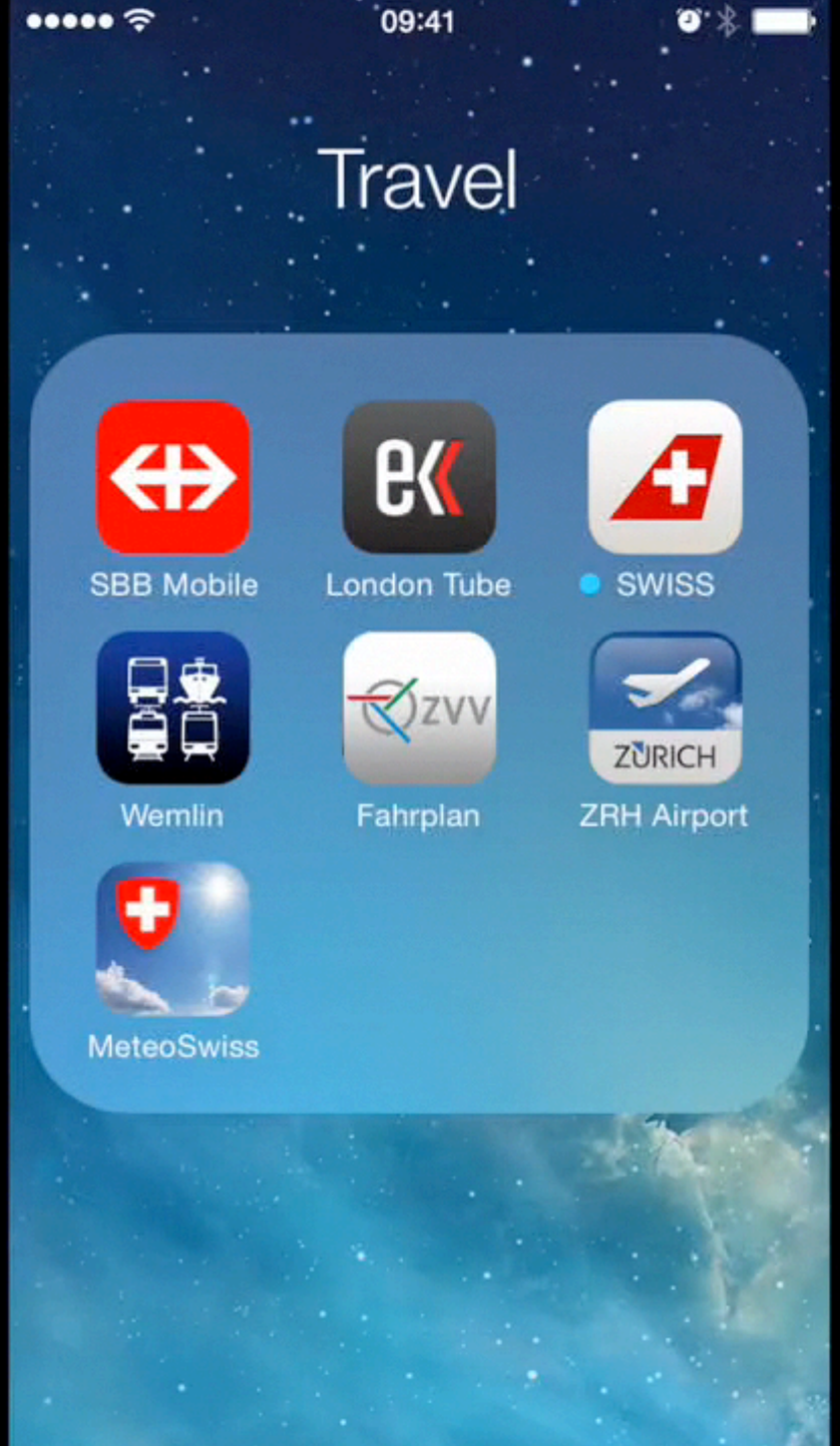
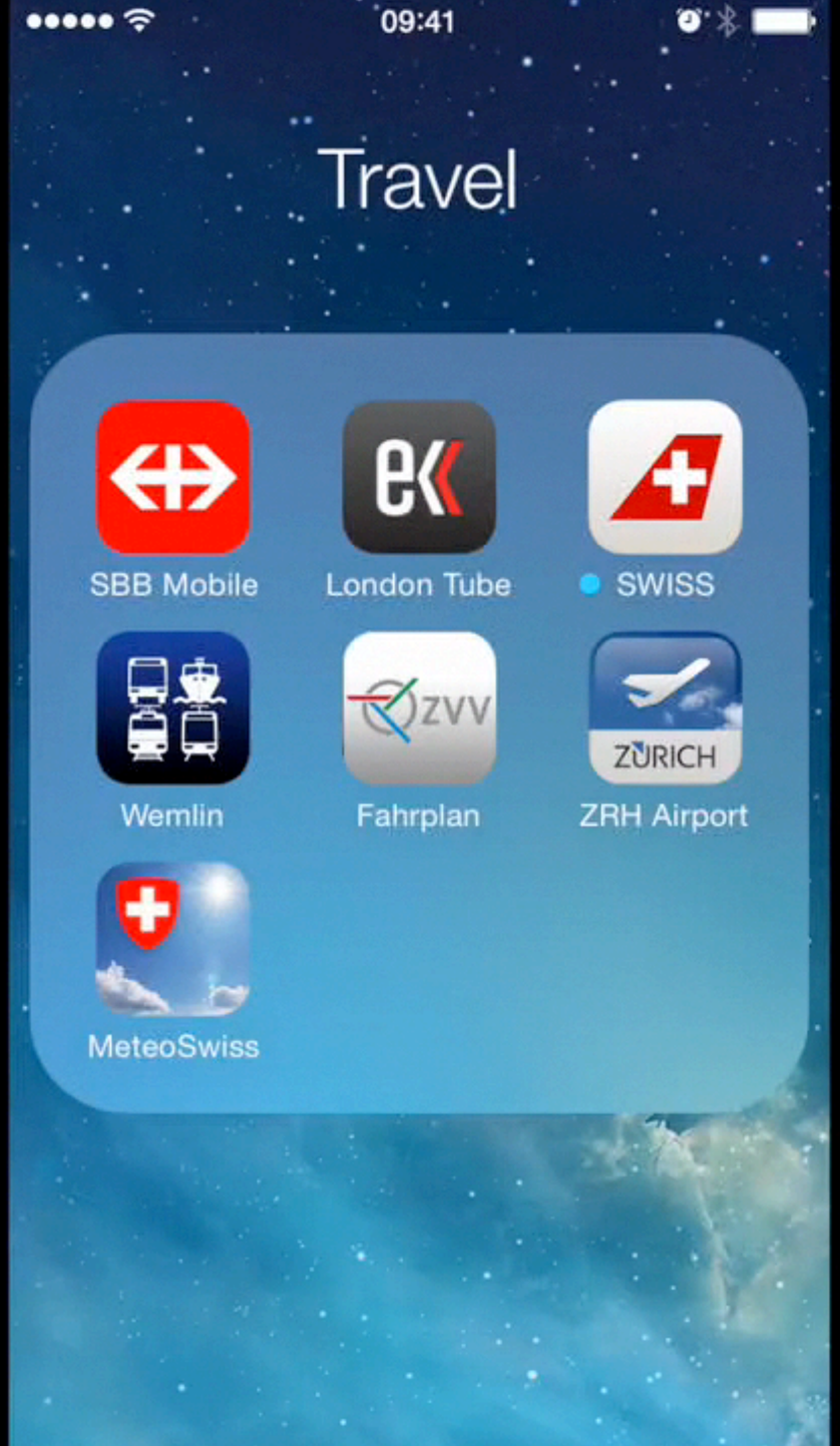
The Architecture of **Wemlin Hub**

Ognen Ivanovski, Netcetera

Jug.ch ‘15

Wemlin


09:41
Travel
SBB Mobile
London Tube
SWISS
Wemlin
Fahrplan
ZRH Airport
MeteoSwiss

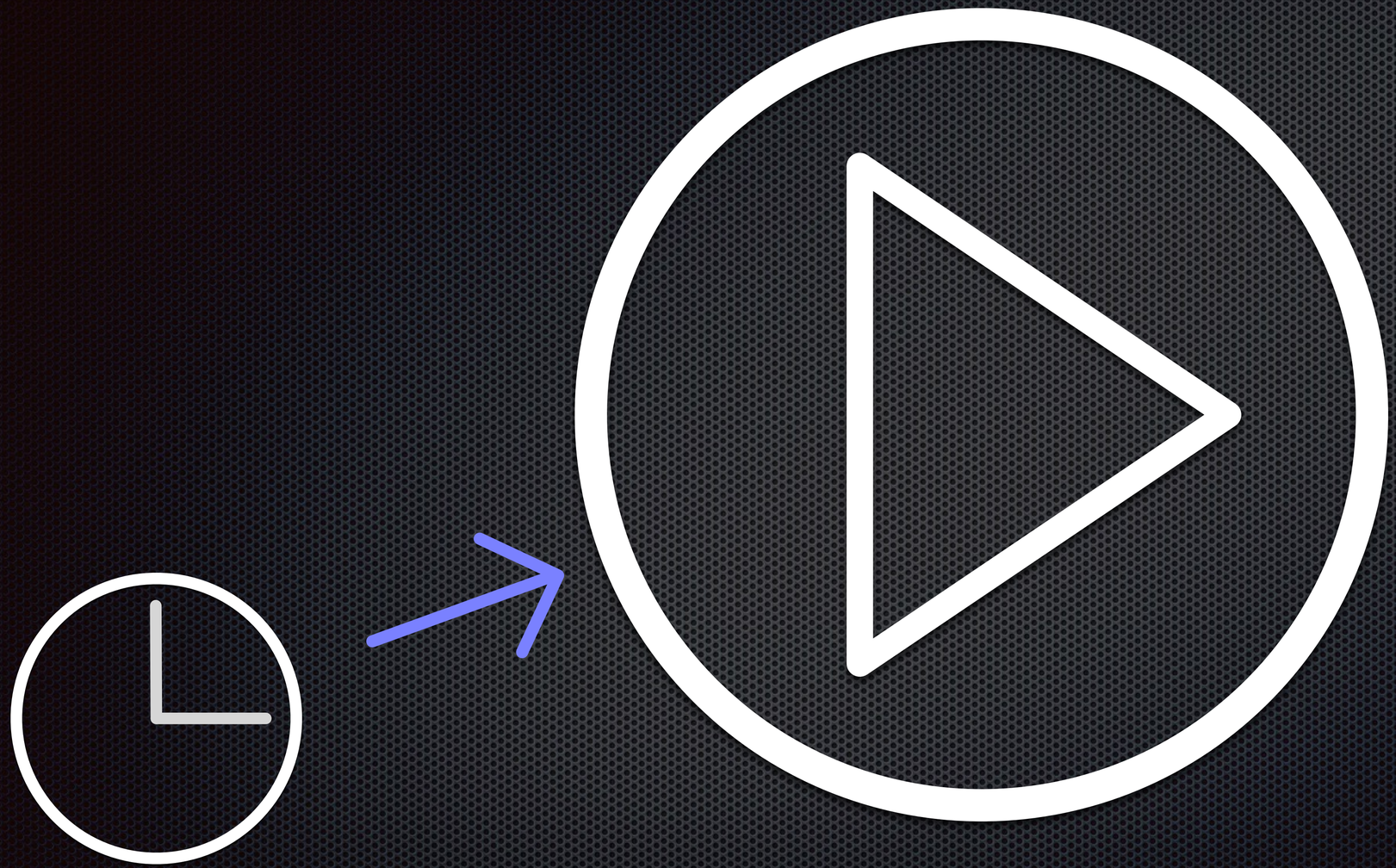

09:41
Travel
SBB Mobile
London Tube
SWISS
Wemlin
Fahrplan
ZRH Airport
MeteoSwiss

Data

Data

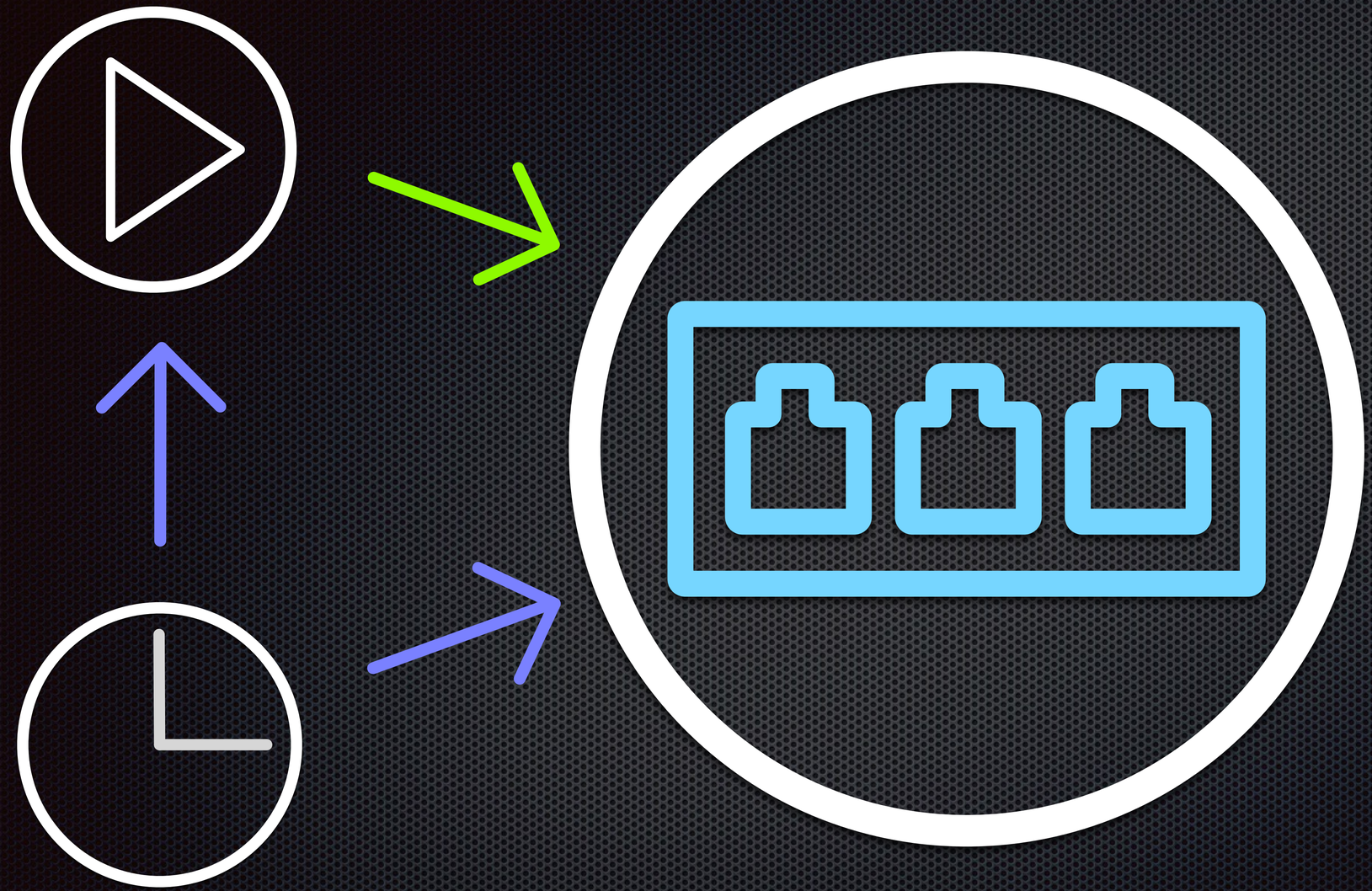
Planning Software

Planning Software

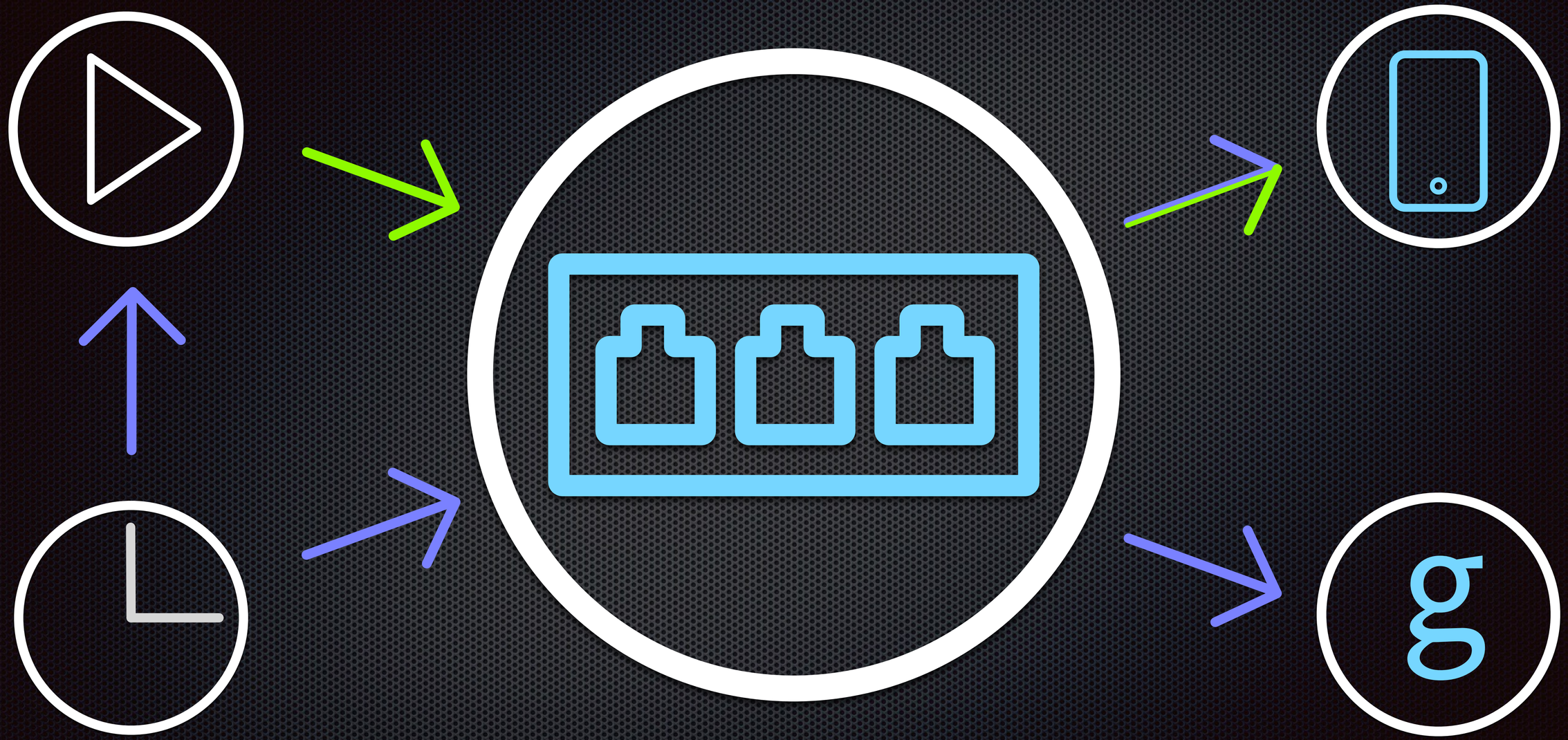


AVCS


AVCS



Wemlin **Hub**



Wemlin **Hub**

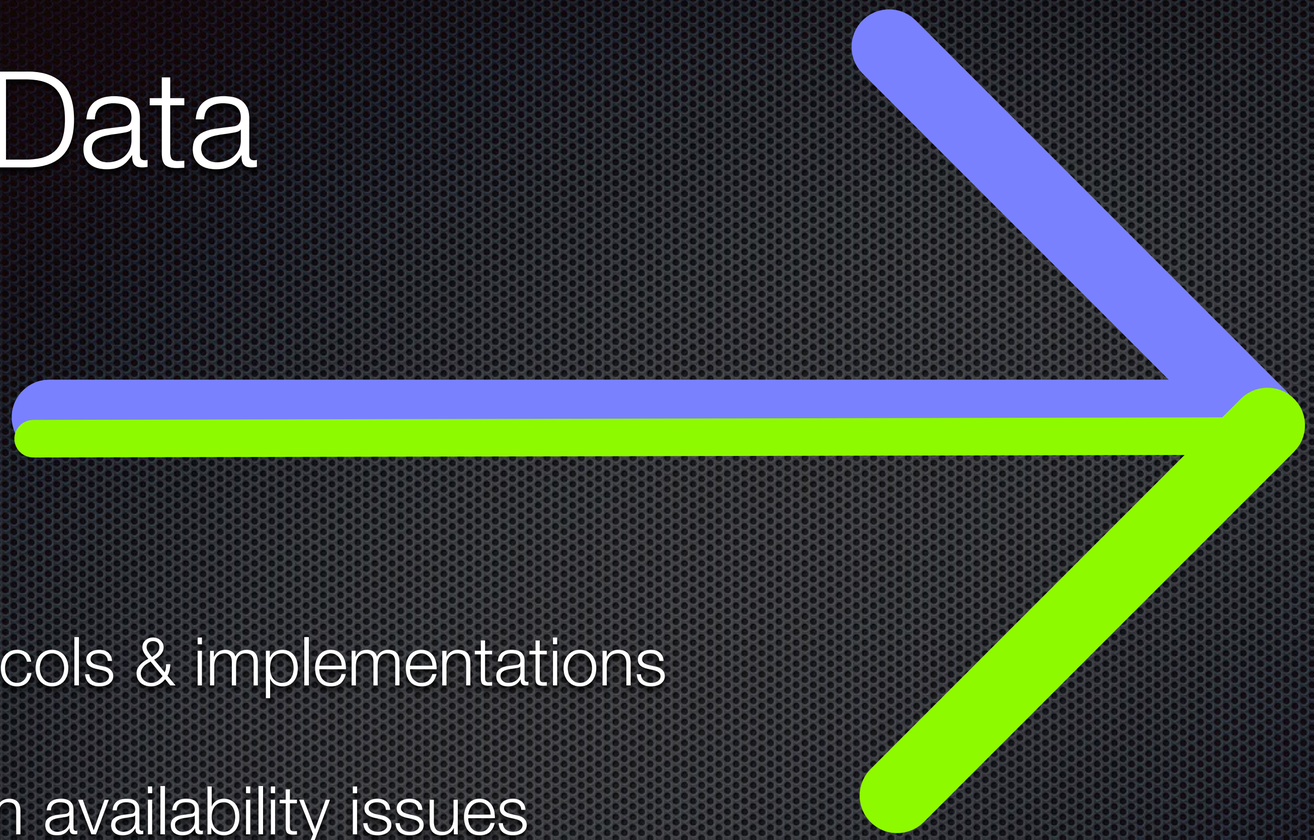
Plan Data

- transport schedule (plan) over certain time
- package in nature
- Formats: HAFAS, GTFS, VDV 453 REF, VDV 454 REF, REST

RT Streams

- updates on planned data
- stream in nature
- Formats: GTFS RT, VDV-453, VDV-454, REST

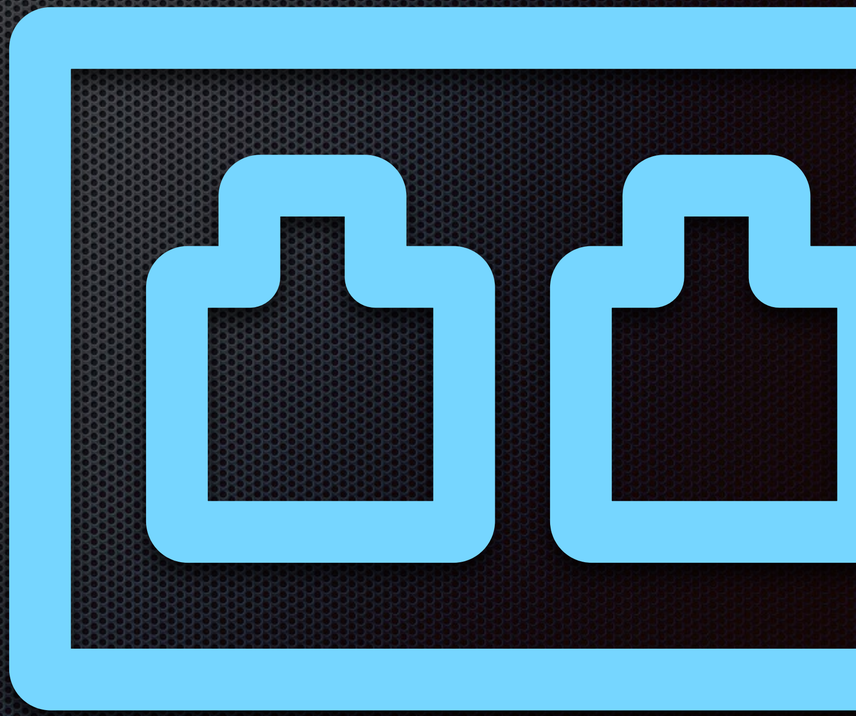
Getting Data



- different protocols & implementations
- remote system availability issues

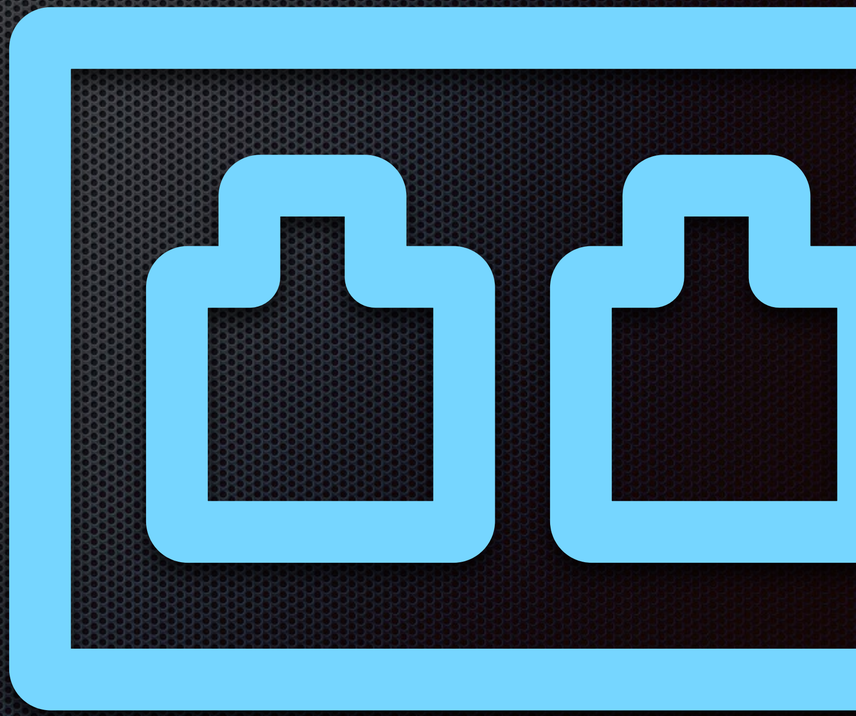
Figuring things out

- different sources, no referential guarantees
- every data source is different
- data quality issues
  - Example: station refeferences
  - Example: trip references

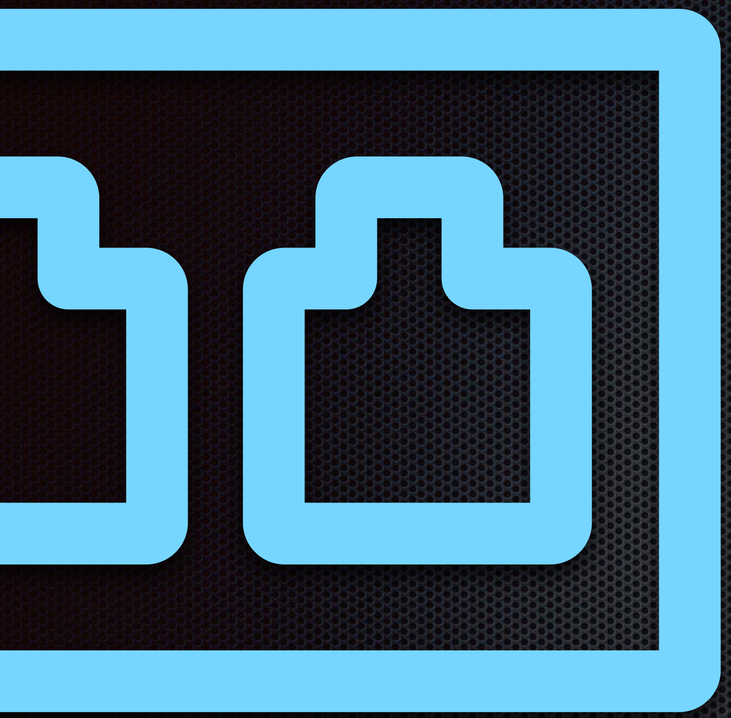


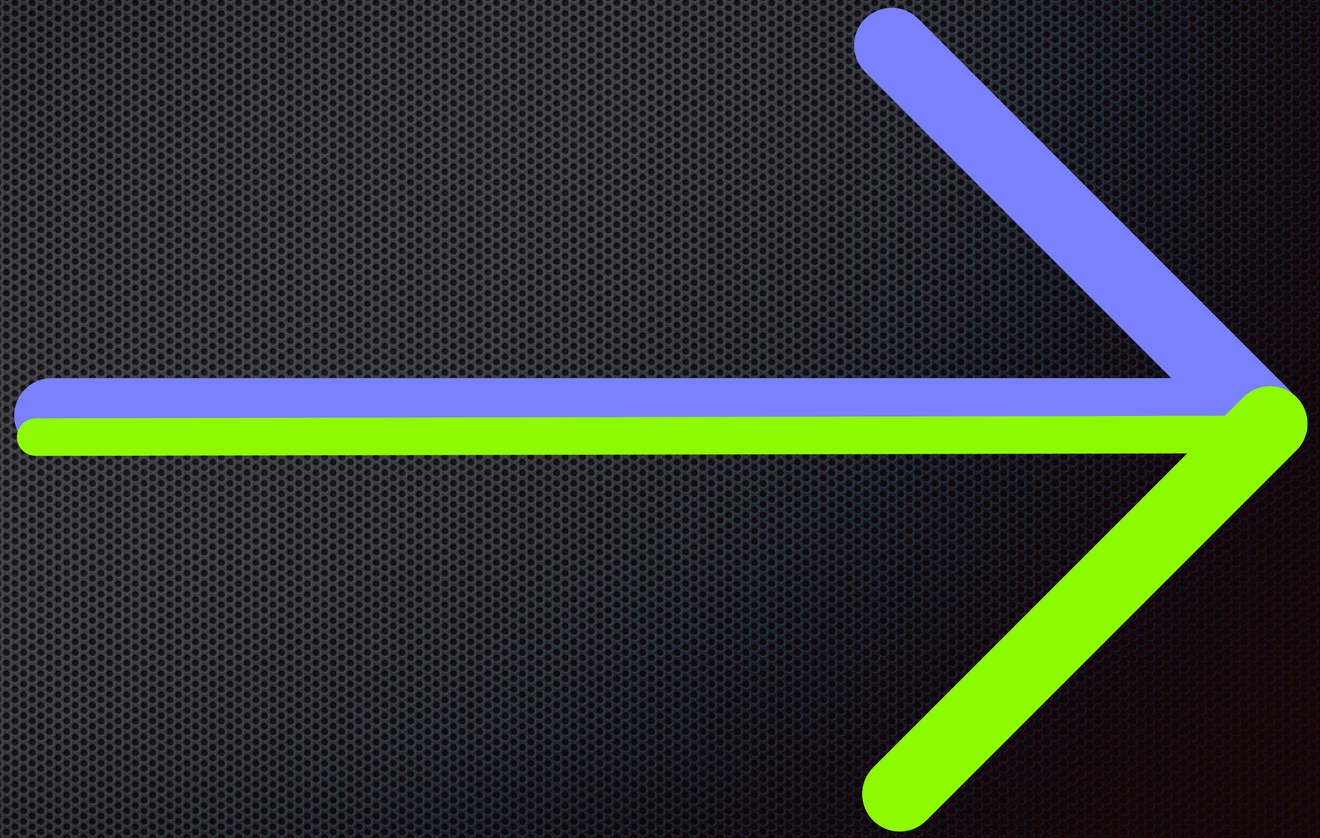
Data Enrichment

- line colors
- location
- station metadata (e.g. which lines stop there, which stations are near by)



Serve many users





Key Concerns

RT

low latency

throughput

available

excllent failure management

Batch

throughput

Key Concerns

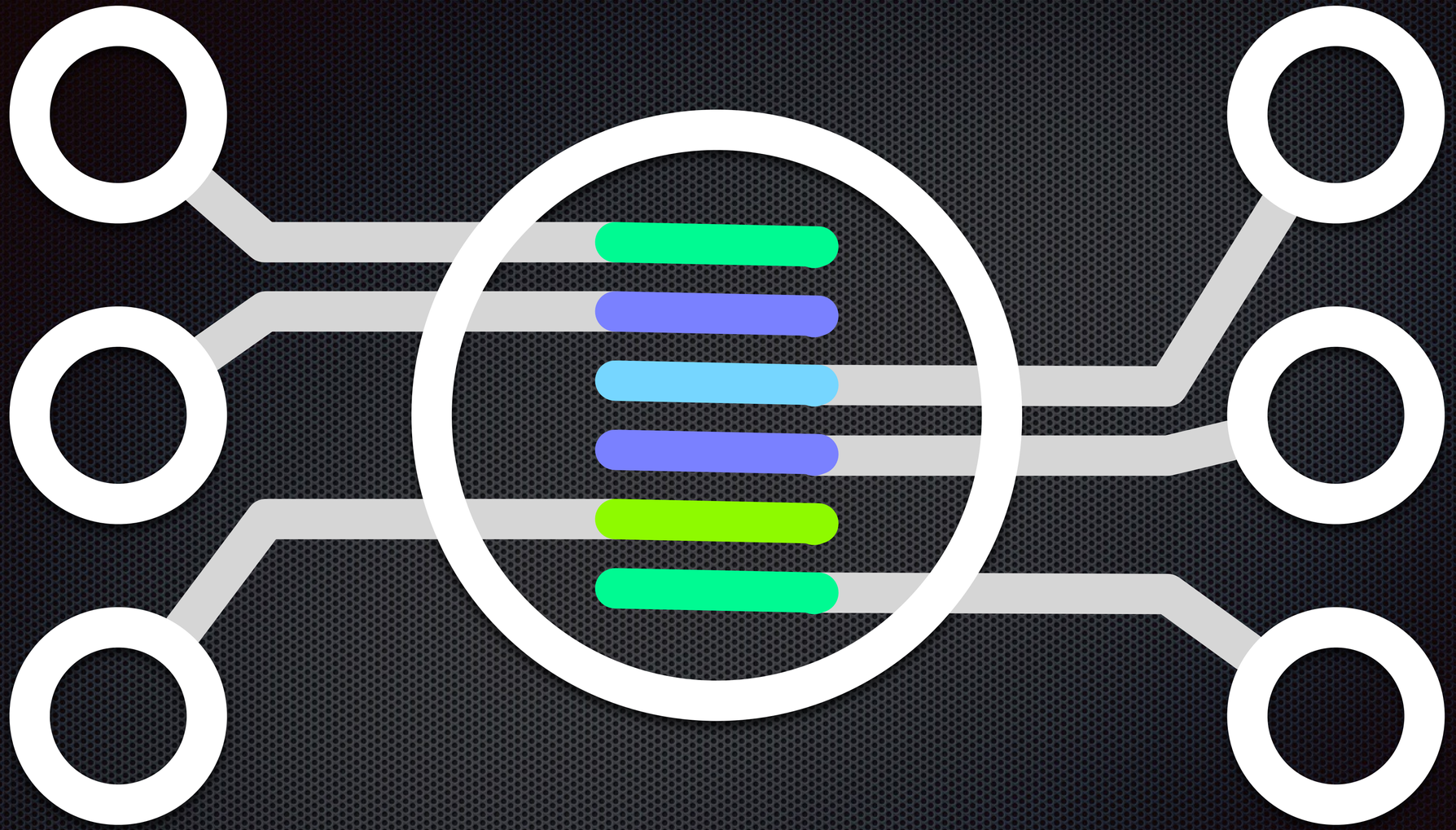
processing programs the same for both batch and RT data

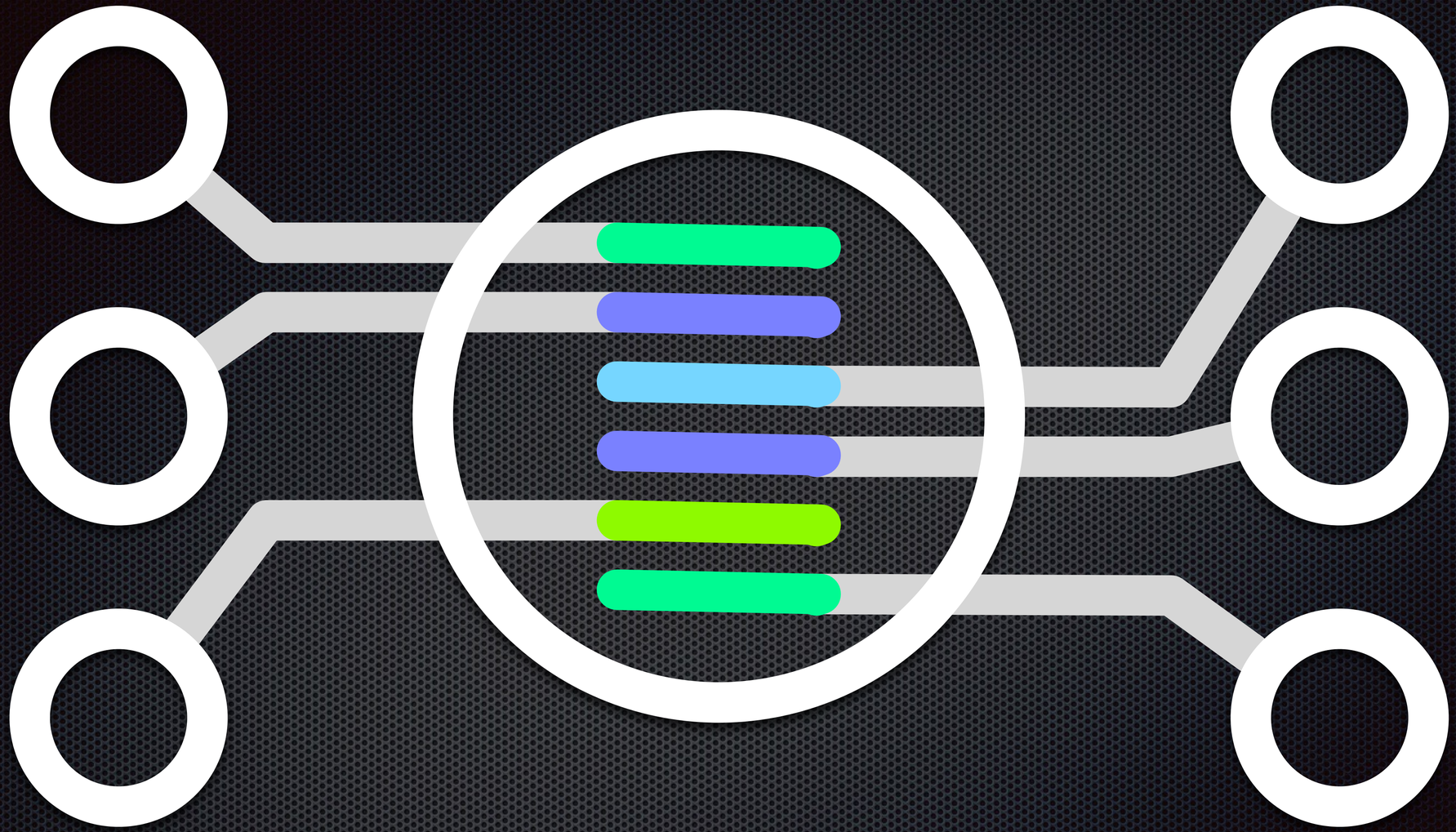
Key Concerns

fast reaction to load changes

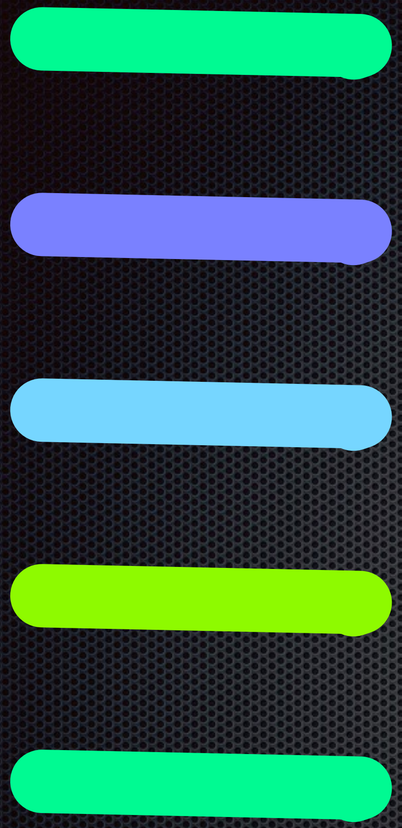
Architecture

- Constraints
- decisions hard to change





Microkernel Architectural Style



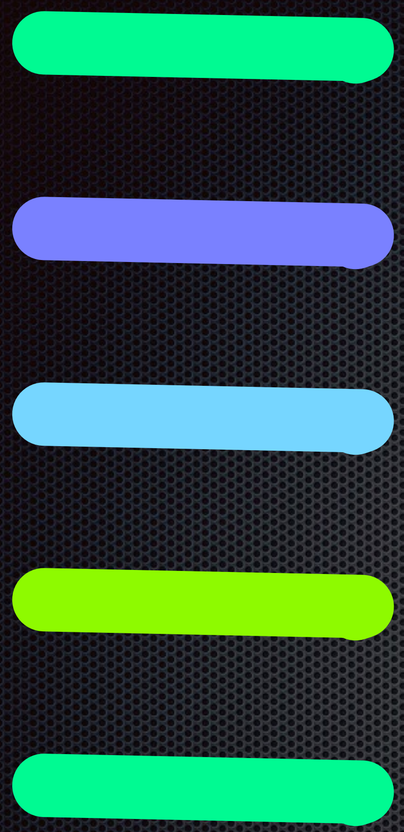
tap

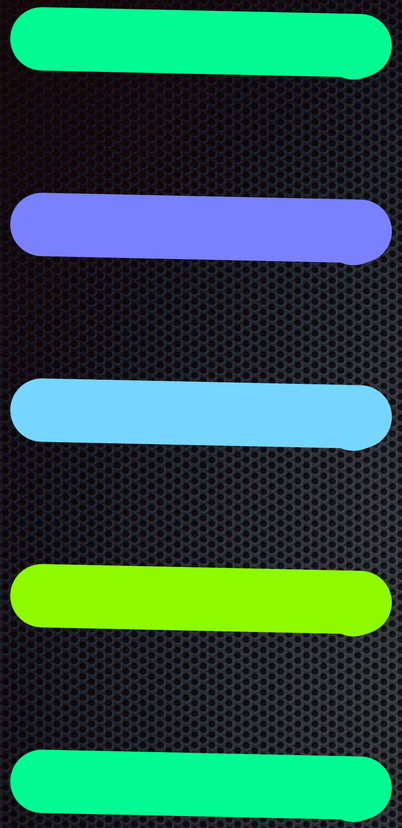
filter

transform

aggregate

sink

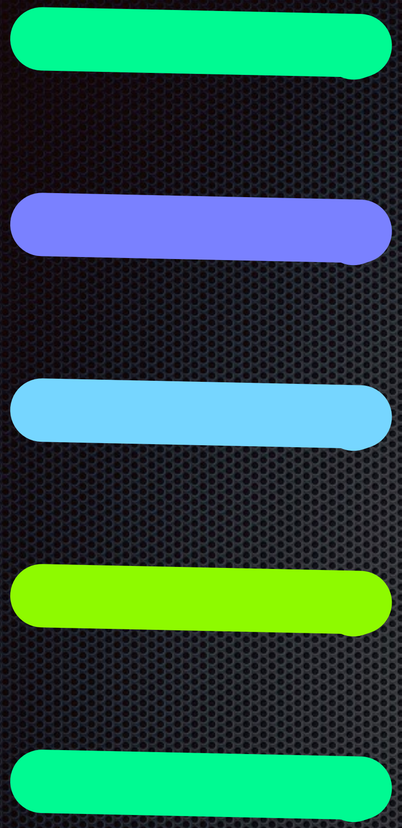

tap
sink



filter

```
public interface Filter {

    boolean accept(Object obj);

}
```

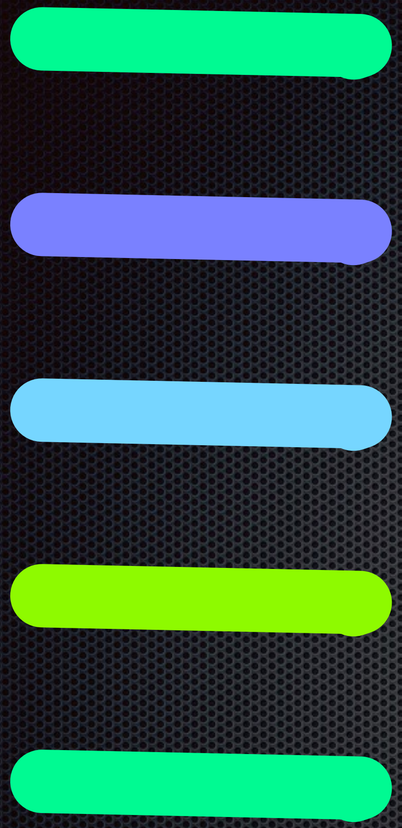
stateless



transform

```
public interface Transformer {

  Object transform(Object original);

}
```

stateless



aggregate

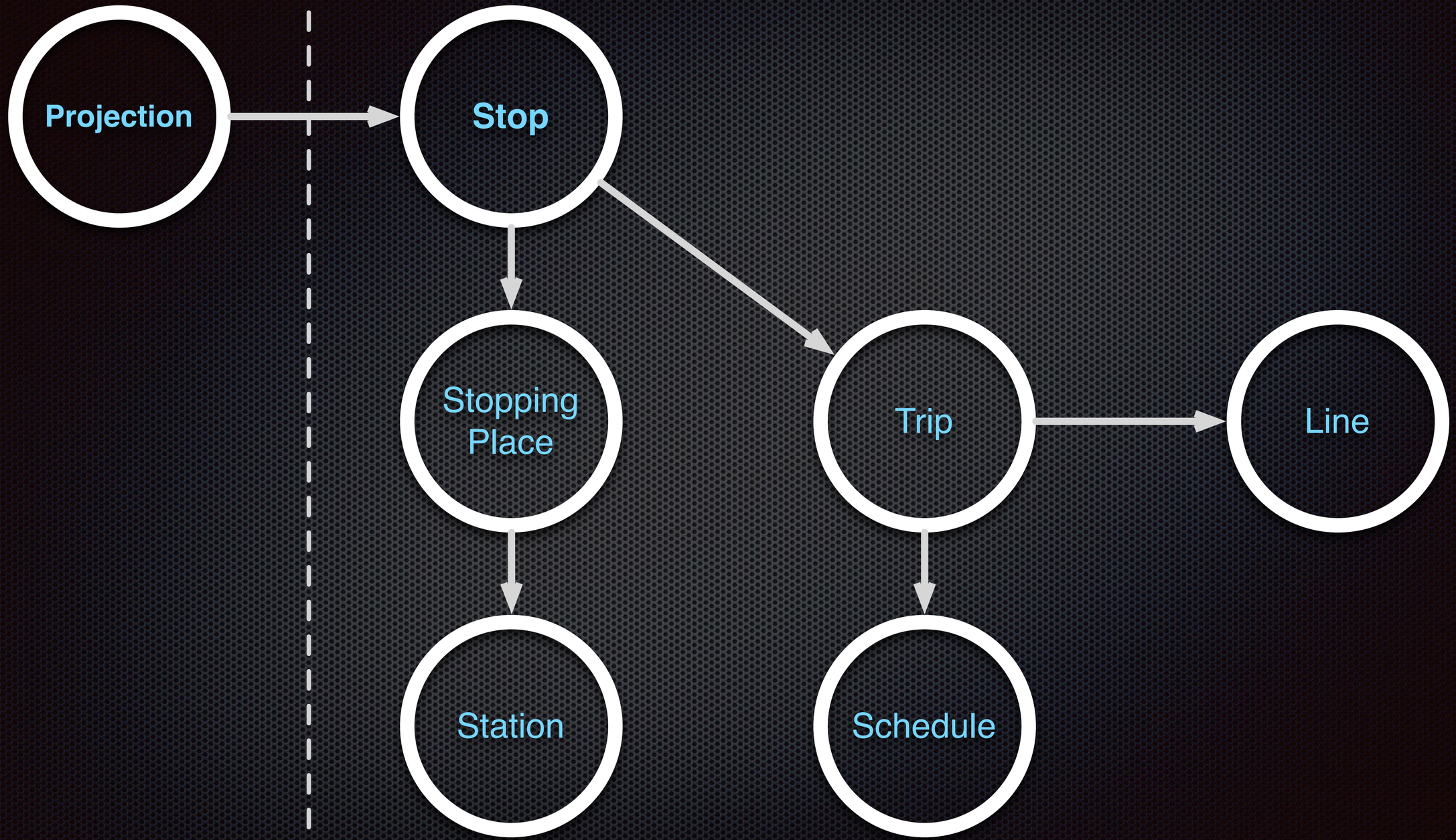
```
public interface Aggregator {

    Optional<?> aggregate(Object obj);

}
```

Composition

```
pipeElement = PipelineBuilder.from(gtfsInputJunction())
                .transform(new StoppingPlaceResolver())
                .filter(new InvalidStopsFilter())
                .transform(new UnresolvedLineVehicleTypeAdder())
                .transform(new UnresolvedLineNameAdjuster())
                .transform(new LineResolver())
                .transform(
                  new LineColorsEnricher(...))
                .aggregate(new CacheAggregator(cache()))
                .to(nullSink());
```

Model

- **immutable**: model objects are values (changing means a new object)
- **algebraic**: each object identity is defined by it's contents.
- **pure**: in the sense of no external dependencies


Projection
Stop
Stopping Place
Trip
Line
Station
Schedule

Architecture

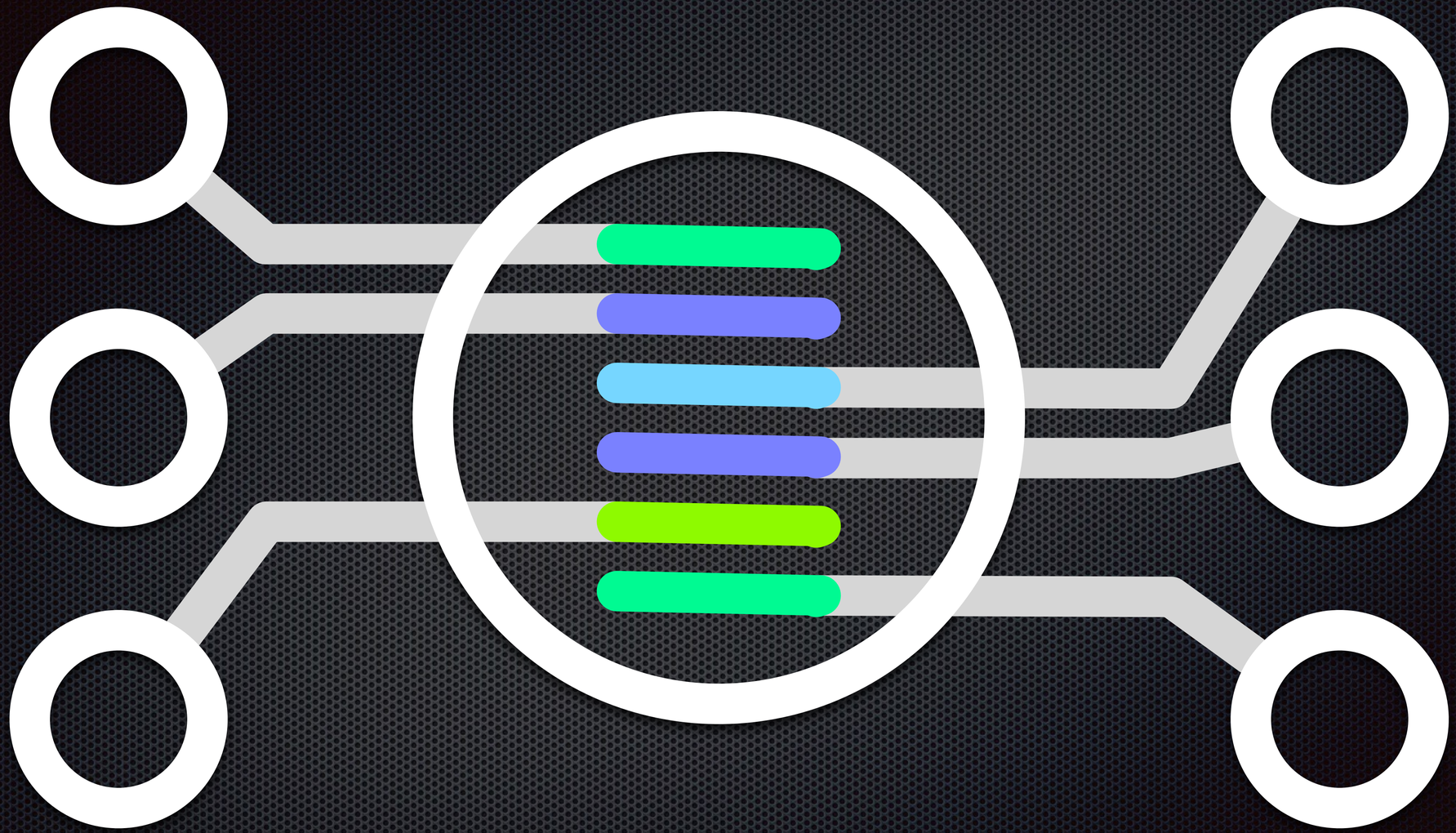
Model

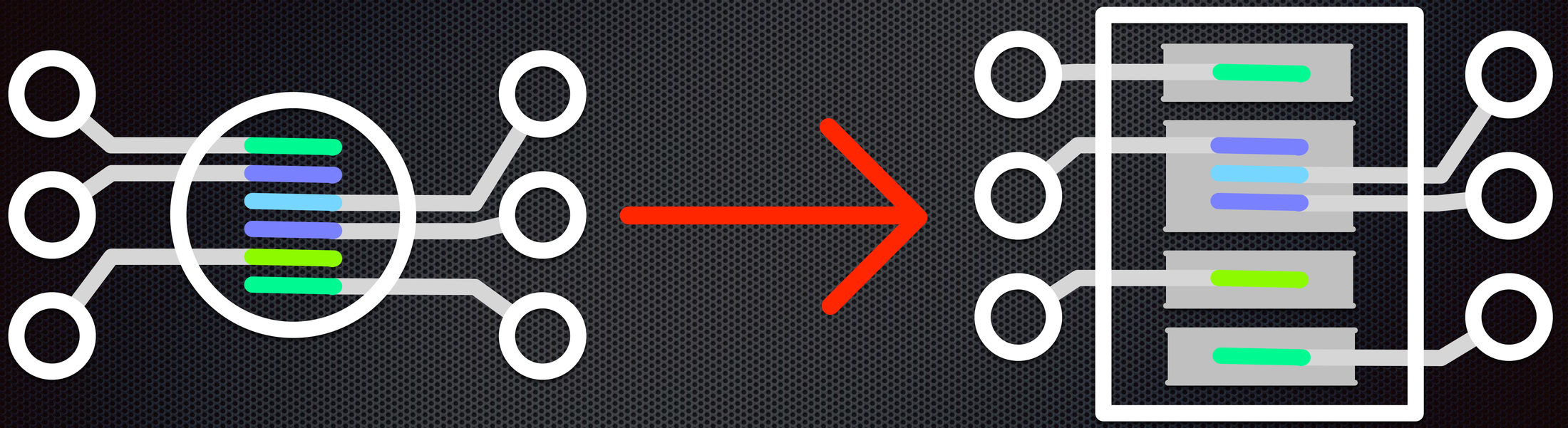
- Pure Java
- Immutable
- Algebraic
- Inverse References

Pipeline

- Functional Microkernel
  - Filter (stateless, pure function)
  - Transformer (stateless, pure function)
  - Aggregator (stateful, function)
  - Sink (consumer) / Tap (producer)

```
pipeElement = PipelineBuilder.from(gtfsInputJunction())
                .transform(new StoppingPlaceResolver())
                .filter(new InvalidStopsFilter())
                .transform(new UnresolvedLineVehicleTypeAdder())
                .transform(new UnresolvedLineNameAdjuster())
                .transform(new LineResolver())
                .transform(
                  new LineColorsEnricher(...))
                .aggregate(new CacheAggregator(cache()))
                .to(nullSink());
```





pipeline “compile” assembly

Assemblies

- Spring Integration based
- Thread-pool based
- Fork-join based

Segments

- parallel / serial
- separated by queues
- segmentation based on the used interfaces

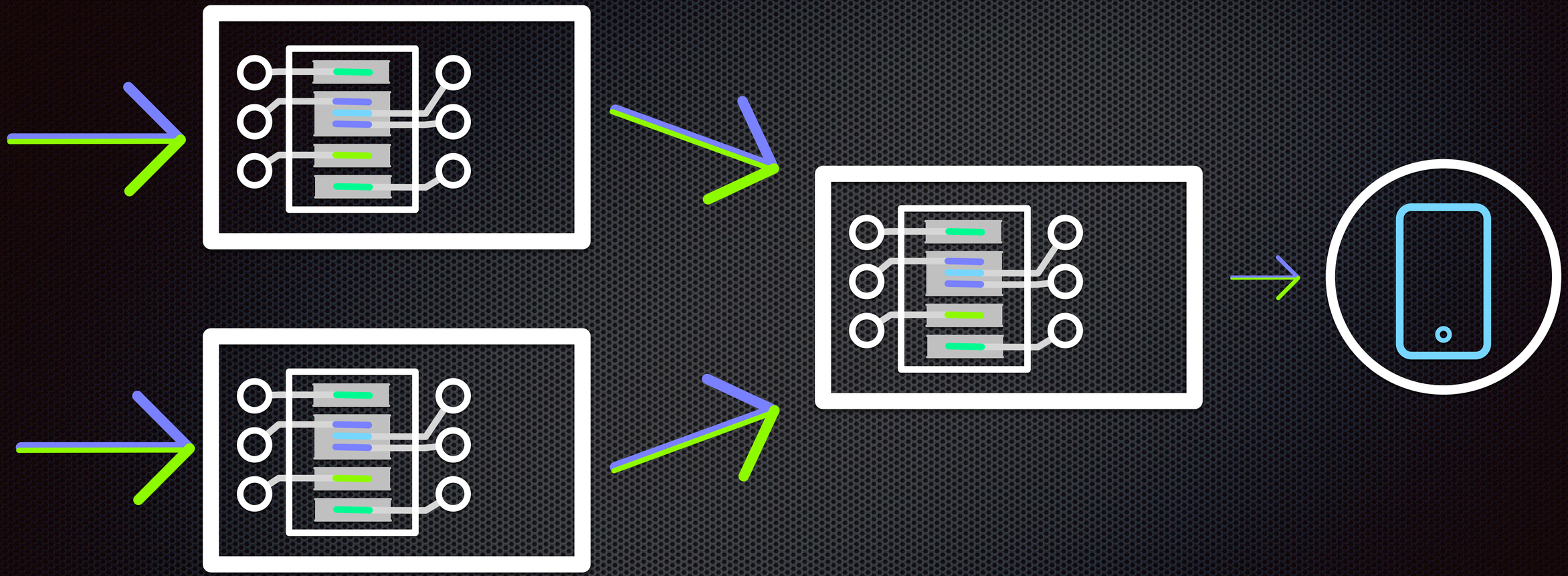
Fork-Join

- to be effective, one must batch
- batching introduces latency (in RT)
- trick
- batch mode
- RT mode

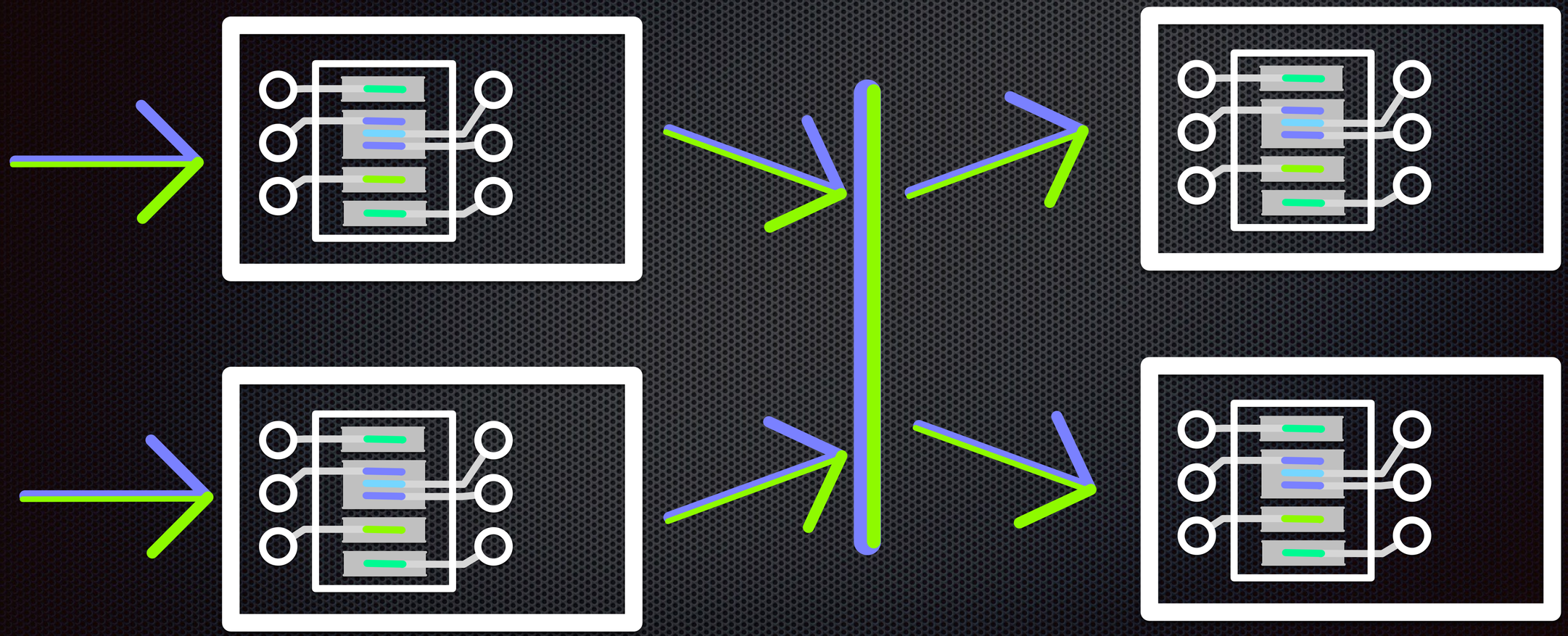
Memoization at construction time

- **Problem:** immutable objects —> lots of created objects

- @DesignatedFactoryMethod
- AspectJ runtime weaver


backends
hornet
frontend

Scaling



Storage

- In-memory custom store
- replacable
- just an *Aggregator*
- makes blue-green deployments a beeze

Wins

Constraint

- immutable model
- algebraic model
- Inverse references
- controlled state
- (functional) microkernel style

Gain

- execution stragety freedom
- parallelism
- scalability
- memoization at constructor time
- cacheability
- composability

Wins

Constraint

- immutable model
- algebraic model
- Inverse references
- controlled state
- (functional) microkernel style

Gain

- execution stragety freedom
- parallelism
- scalability
- memoization at constructor time
- cacheability
- composability

Wins

Constraint

- immutable model
- algebraic model
- Inverse references
- controlled state
- (functional) microkernel style

Gain

- execution stragety freedom
- parallelism
- scalability
- memoization at constructor time
- cacheability
- composability

Wins

Constraint

- immutable model
- algebraic model
- Inverse references
- controlled state
- (functional) microkernel style

Gain

- execution stragety freedom
- parallelism
- scalability
- memoization at constructor time
- cacheability
- composability

Wins

Constraint

- immutable model
- algebraic model
- Inverse references
- controlled state
- (functional) microkernel style

Gain

- execution stragety freedom
- parallelism
- scalability
- memoization at constructor time
- cacheability
- composability

Wins

Constraint

- immutable model
- algebraic model
- Inverse references
- controlled state
- (functional) microkernel style

Gain

- execution stragety freedom
- parallelism
- scalability
- memoization at constructor time
- cacheability
- composability

Architecture is Important

- It's the **set of constraints** you choose for your system
- It how those constraints **work in concert**
- It happens on a **smaller scale** than you usually think

Acknowledgments

- Clojure
  (especially Rich Hickey's talk on values, state and identity)
- Lamina
  https://github.com/ztellman/lamina
- Apache Storm
  https://storm.incubator.apache.org
- Casading
  http://www.cascading.org
- Akka
  http://akka.io

Thank You

ognen.ivanovski@netcetera.com

@ognenivanovski