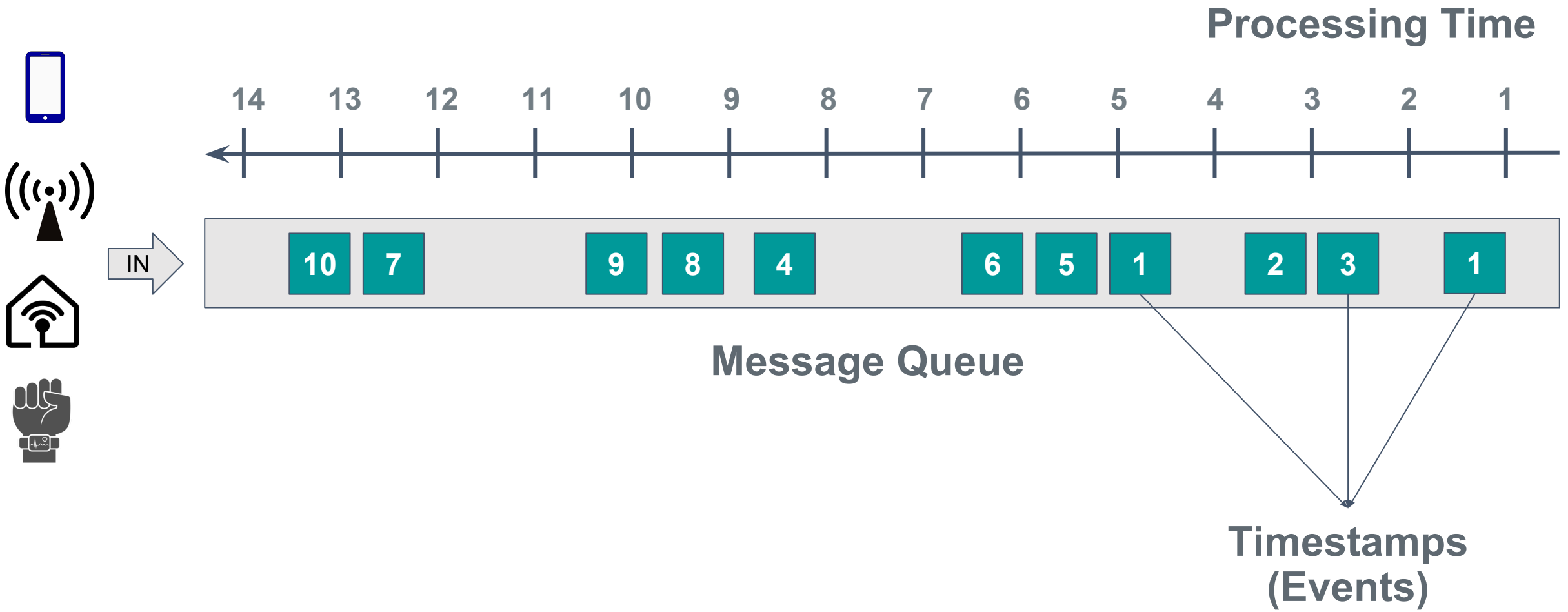
Example by Kostas Tzoumas
@DataArtisans

Event Time

Processing Time

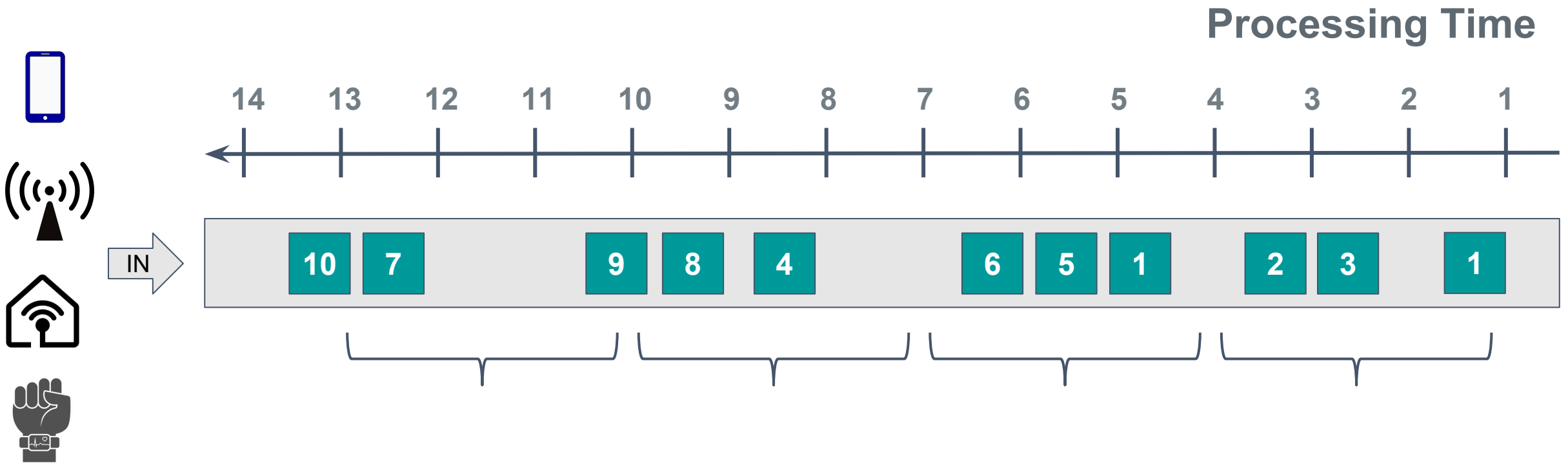
RADICALBIT
Deep, Different, Disruptive.

What is Event-Time Processing abstraction



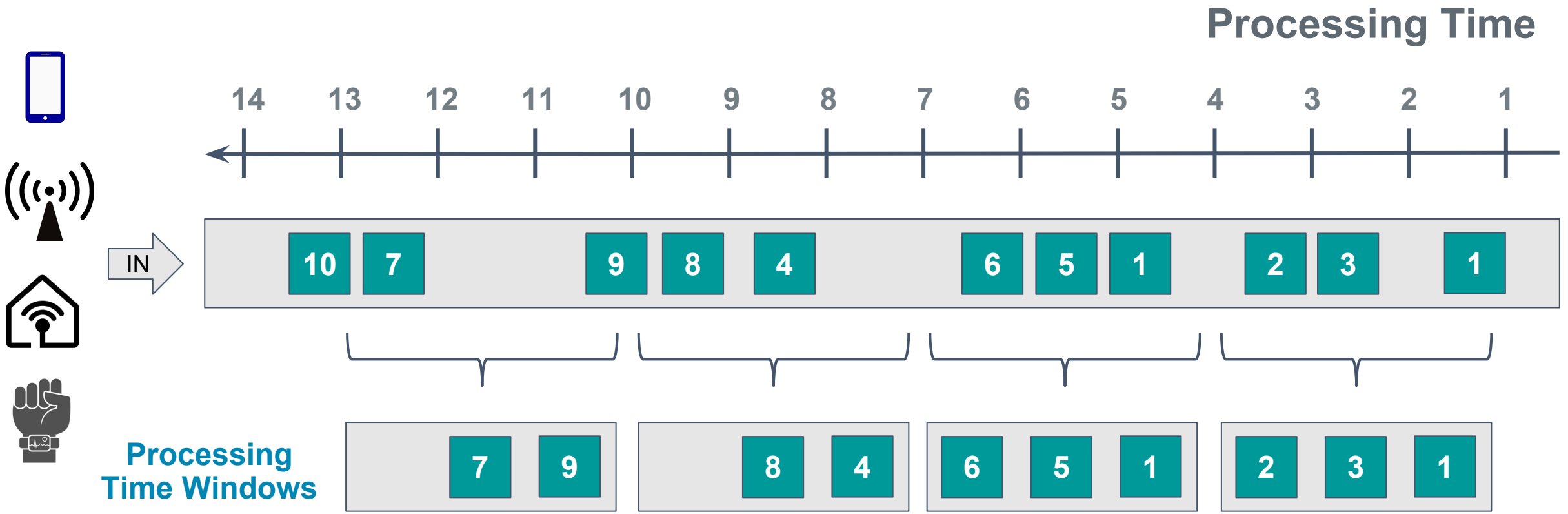
RADICALBIT
Deep, Different, Disruptive.

What is Event-Time Processing abstraction



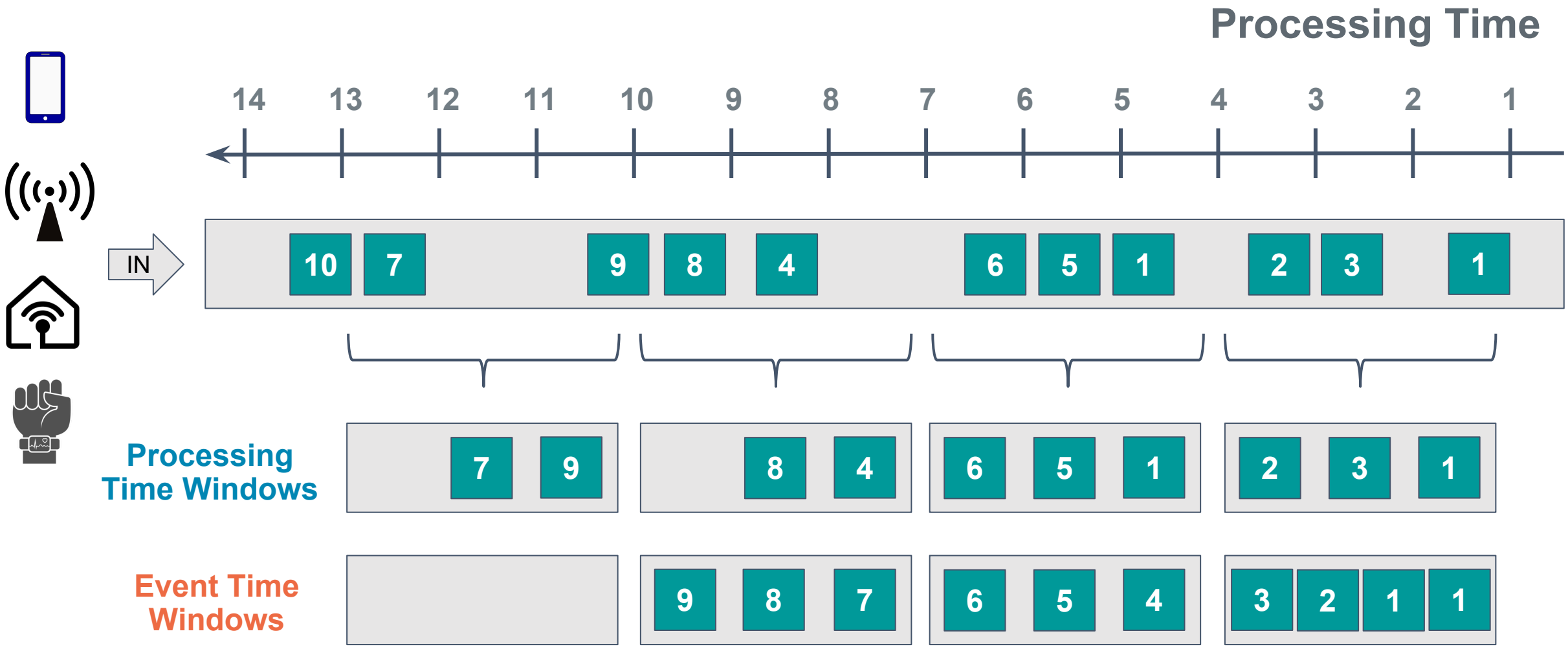
RADICALBIT
Deep, Different, Disruptive.

What is Event-Time Processing abstraction



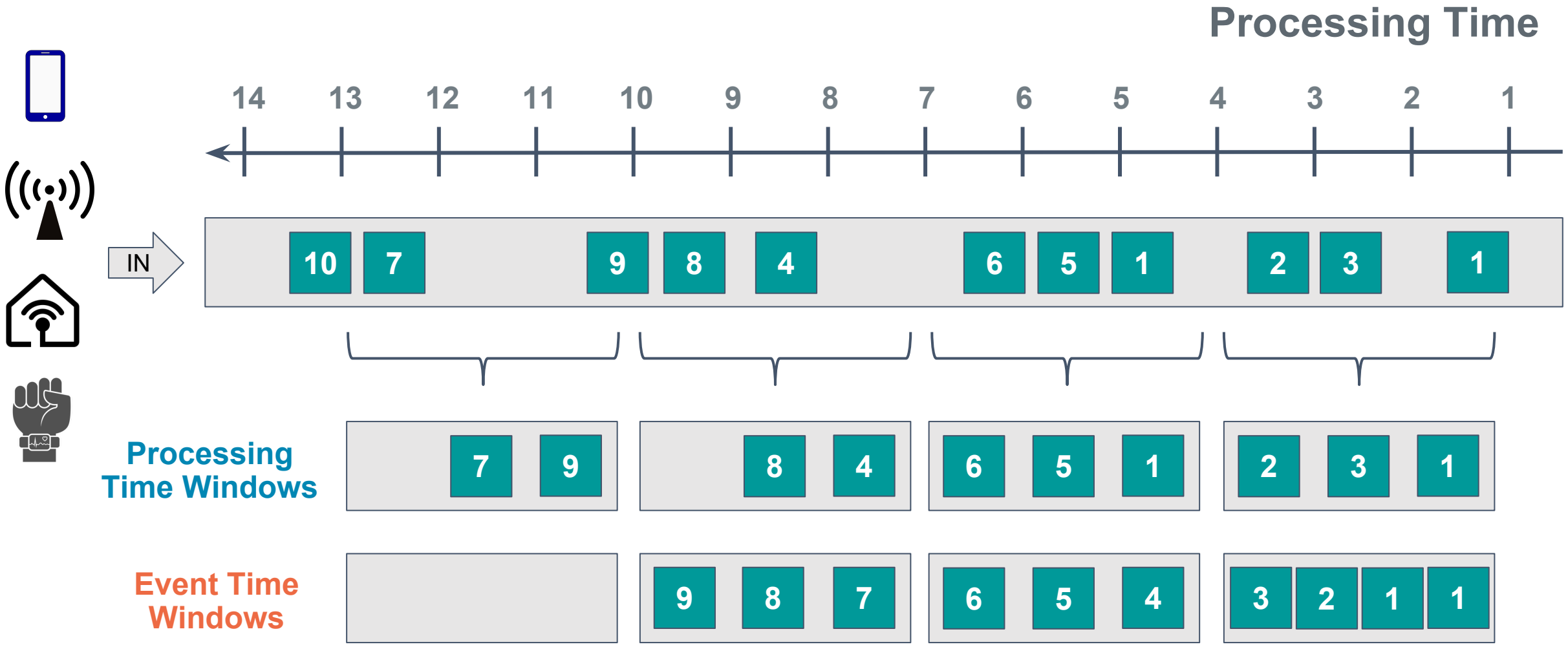
RADICALBIT
Deep, Different, Disruptive.

What is Event-Time Processing abstraction



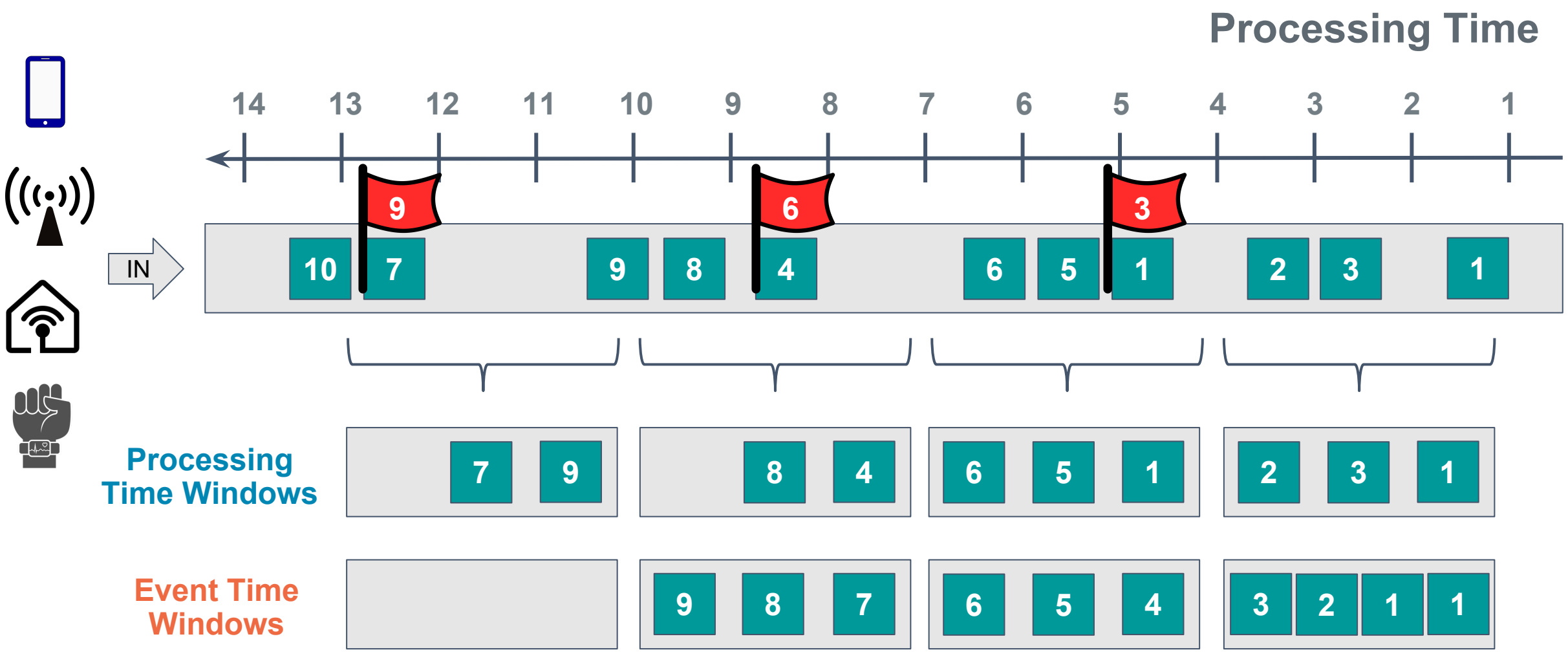
RADICALBIT
Deep, Different, Disruptive.

How can we handle *late events*?



RADICALBIT
Deep, Different, Disruptive.

Watermarks

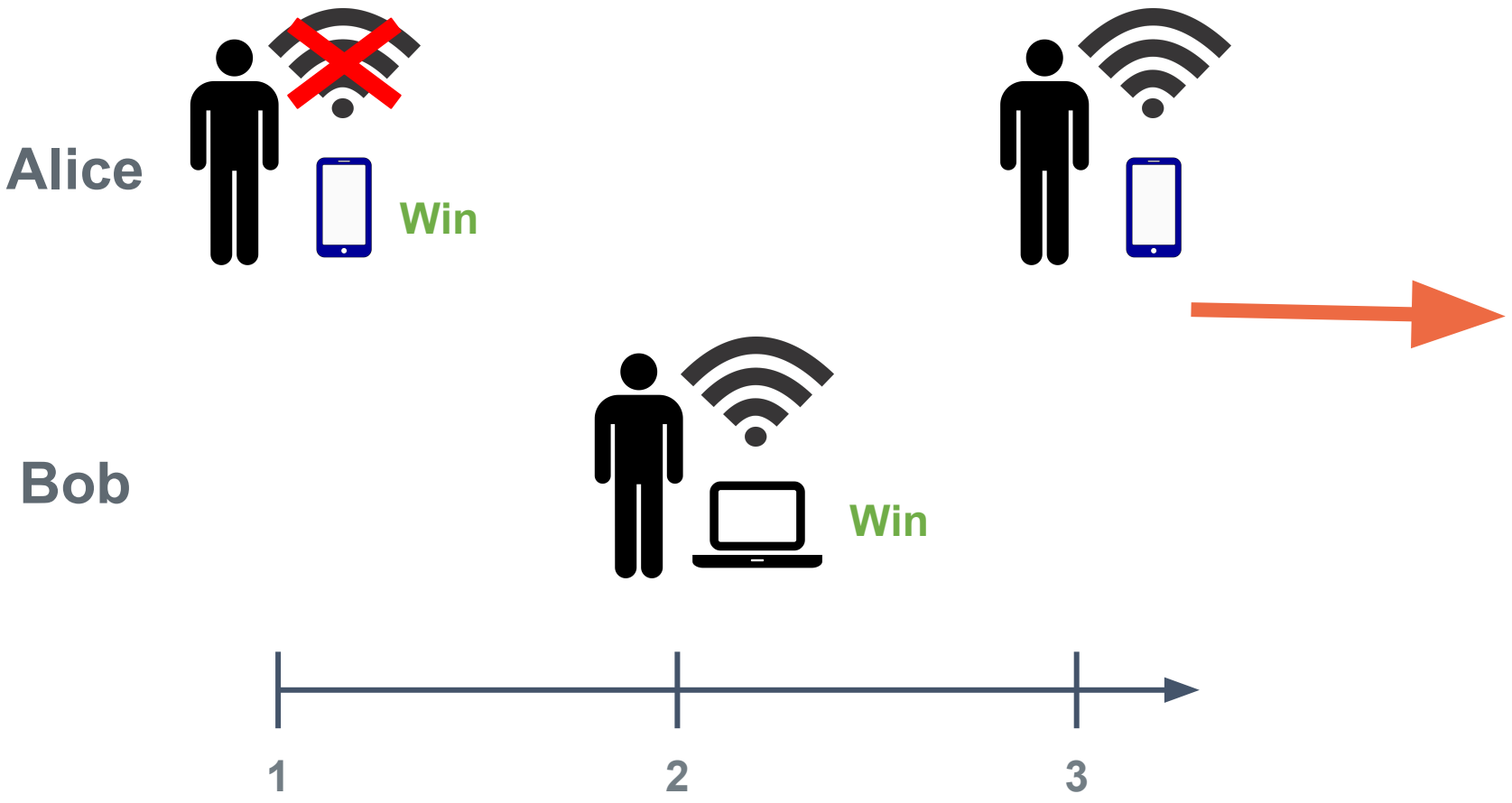


RADICALBIT
Deep, Different, Disruptive.

Time Mismatch → Processing Time v. Event Time

	BIG	SMALL
NETWORK	Failure, Disconnection, Delay	Network Traffic
SYSTEM	Failure, Delay	Clock Time Discrepancy Distributed disposition

IoT needs Event-Time Processing - Game Contest

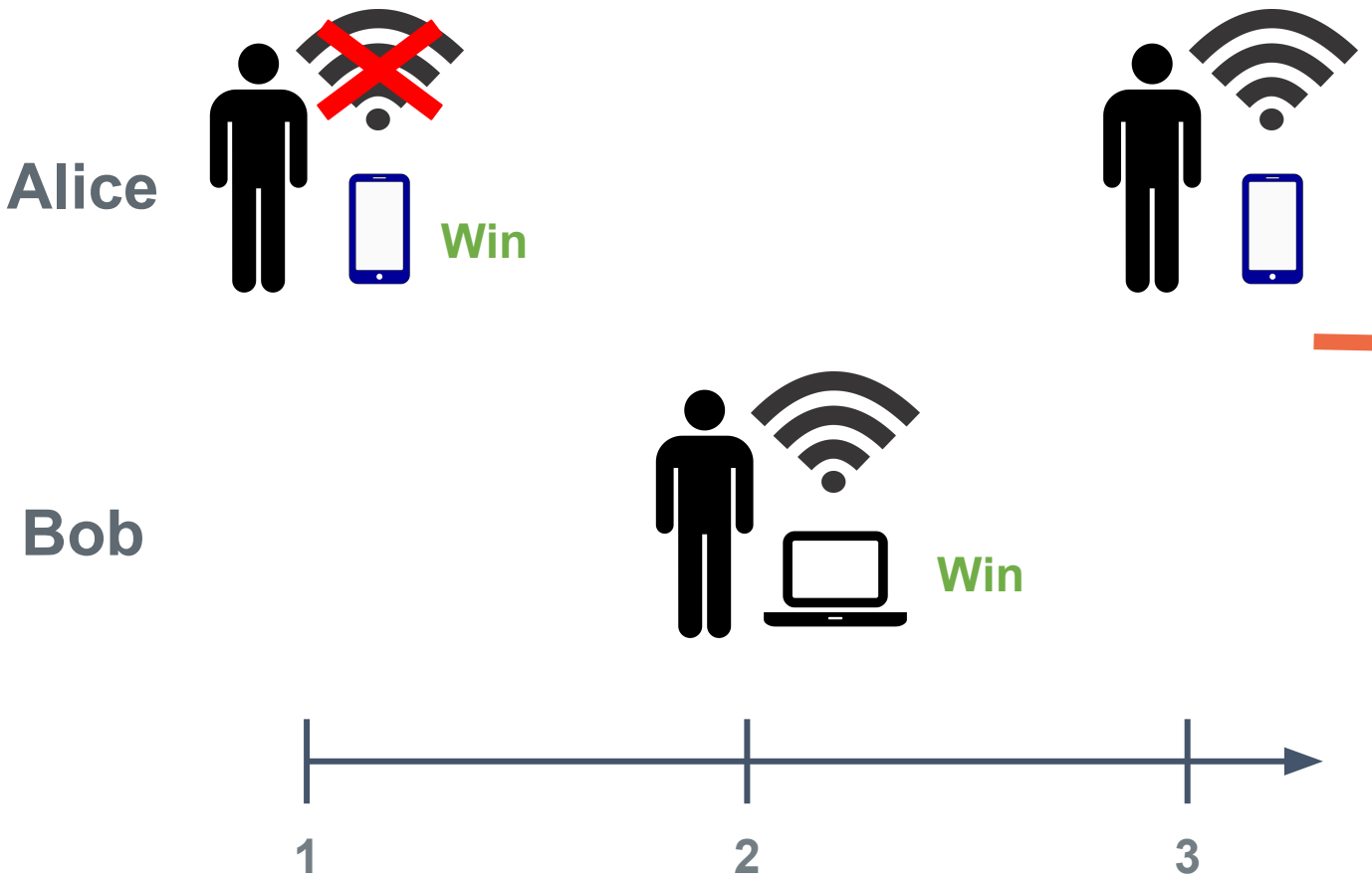


Player	Processing Time (Win)	Event Time (Win)
Alice	3	1
Bob	2	2

Alice Wins!

RADICALBIT
Deep, Different, Disruptive.

IoT needs Event-Time Processing - Game Contest



Player	Processing Time (Win)	Event Time (Win)
Alice	3	1
Bob	2	2

Alice Wins!

Even small mismatches between times will lead to wrong results!

RADICALBIT
Deep, Different, Disruptive.

Keep your hands dirty!

http://data-artisans.com/robust-stream-processing-flink-walkthrough/

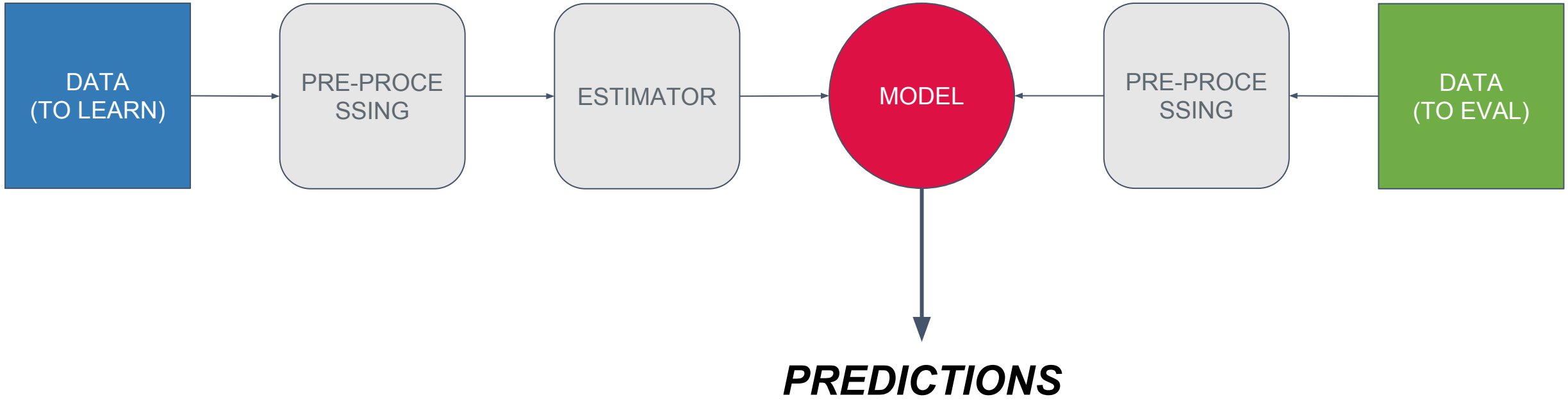
Machine Learning

RADICALBIT
Deep, Different, Disruptive.

A (really) basic Machine Learning Pipeline

RADICALBIT
Deep, Different, Disruptive.

Another basic Machine Learning Pipeline



RADICALBIT
Deep, Different, Disruptive.

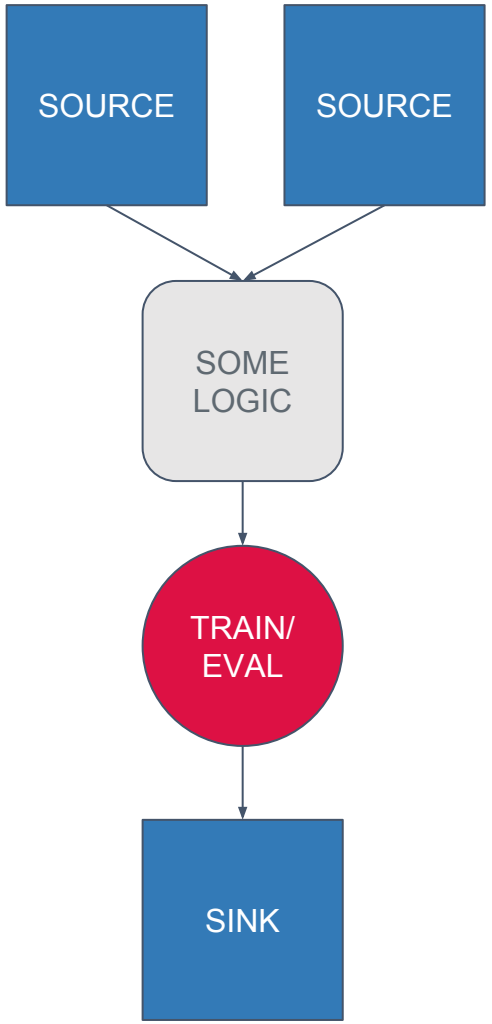
Machine Learning on Apache Flink

We discussed why we use Flink

- *high throughput*, *low latency*, *streaming first*, *fault tolerant* …

Moreover

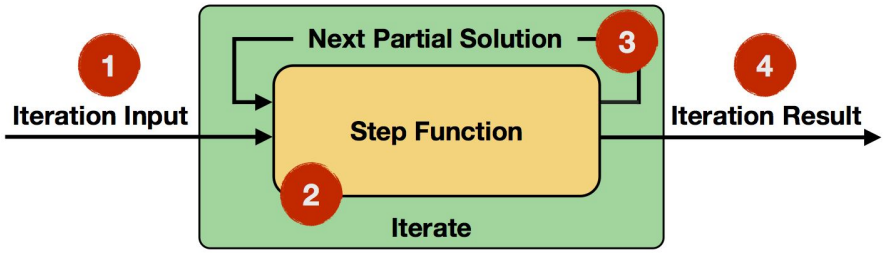
- machine learning pipelines (inspired by *scikit-learn*)
- iterate operator
- broadcasting



RADICALBIT
Deep, Different, Disruptive.

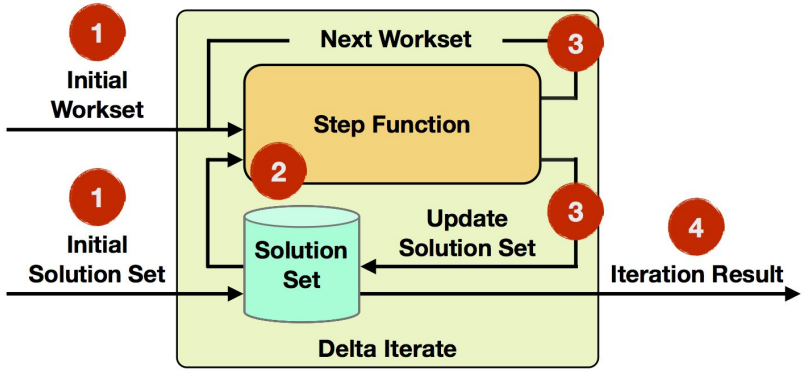
Flink Iterate Operators

Iterations are a typical ML feature, a big Hadoop problem



Flink provides *native* iteration operators

- Bulk Iterate
- Delta Iterate



And this is good because of

- Stateful Operations, No big intermediate states, Limiting Element Scope

RADICALBIT
Deep, Different, Disruptive.

FlinkML

- Wide scope features
  - Supervised (SVM, GD), Unsupervised (K-nearest), Regression, Recommendations
  - Data pre-processing, Data validation

Pros

- Implemented with Flink API Java/Scala - ready Out of The Box

Cons

- Few algorithms, community effort to streaming purposes
- No Model Persistence
- FlinkML is a batch feature

What we're able to do now

	BATCH	STREAMING
TRAIN		
EVAL		 *

*** flink-jpmml**

RADICALBIT
Deep, Different, Disruptive.

Related Work

- Apache SAMOA
- Apache Samsara
- Streamline, Proteus
- …
- Apache Beam

The **GOAL**

Scalable Online Predictive Analysis

WE ARE HIRING !!!

RADICALBIT
Deep, Different, Disruptive.

THANKS!

Riccardo Diomedi

riccardo.diomedi@radicalbit.io
@riccardodiomedi

Andrea Spina

andrea.spina@radicalbit.io
@Spina89

RADICALBIT
Deep, Different, Disruptive.

Credits

- *Apache Flink for IoT*, Aljoscha Krettek - https://www.youtube.com/watch?v=XQnEsB4Ewrk
- *Apache Flink Documentation - https://ci.apache.org/projects/flink/flink-docs-release-1.2/index.html*
- *Stream processing at Bouygues Telecom with Apache Flink*, Kostas Tzoumas - http://data-artisans.com/flink-at-bouygues-html/
- Robust Stream Processing with Apache Flink, Michael Winters - http://data-artisans.com/robust-stream-processing-flink-walkthrough
- Data Intensive Applications with Apache Flink, Simone Robutti - https://www.youtube.com/watch?v=ABQoQ3-pBI4&feature=youtu.be
- Machine Learning with Apache Flink, Till Rohmann - http://www.slideshare.net/tillrohrmann/machine-learning-with-apache-flink
- Streamline - https://streamline.sics.se/node/26

Images

- *IoT*, introduction slide - http://www.techeconomy.it/wp-content/uploads/2014/07/internet-of-things-IoT.png