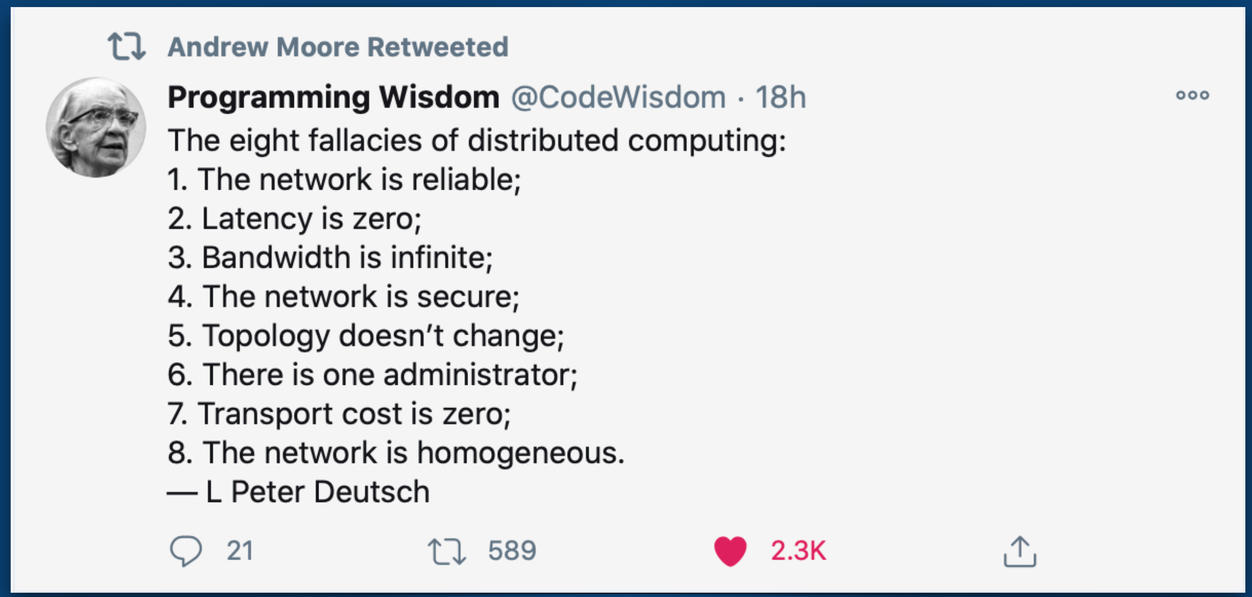
University of Applied Sciences
HOCHSCHULE EMDEN·LEER Networked Systems

PICCOLO

Information-Centric Dataflow

Re-Imagining Reactive Distributed Computing

Dirk Kutscher, Laura Al Wardani, T M Rayhan Gias
ACM ICN-2021



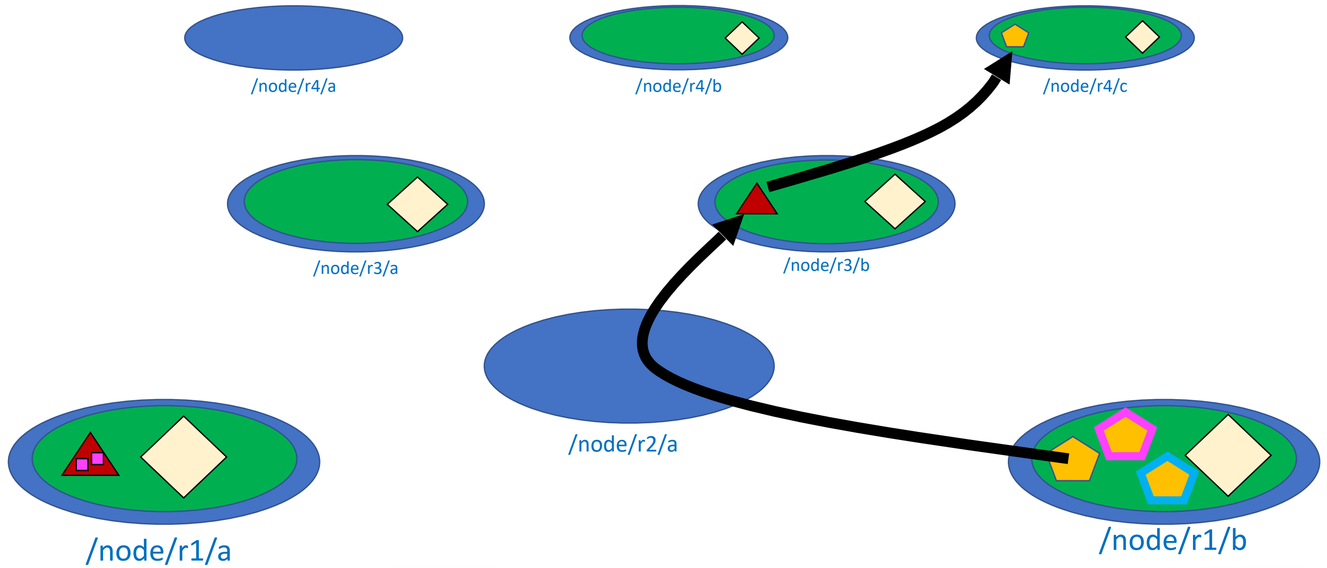
University of Applied Sciences

HOCHSCHULE EMDEN·LEER Networked Systems

Distributed Computing

Many Different Types of Interactions

- Message passing
- Remote Method Invocation
- Dataset synchronization
- Key-value store



Compute First Networking: Distributed Computing meets ICN

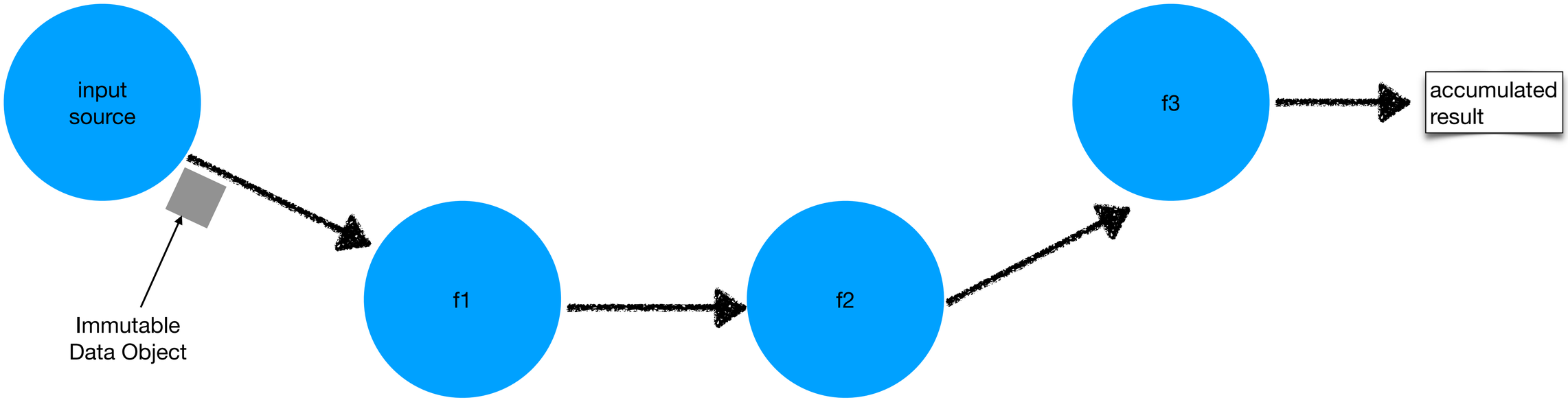
Michał Król[1], Spyridon Mastorakis[2], Dave Oran[3], Dirk Kutscher[4]

[1]University College London/UCLouvain
[2]University of Nebraska, Omaha
[3]Network Systems Research & Design
[4]University of Applied Sciences Emden/Leer

University of Applied Sciences
HOCHSCHULE EMDEN·LEER Networked Systems

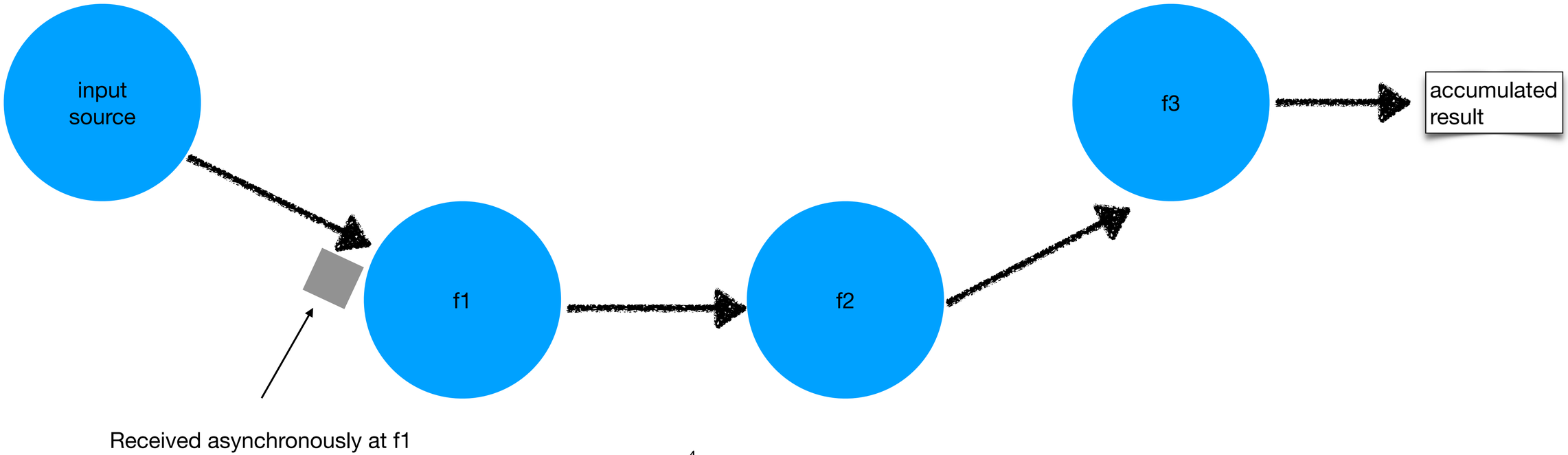
Dataflow

Structured Distributed Data Processing



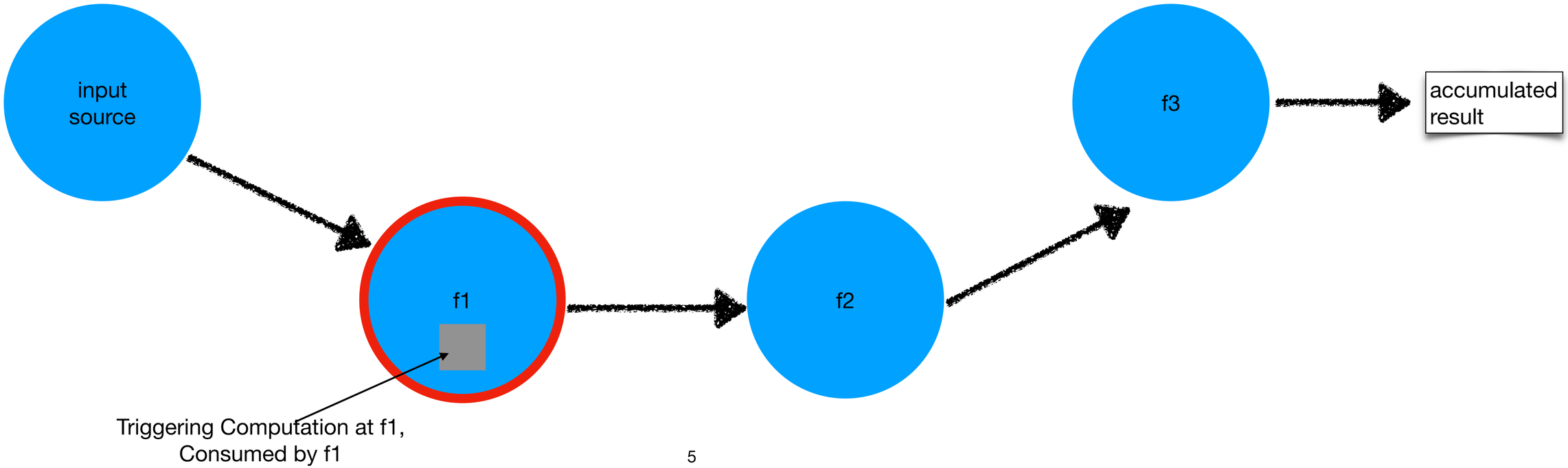
Dataflow

Structured Distributed Data Processing



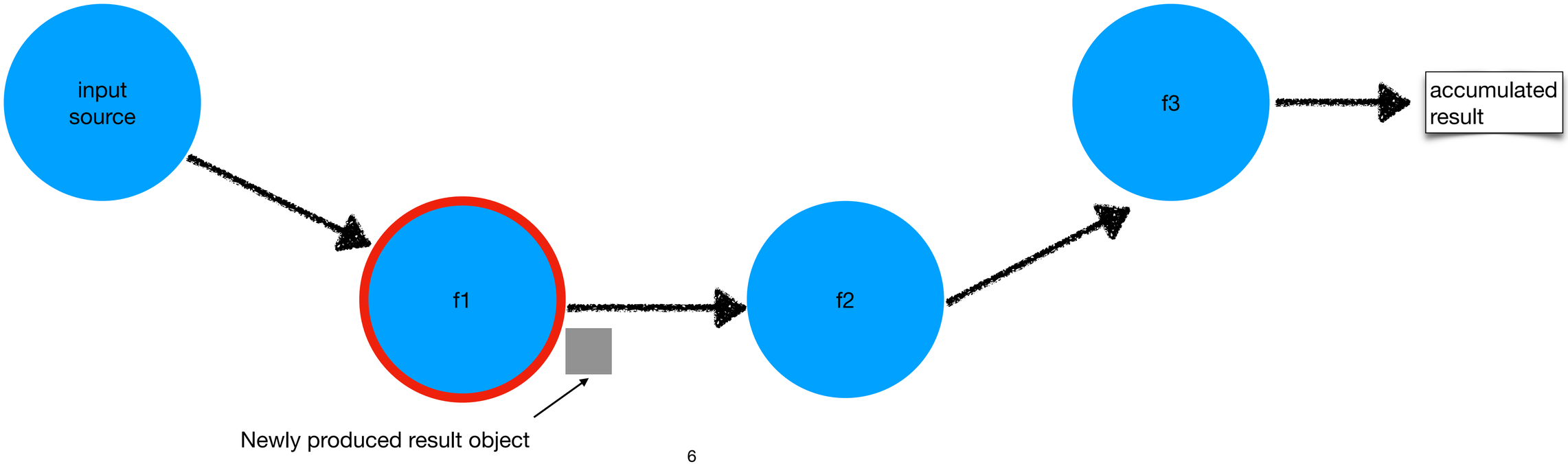
Dataflow

Structured Distributed Data Processing



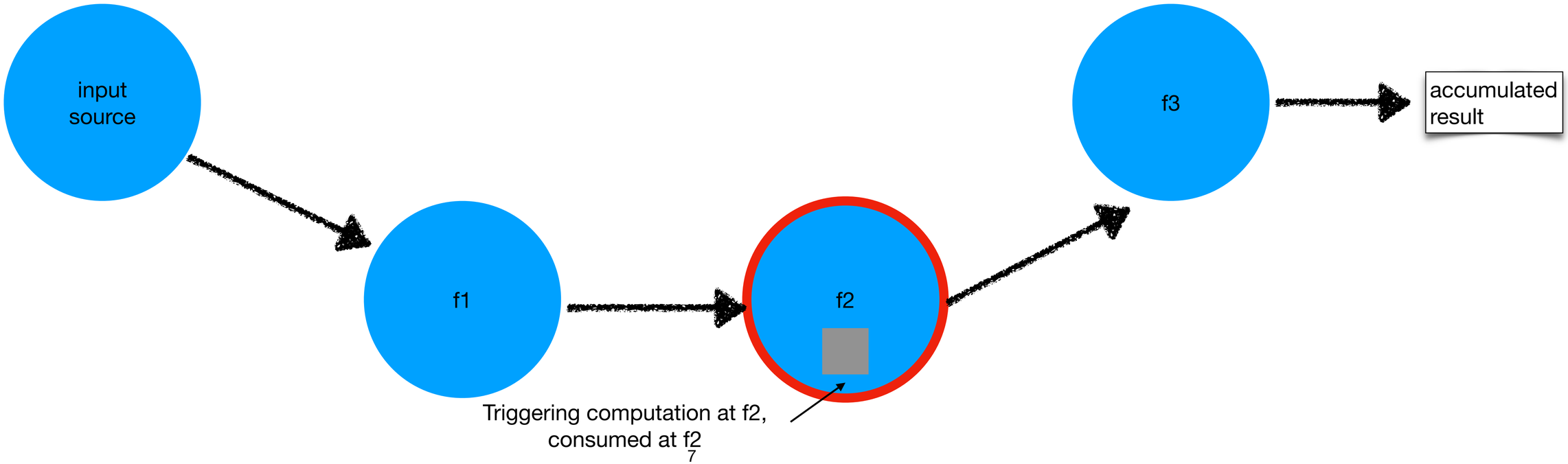
Dataflow

Structured Distributed Data Processing



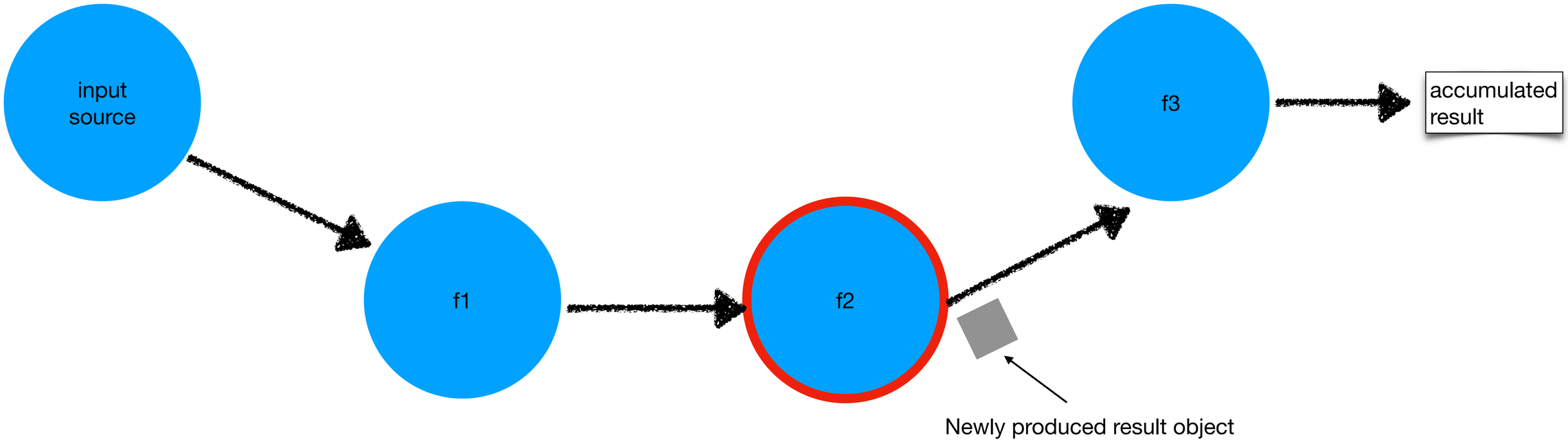
Dataflow

Structured Distributed Data Processing



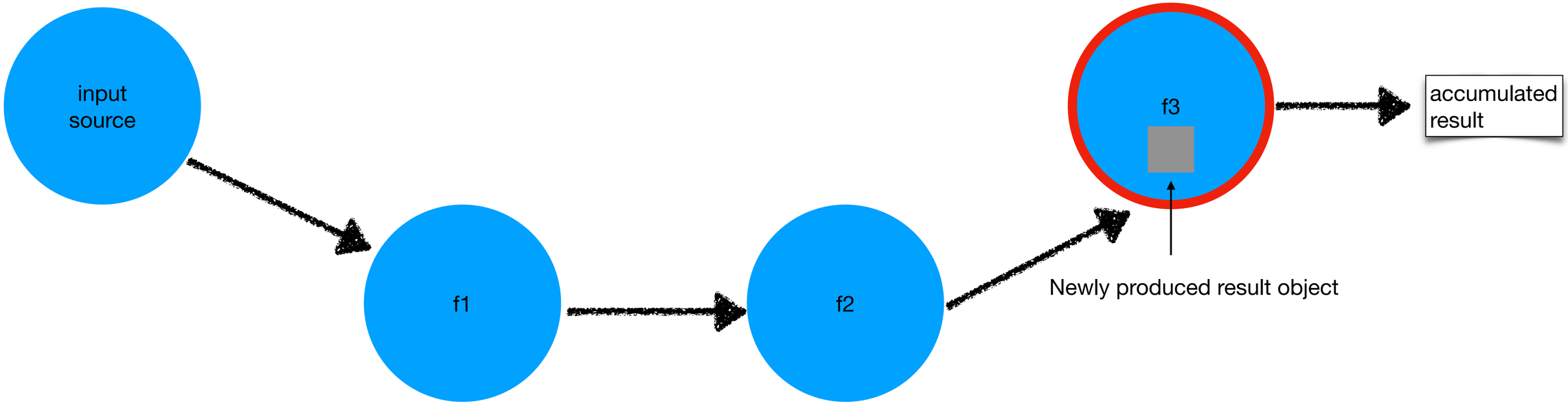
Dataflow

Structured Distributed Data Processing



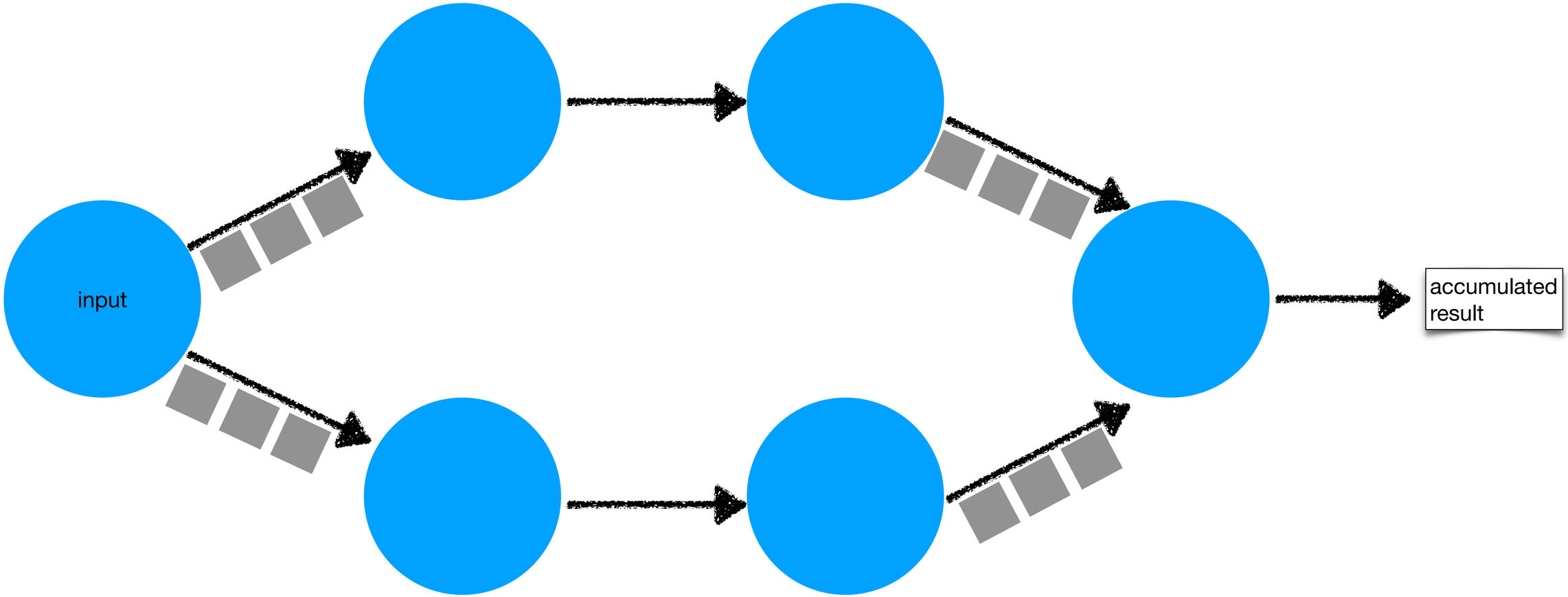
Dataflow

Structured Distributed Data Processing



Dataflow

Structured Distributed Data Processing

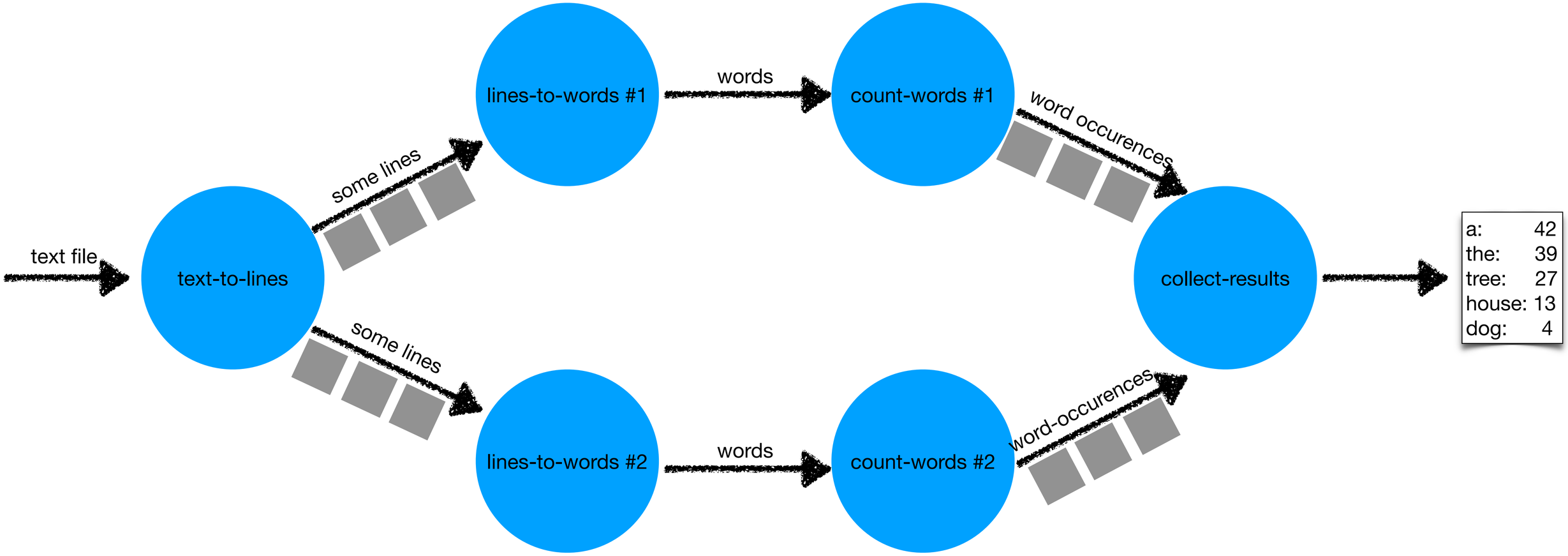


University of Applied Sciences
HOCHSCHULE EMDEN·LEER Networked Systems

Dataflow

Poster Child Example: word-count

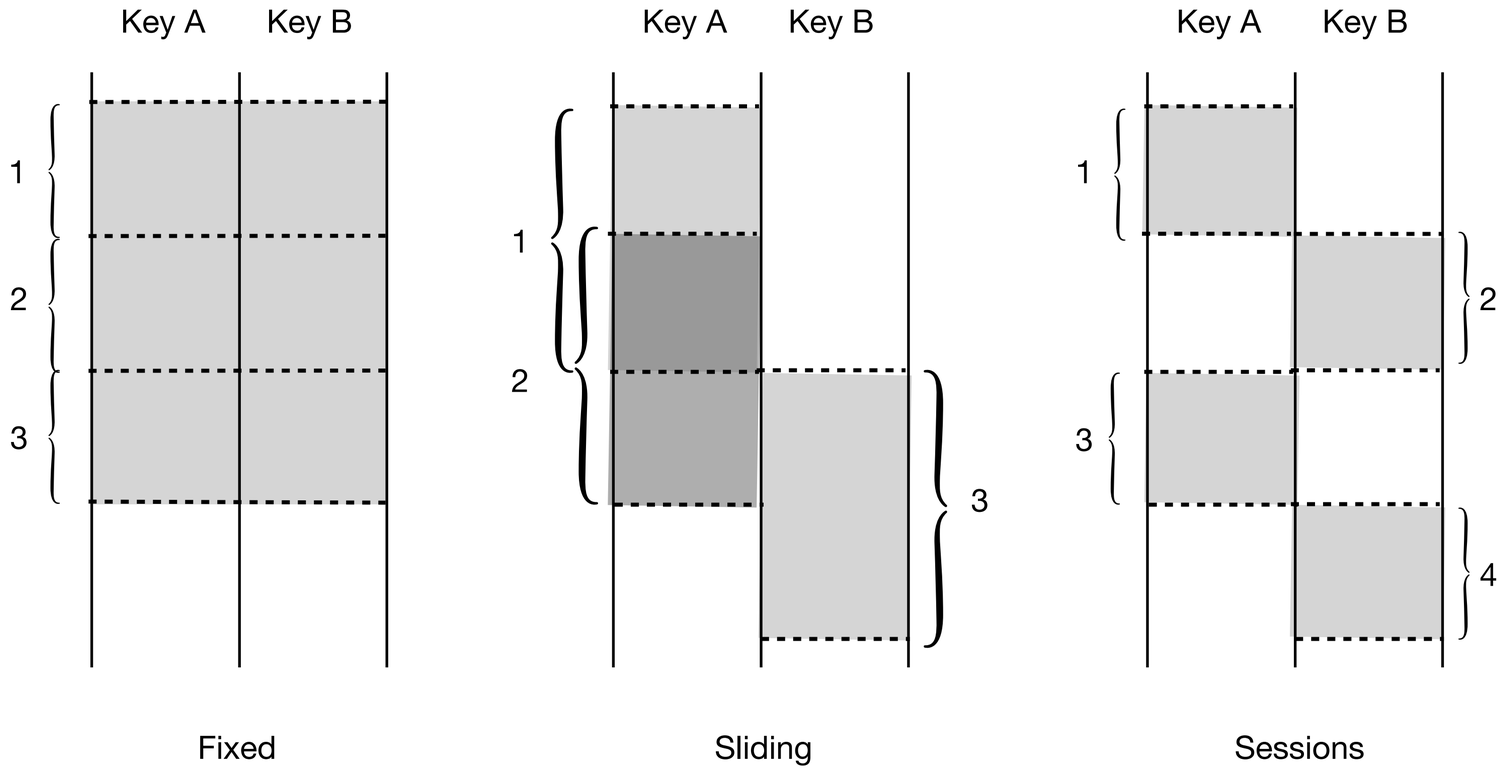
```
text-to-lines -> lines-to-words -> word-count ->
    ↪ collect-results
```



Dataflow Concepts

Batch & Stream Processing

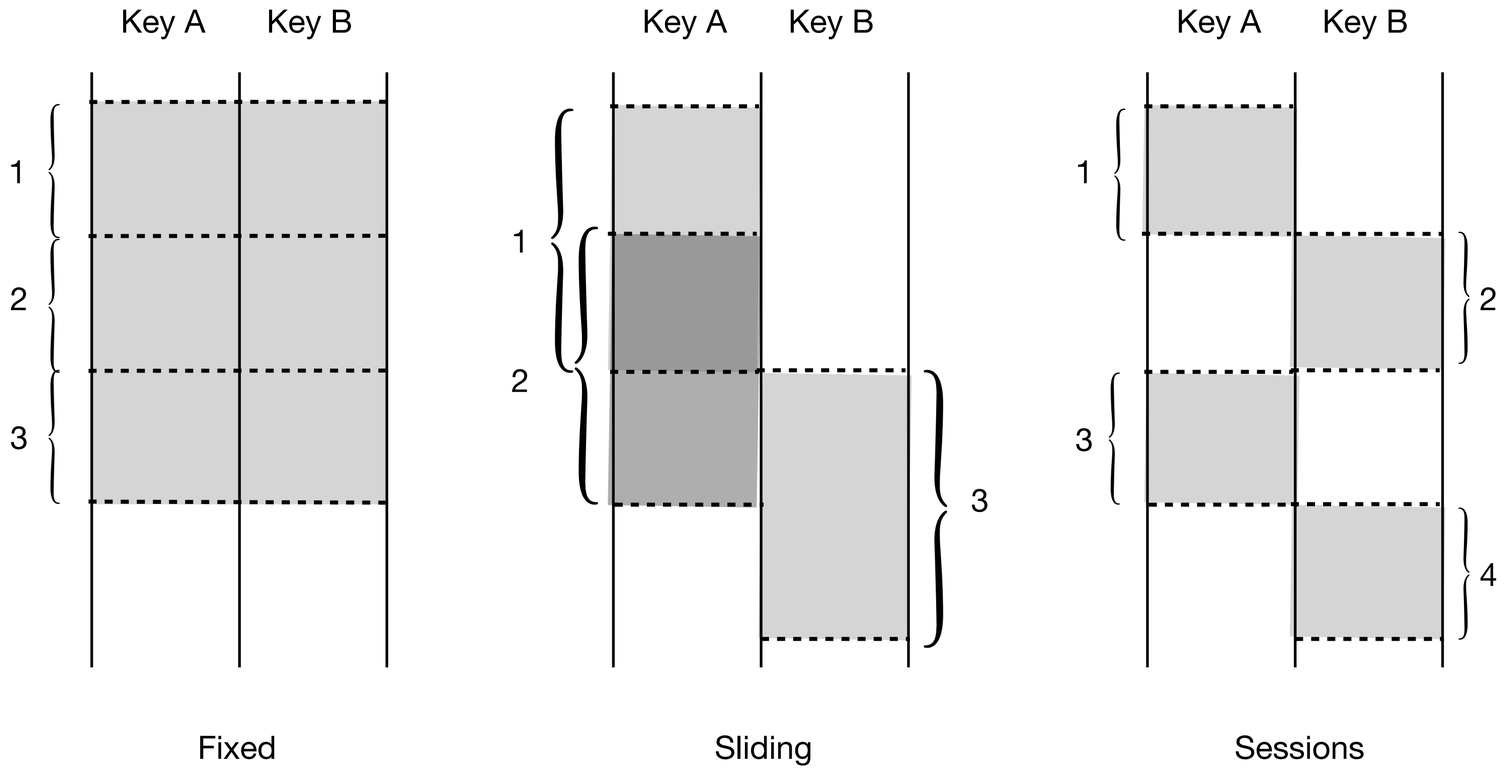
- Data objects as asynchronous events
  - Stream processing: each data object processed independently (unbounded)
  - Batch processing: grouping of data objects (bounded)



Dataflow Concepts

Windowing

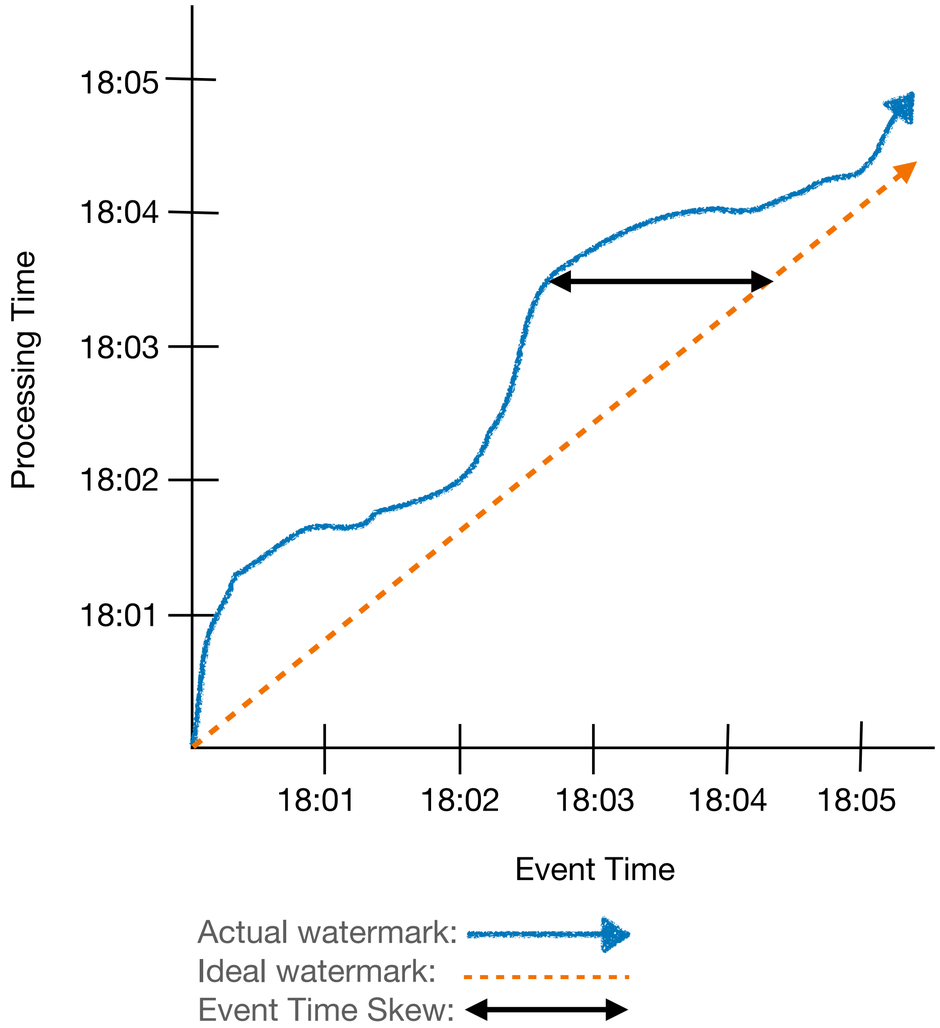
- Slicing data sets for processing as a group (aggregation)
- One data item can be assign to more than one group
- Directing data to specific consumers



Dataflow Concepts

Timing

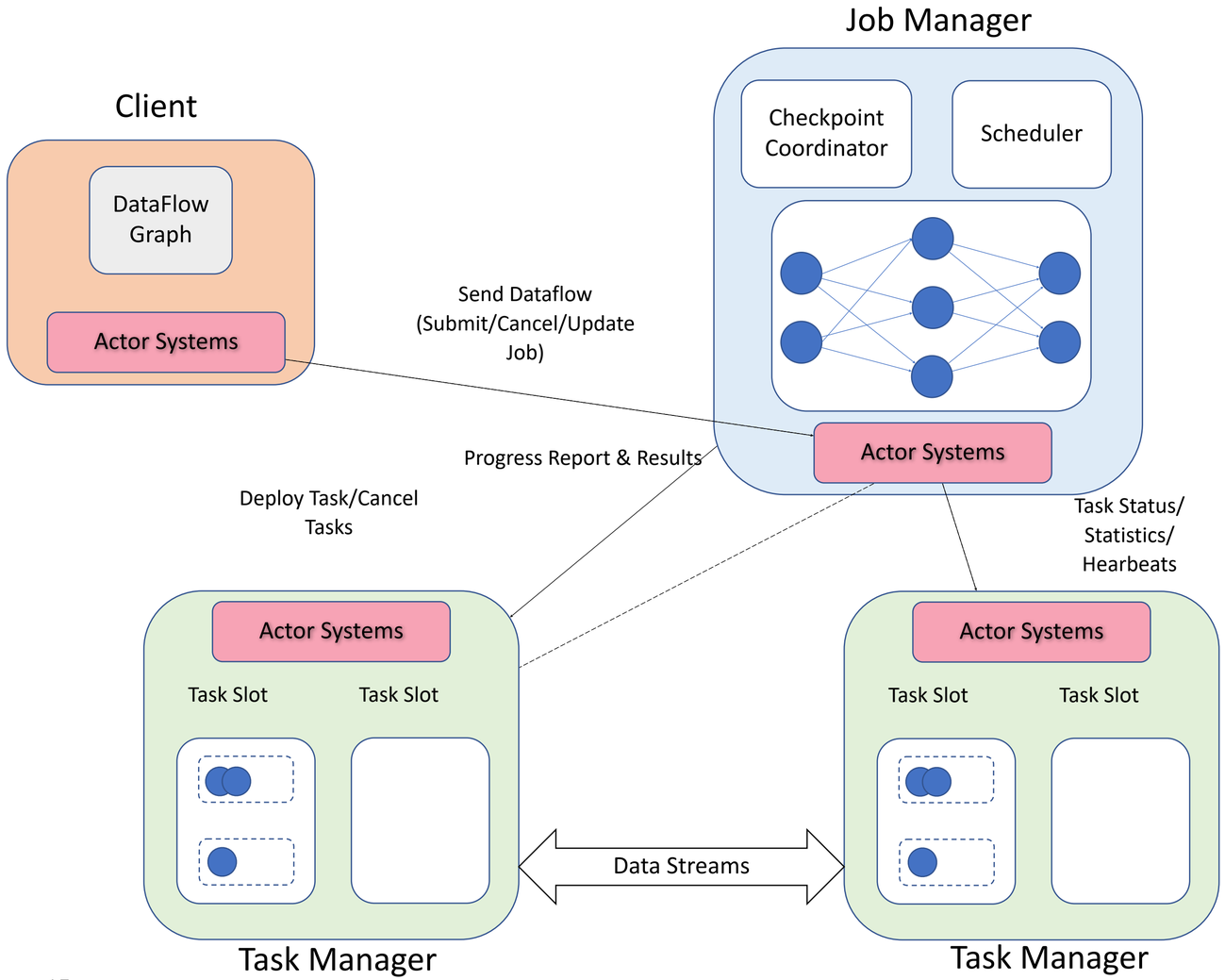
- Elastic data processing
- Asynchronous sourcing
- Unpredictable transport and processing delays
- Ideally: processing matches production rate
- Task of a Dataflow system: adjust processing graph to production rate and "real-time requirements"



Dataflow

Mainstream Implementations

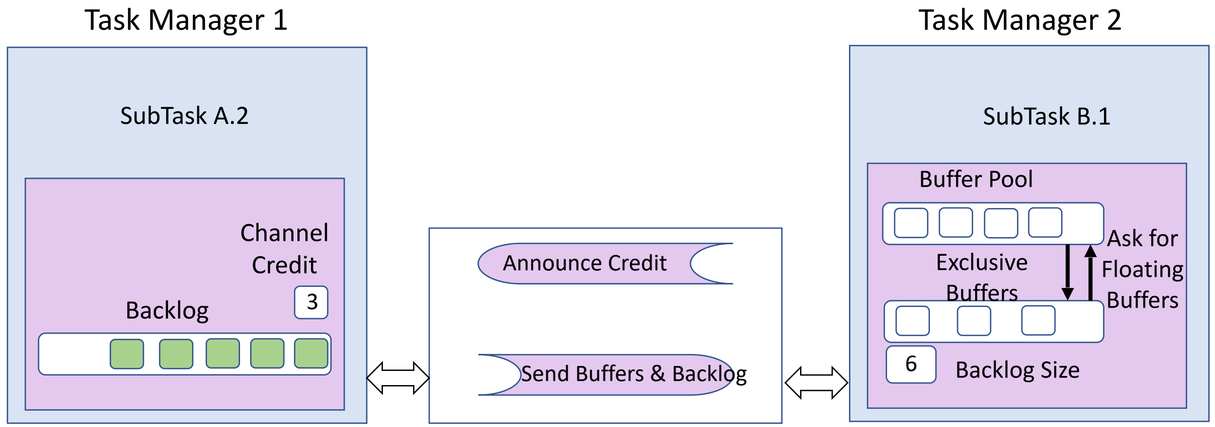
- Apache BEAM
  - Unified programming model for data processing pipelines
- Dataflow runners
  - Execution environments for Dataflow applications
  - Apache Flink, Samza, Spark
  - Google Cloud Dataflow



Dataflow

Transport and Back Pressure

- Example: Apache Flink
- Connections connect task managers, not tasks
- Need to regulate upstream processing rates

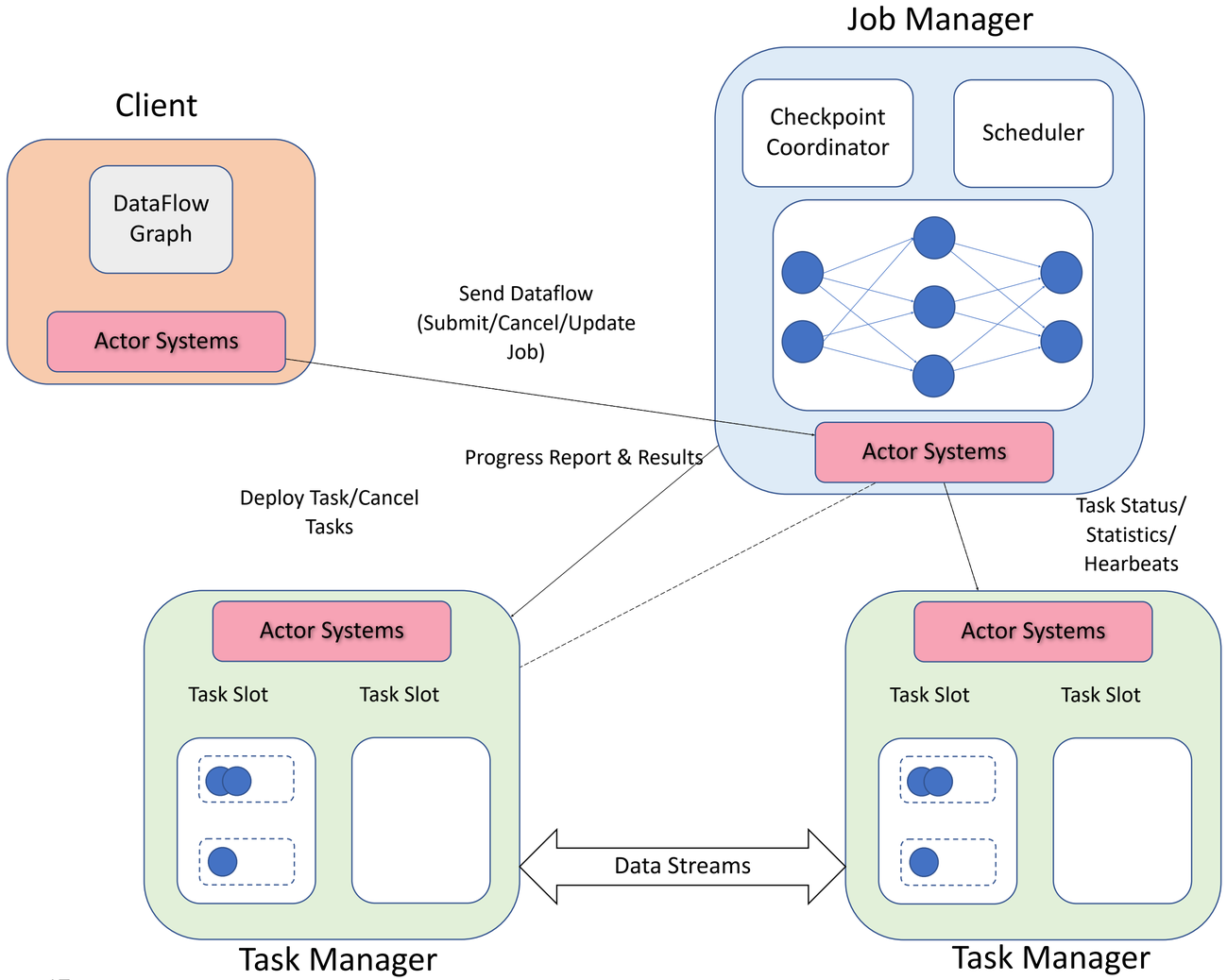


University of Applied Sciences
HOCHSCHULE EMDEN·LEER Networked Systems

Problem Statement

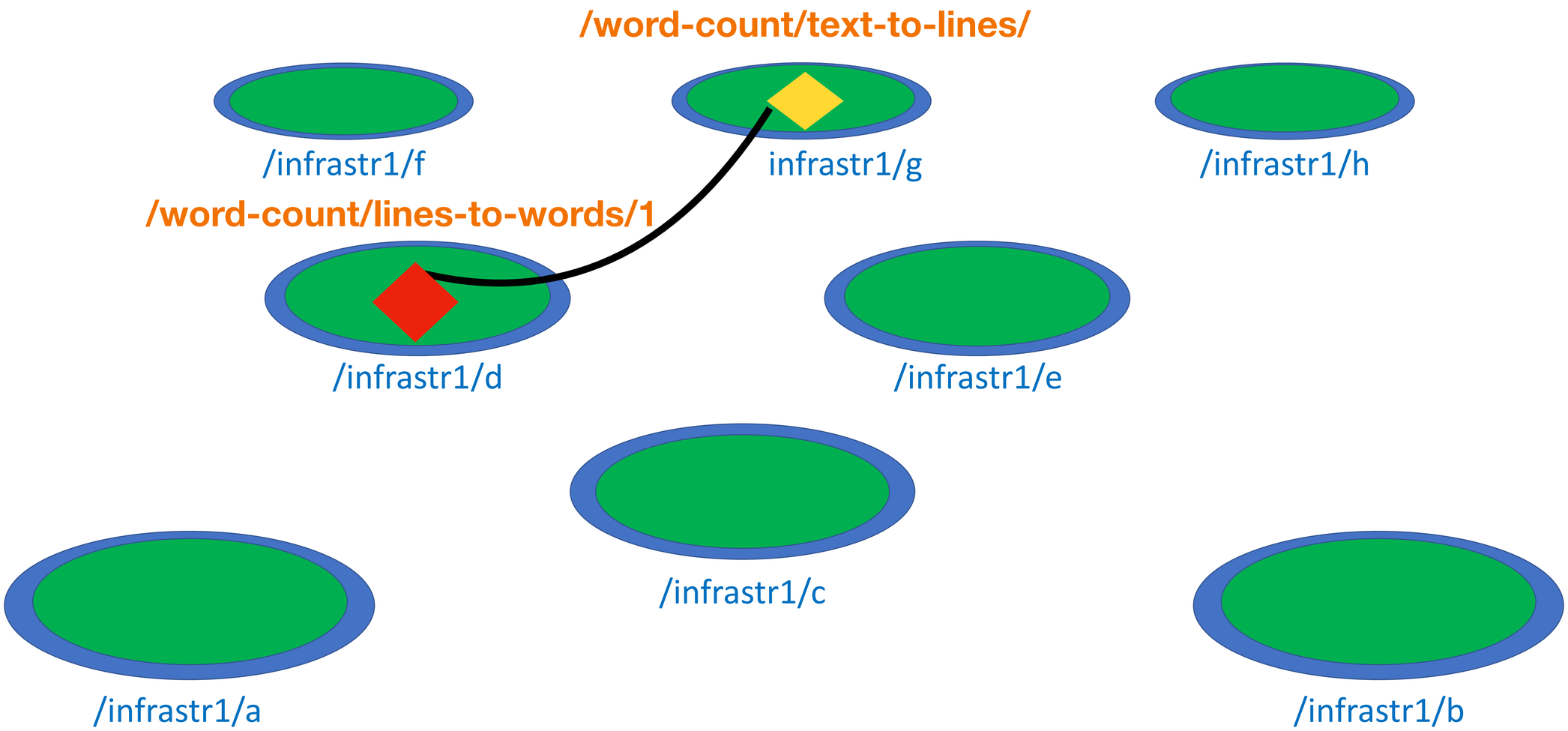
Overlays, Pipes, Address Mappings, Orchestration

- **Overlays do not match the inherent logic of processing immutable data objects**
  - Data is locked into connections
  - Connections are virtual channels between IP hosts
  - Orchestrator required to track resources, maintain mappings of task relationships to connections between hosts
- **Elastic Dataflow requires agile function instantiation, flow graph updates etc.**
- **Performance is a function of upstream data rates, network throughput, processing speed**
  - Limited visibility into root causes of performance problems at orchestrator



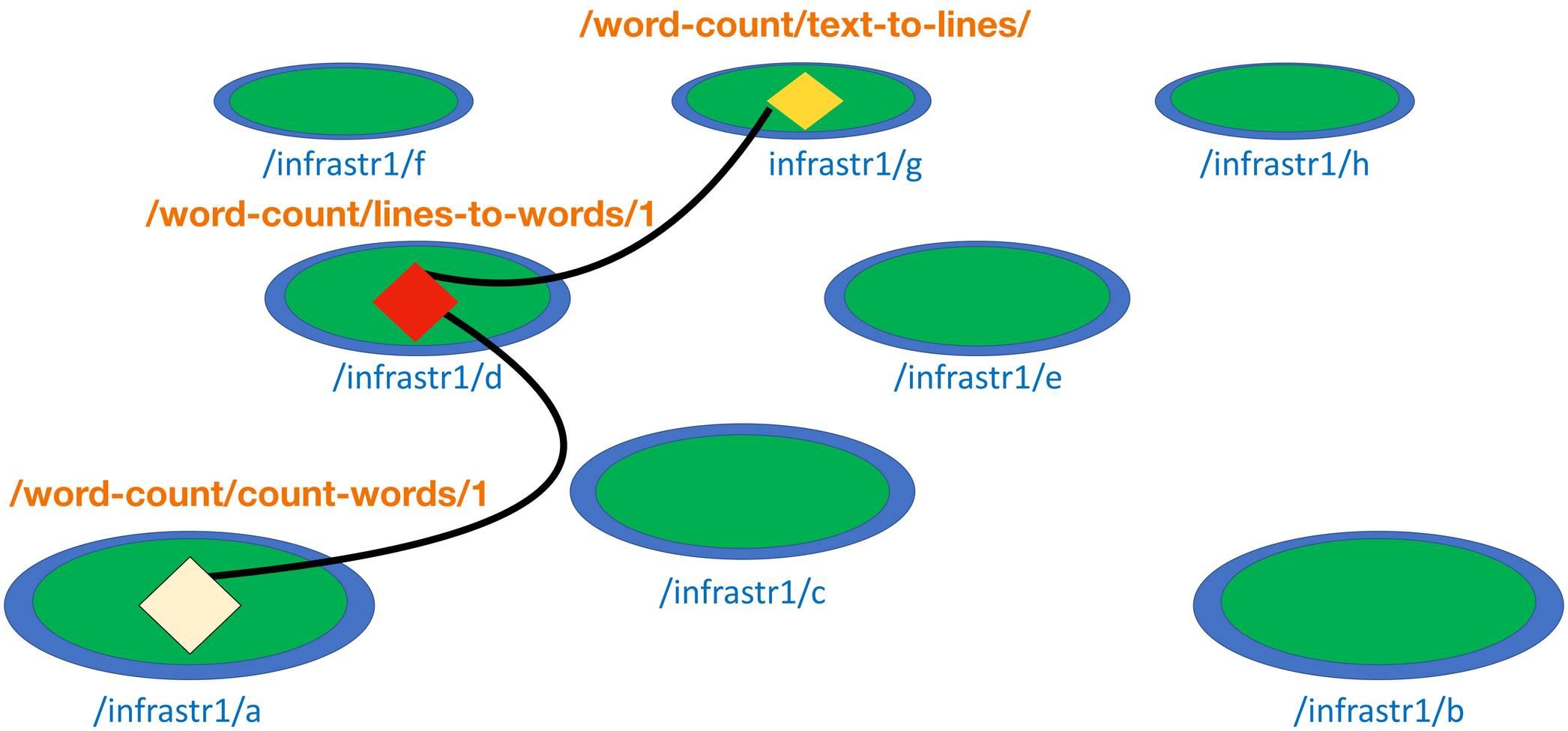
IceFlow

Information-Centric Dataflow



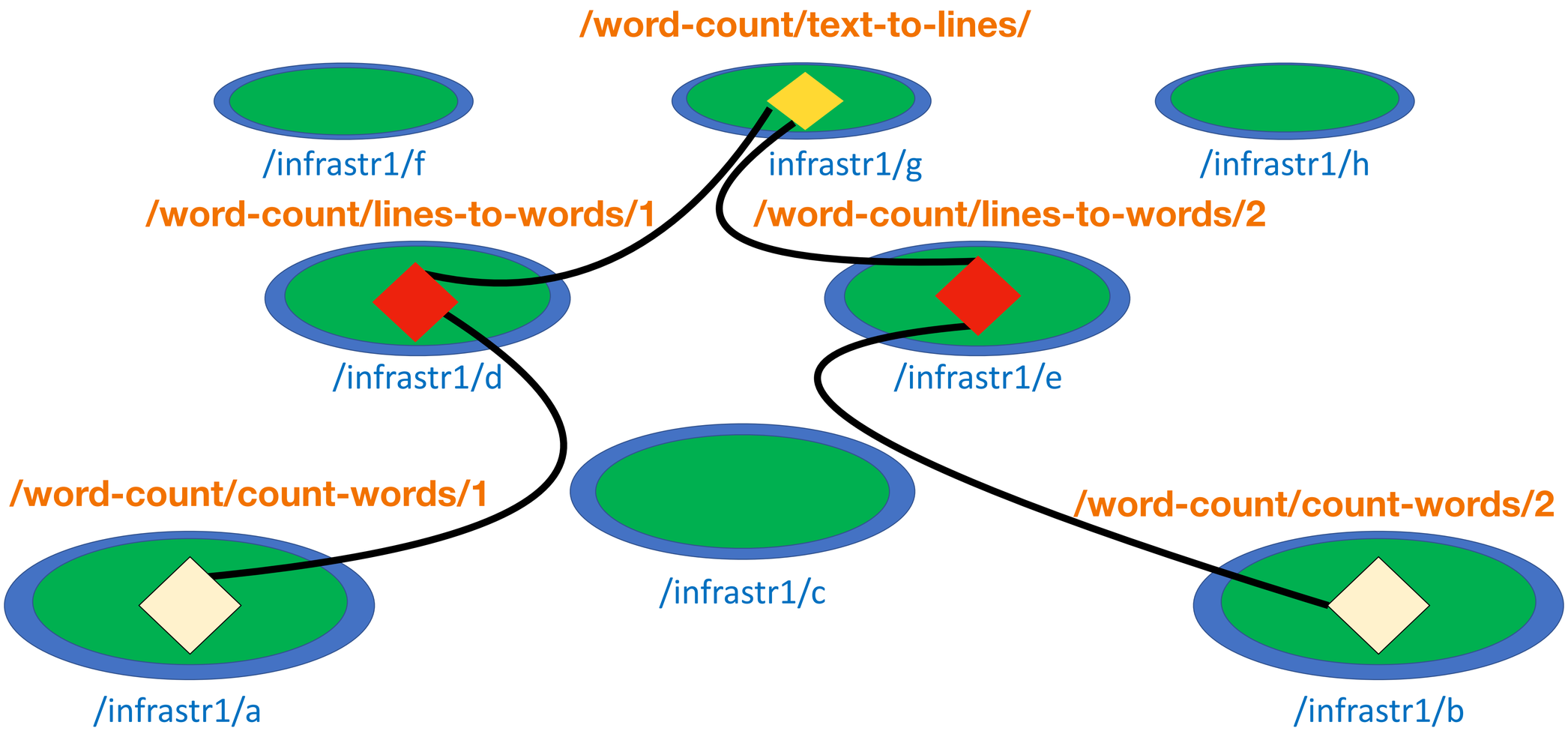
IceFlow

Information-Centric Dataflow



IceFlow

Information-Centric Dataflow



IceFlow

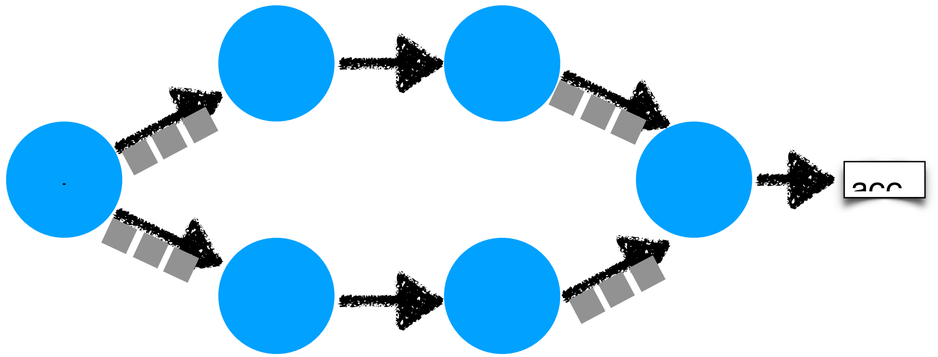
Concepts

- **Just Names**
  - For infrastructure
  - And for actors
- **Computation results as Named Data Objects**
  - Usual ICN properties...
- **Asynchronous data production**
  - Consumer has to know when data is available
- **Flow control**
  - Some coupling between consumers and producers
- **Garbage collection**
  - producers may be resource-constrained
  - cannot keep data forever

```
/[app]/[actor]/[instance]/data/[partition]/[object]
```

app	the name of the application
actor	the name of a Dataflow actor
instance	actor instance number
partition	monotonically increasing partition number to structure data objects on the producer's side
object	monotonically increasing sequence number

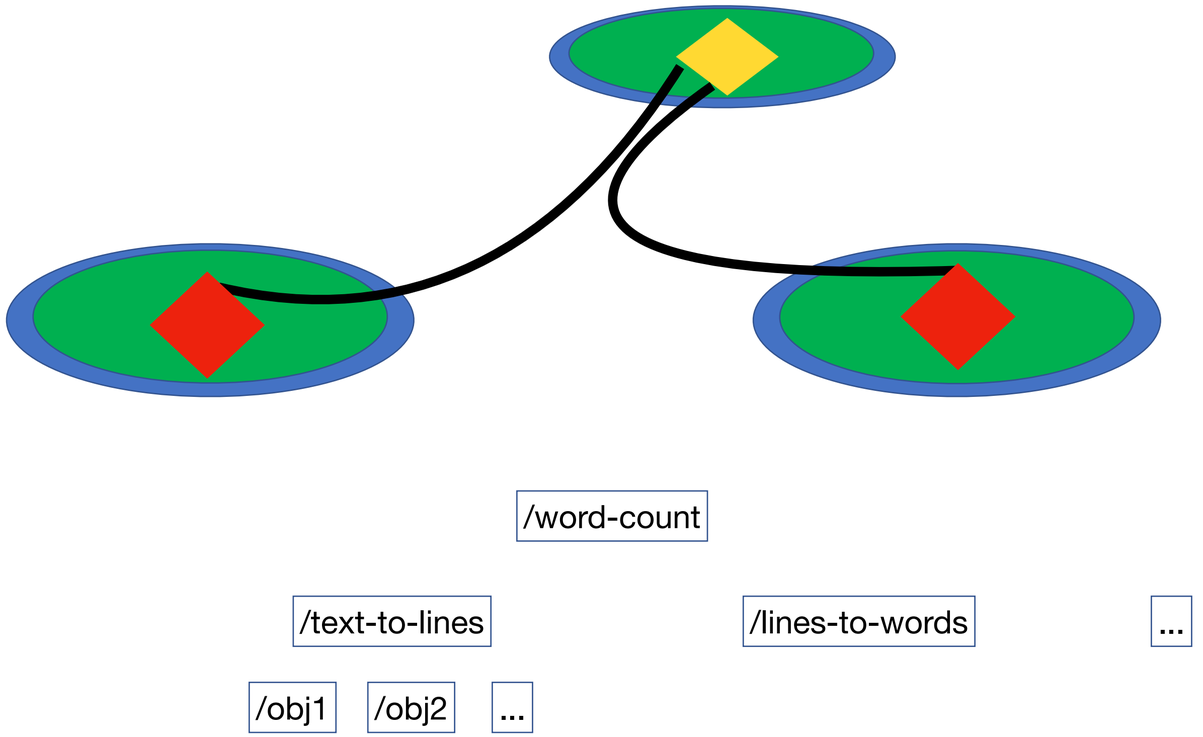
```
/word-count/text-to-lines/1/data/1/1
/word-count/lines-to-words/2/data/3/27
```



IceFlow Operation

Dataset Synchronization

- Producers produce data under a known prefix
  - Consumers subscribe to prefix
  - And learn update new input data
- Ideally: one prefix for whole application ("word-count")
  - Everyone could learn about all data in the app context
  - For practical reasons: need indirection
  - One prefix per consumer group

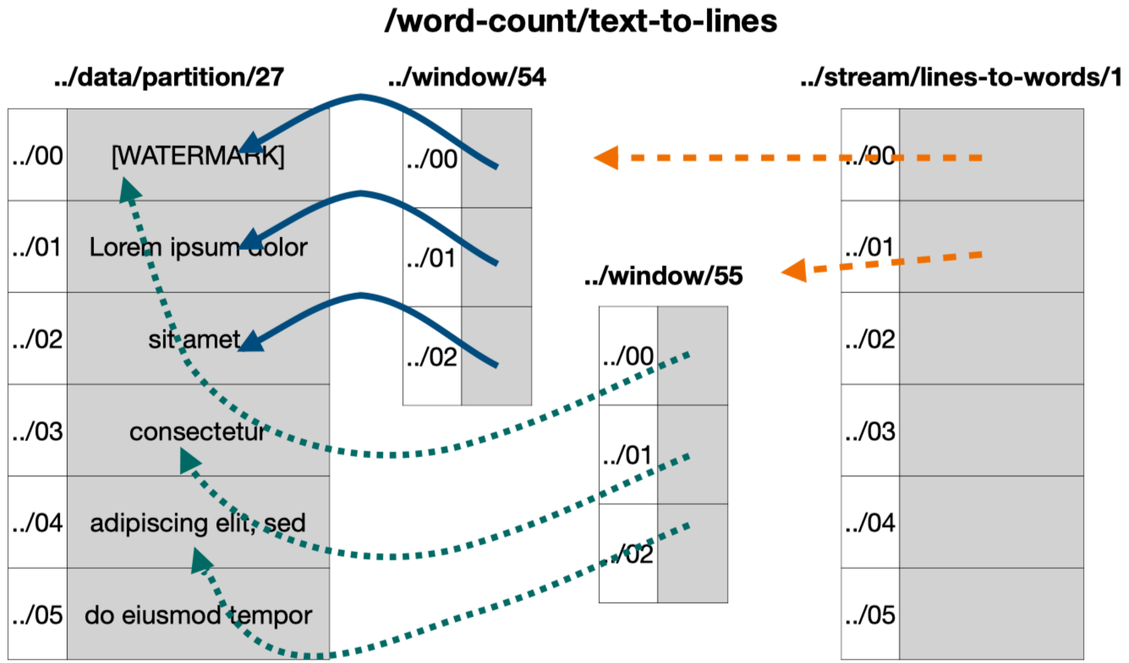


University of Applied Sciences
HOCHSCHULE EMDEN·LEER Networked Systems

IceFlow

Windows and Result Sharing

- Need more flexibility to re-use computation results in different contexts
  - Group data objects in windows
  - Group windows under per-consumer name prefixes

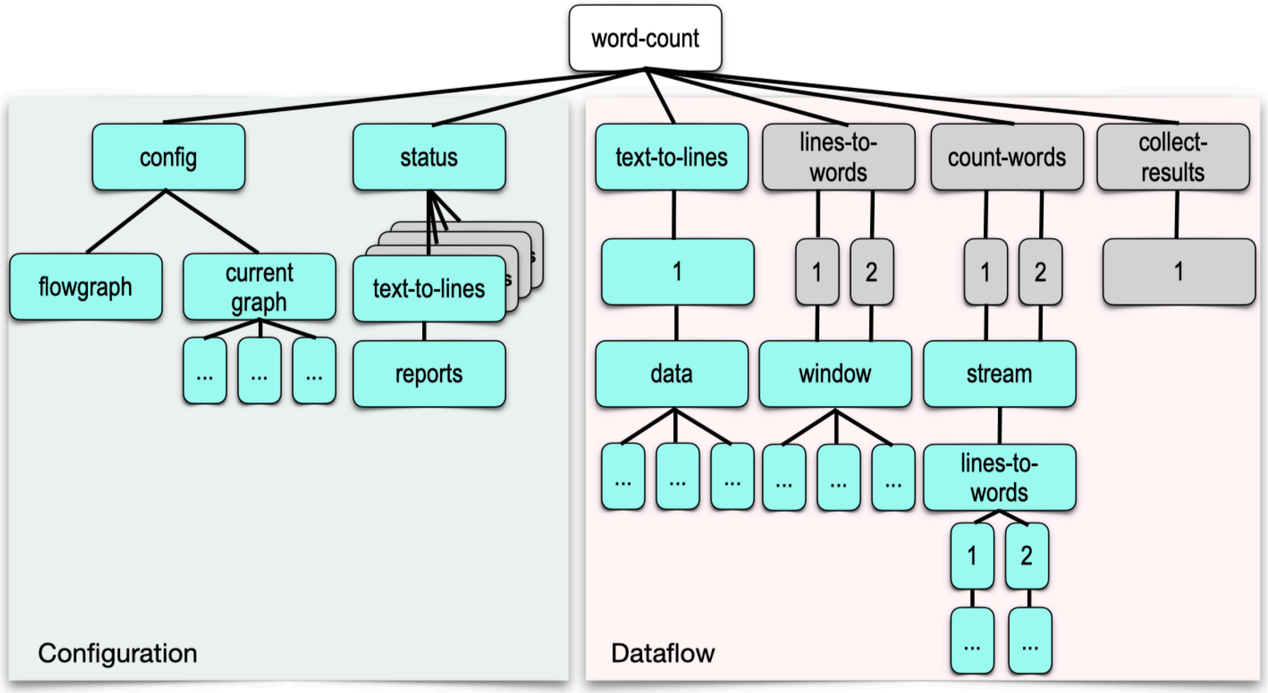


University of Applied Sciences
HOCHSCHULE EMDEN·LEER Networked Systems

IceFlow

Dataflow data and configuration

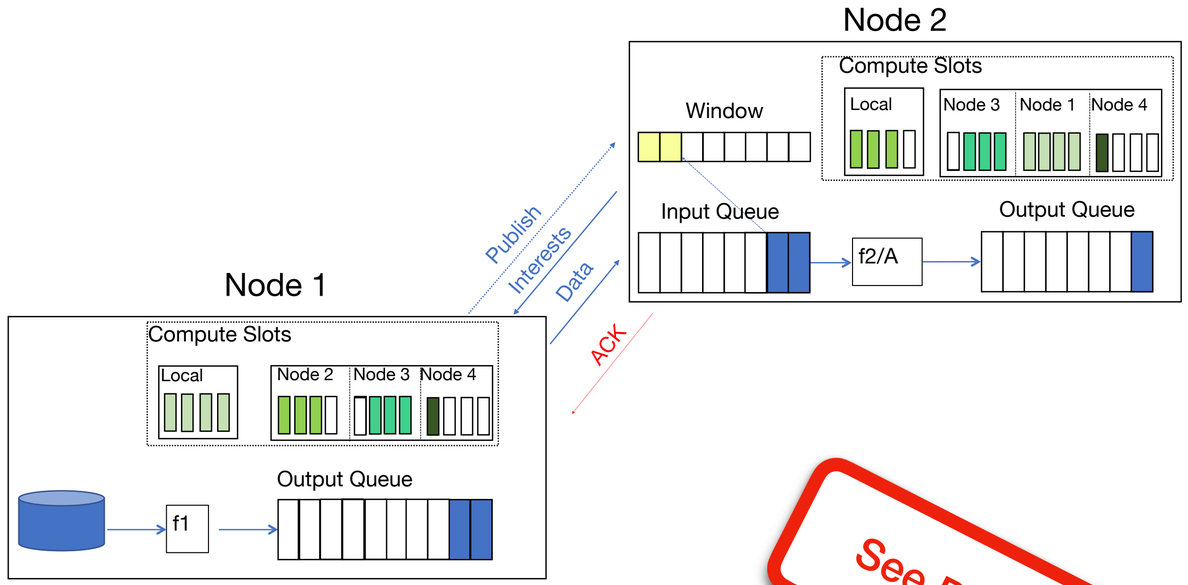
- Need additional shared information
  - Static application flowgraph
  - Actual current dynamic flowgraph
- Also: loose coupling between consumers and producers
  - Consumers reports: what windows have been processed
  - So that producer can advance
- Result: share namespace with Dataflow data and configuration info
  - Some config info represented in CRDTs (like in CFN)



IceFlow

Resource Management

- IceFlow can be smarter than receiver-driven AIMD
  - No need to fetch data that cannot be processed at throughput speed
  - "Receive Window"
- Producers should not overrun consumers
  - Output queue occupancy...
  - When consistently full: trigger scale-out



See Demo!

IceFlow

Insights So Far

- **Todays Dataflow systems are powering many data science applications**
- Overlay approach
  - Usual address mapping and virtual circuit issues
  - Limited data sharing
  - Centralized orchestration
- **Real opportunity for redesigning distributed data processing with ICN**
  - Elegant name-based approach: no mappings, no resolution – just data
  - Direct sharing of computation results
  - Potentially better visibility into network performance
- **Dataset synchronization in principle the right approach**
  - NDN Psync performance not great in experiments (NFD)
  - Also requires multicast forwarding strategy
- **Additional mechanisms needed**
  - Name-based routing (NLSR should be fine)
  - Failure recovery