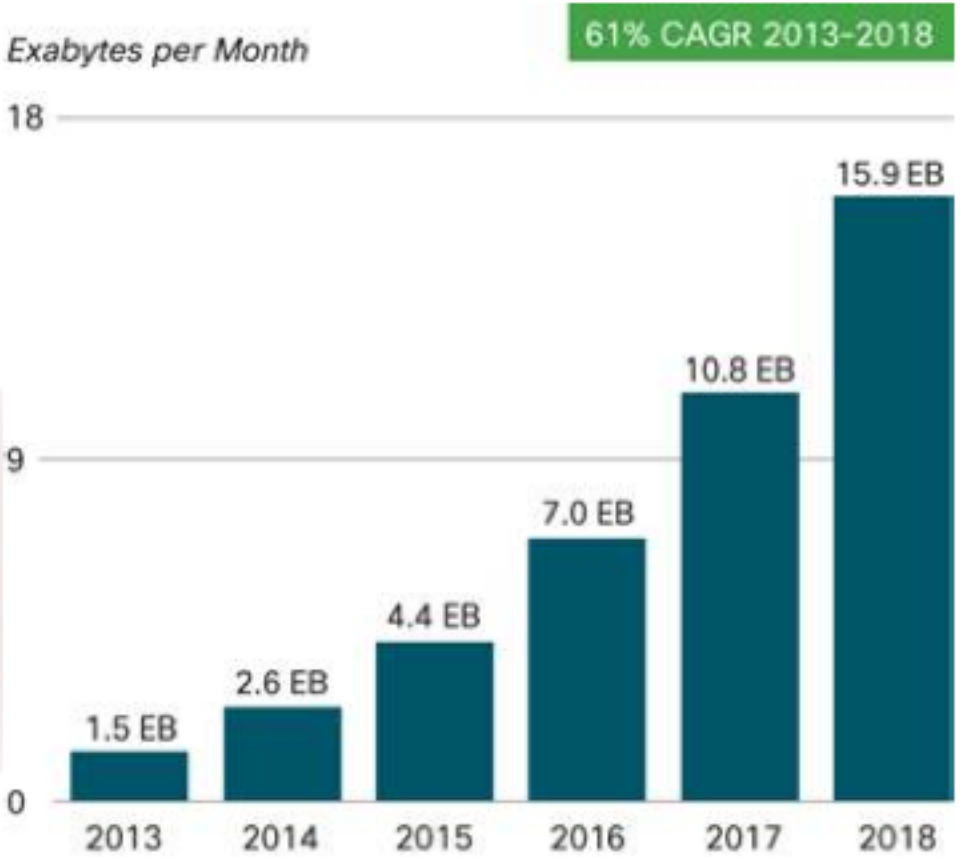
Epidemic store for massive scale systems

Francisco Maia
Advisor: Rui Oliveira

Braga, 30th April 2015

Context

- **Information systems are present in all aspects of our society**
- Ubiquity of **connectivity**
- **Escalation** in the number of connected devices
- **Unprecedented amounts of data** being produced



Source: Cisco VNI Mobile, 2014

Cisco Forecasts 15.9 Exabytes per Month of Mobile Data Traffic by 2018

HASLab

Context

- Massive scale systems:
  - Very high dynamism (**churn**)
  - **Faults** are the rule, not the exception
- **Current approaches to data management are not suitable** for these scenarios

Context

- Traditional relational database systems **scale rather poorly** (tens of nodes)
- NoSQL data stores **compromise consistency** guarantees **favouring availability** (simpler put and get API)
- These scale better but rely on a set of **master nodes or on a DHT**
- These assume moderately stable environments – very **sensitive to churn**


Cassandra


Google Bigtable


Problem and objectives

- We address the problem of data management in **massive scale** environments
- The goal is to build a **new generation** data store
- A data store that is inherently **scalable and can handle very high levels of churn**

Contributions

- We propose the design of **DataFlasks, a data store entirely built with epidemic protocols**
- In order to implement our design we:
  - Propose a novel **distributed systems slicing algorithm (Slead)**
  - Propose a novel **group construction algorithm**
- **DataFlasks prototype**

DataFlasks: design

- Core design idea: complete **decentralization**
- **Contrary to any other existing solution,** there is no **node hierarchy or structure**
- DataFlasks nodes are **autonomous and all play the same role**
- Progress based only on **node-local decisions**

DataFlasks: design

- Every node can receive requests
- The data store offers a **put** and **get** interface
- Data is **distributed over all the nodes** for load balancing
- Upon the reception of a request each node locally decides how to process it

Get

- If the node holds the value: replies to the client.
- Otherwise it disseminates the request to other nodes.

Put

- The node **locally decides** to store or discard the data.
- Disseminates the request to other nodes.

HASLab

Challenges

- How to implement the decision of storing or discarding data
- Efficient request dissemination

Storage Decisions

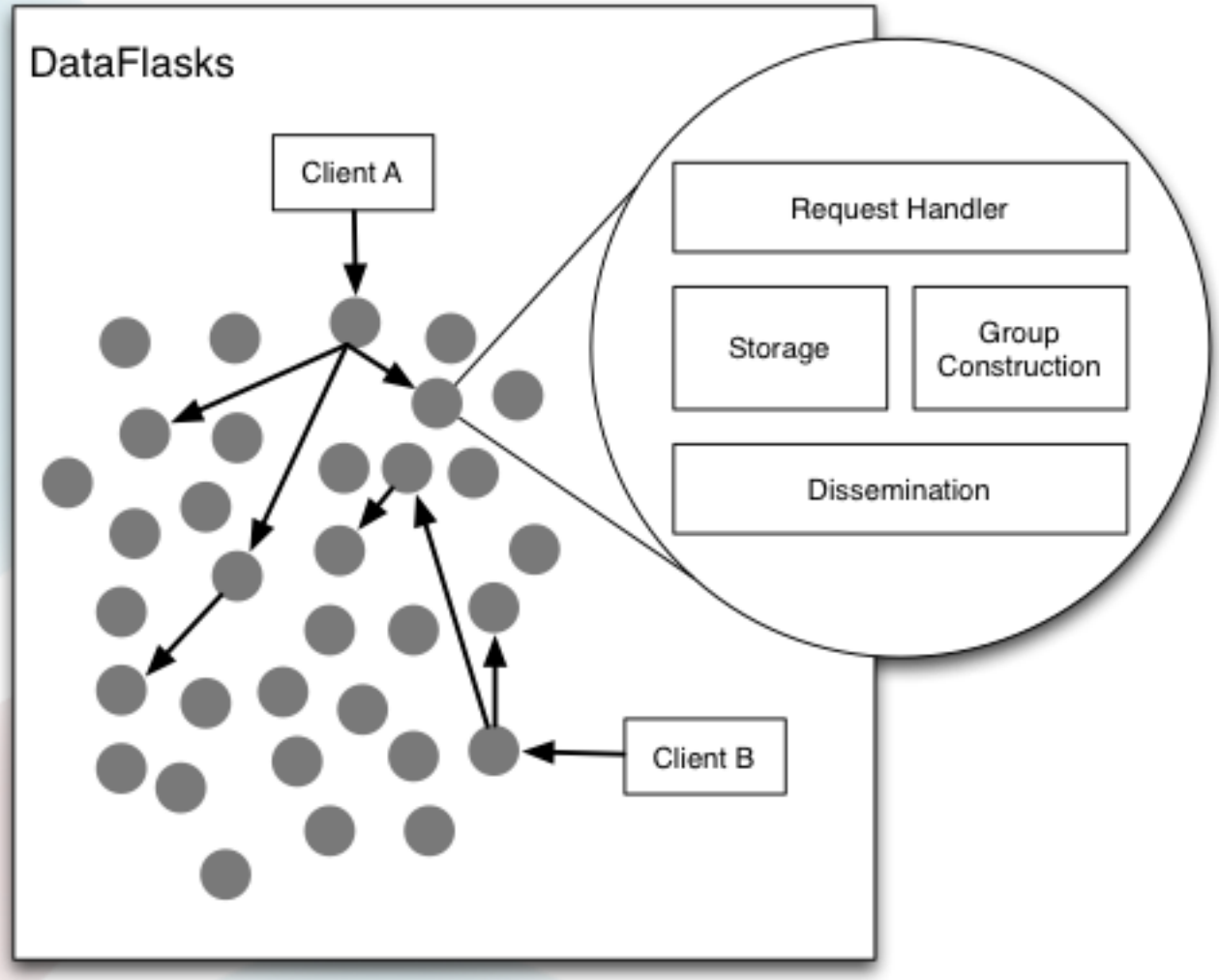
- **Divide nodes into groups**
- Each group is responsible for a set of data
- Data distribution follows a map between data and groups
- The size of the group is the replication factor
- **Group construction algorithms**

Request dissemination

- Epidemic dissemination protocols exist in the literature
- These are efficient and robust

HASLab

DataFlasks: architecture



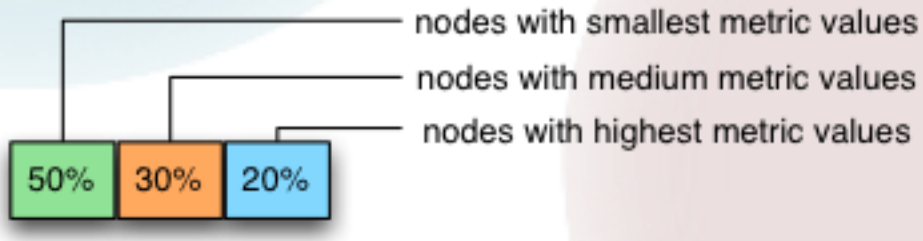
Group construction

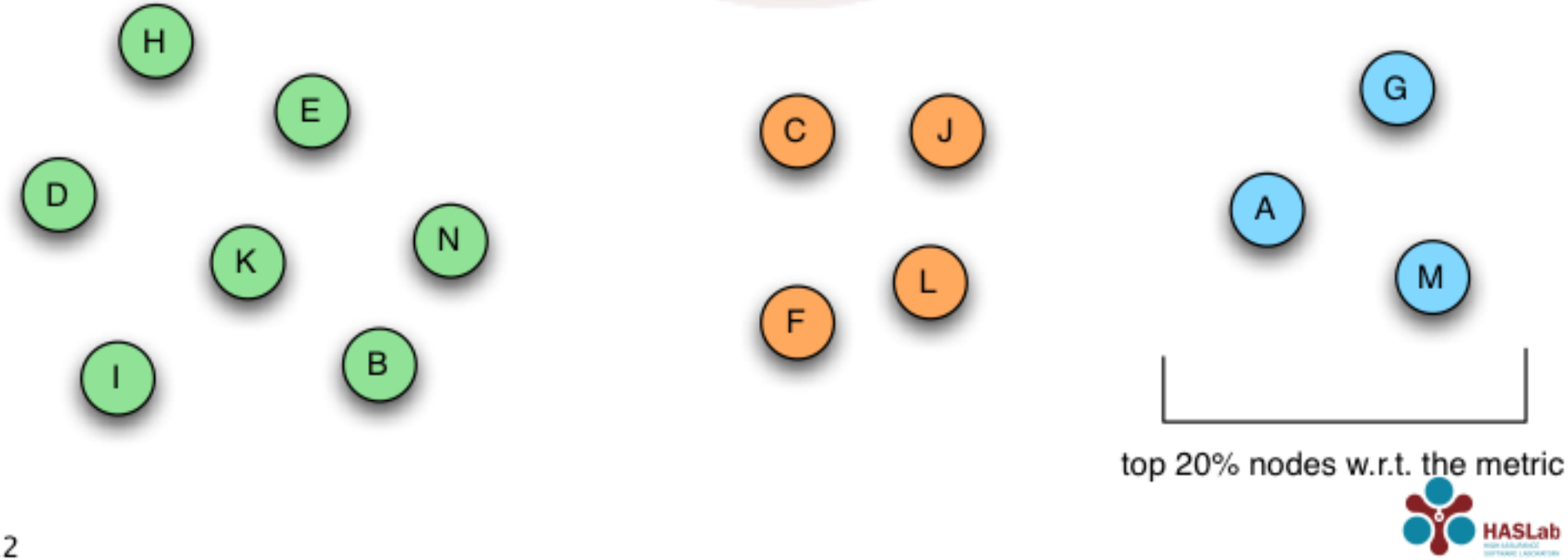
- Must **divide** an arbitrarily large set of nodes **into groups**
- Each node running the protocol must learn to which group it belongs
- The total number of nodes is unknown

- To the best of our knowledge there was no existing suitable **epidemic protocol**
- The best candidates in the literature were **slicing protocols**

HASLab

Distributed Systems Slicing





HASLab

Distributed Systems Slicing

Distributed Systems Slicing

- Each node has access to a **local attribute of some kind** (ex. disk space)
- It also has access to a **set of neighbors** (view) refreshed by a Peer Sampling Service
- **Periodically, each node sends its attribute to the nodes in its view**
- The attributes received are used to estimate the node's relative position in the system
- From this relative position the node calculates the corresponding slice

HASLab

Distributed Systems Slicing

Distributed Systems Slicing

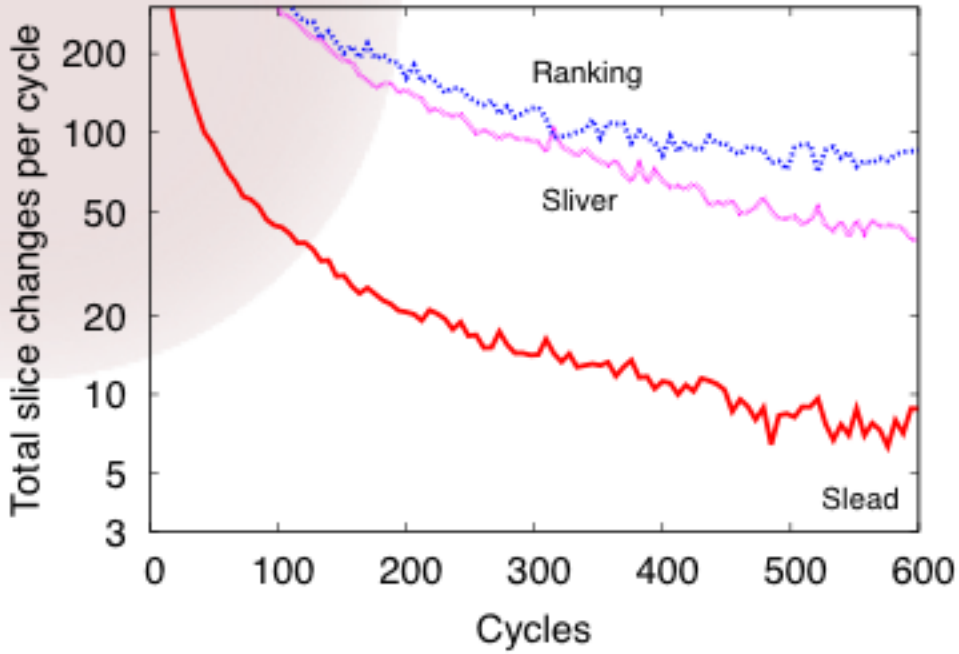
- Slicing protocols exhibit several frailties addressed by our work
  - **high instability** (frequent slice change)
  - excessive memory consumption
  - inability to dynamically change slice configuration
  - inability to have heterogeneous slice sizes

HASLab

Sload

Slead

- We introduced a **hysteresis mechanism**
- This mechanism **avoids unnecessary slice changes**
- It works by introducing a delay in the slice change decision



HASLab

Group construction

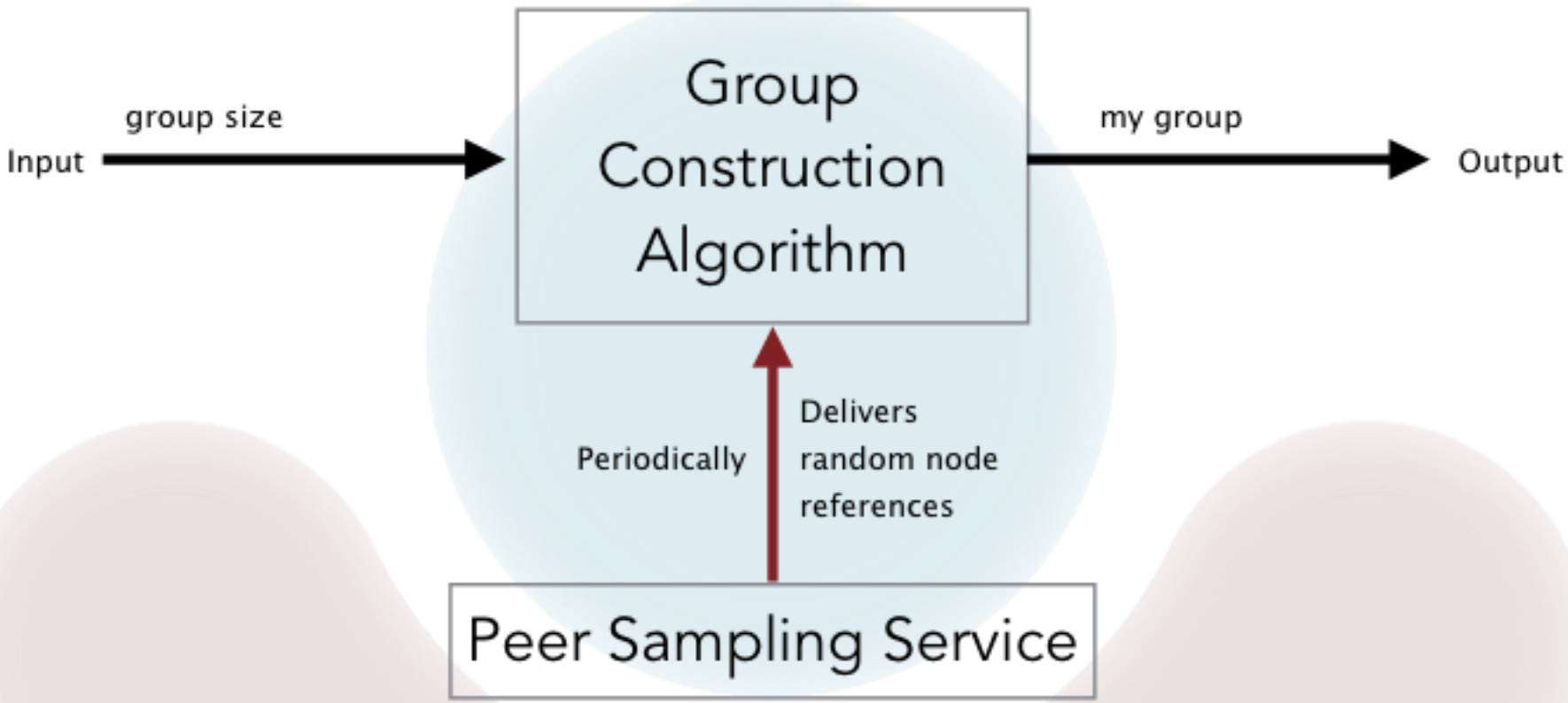
Group construction

- Regardless of the improvements, Slead still exhibits **problems of instability**
- Moreover, it is **not possible to define slice size**
- **We propose a novel group construction algorithm**

HASLab

Group construction

Group construction

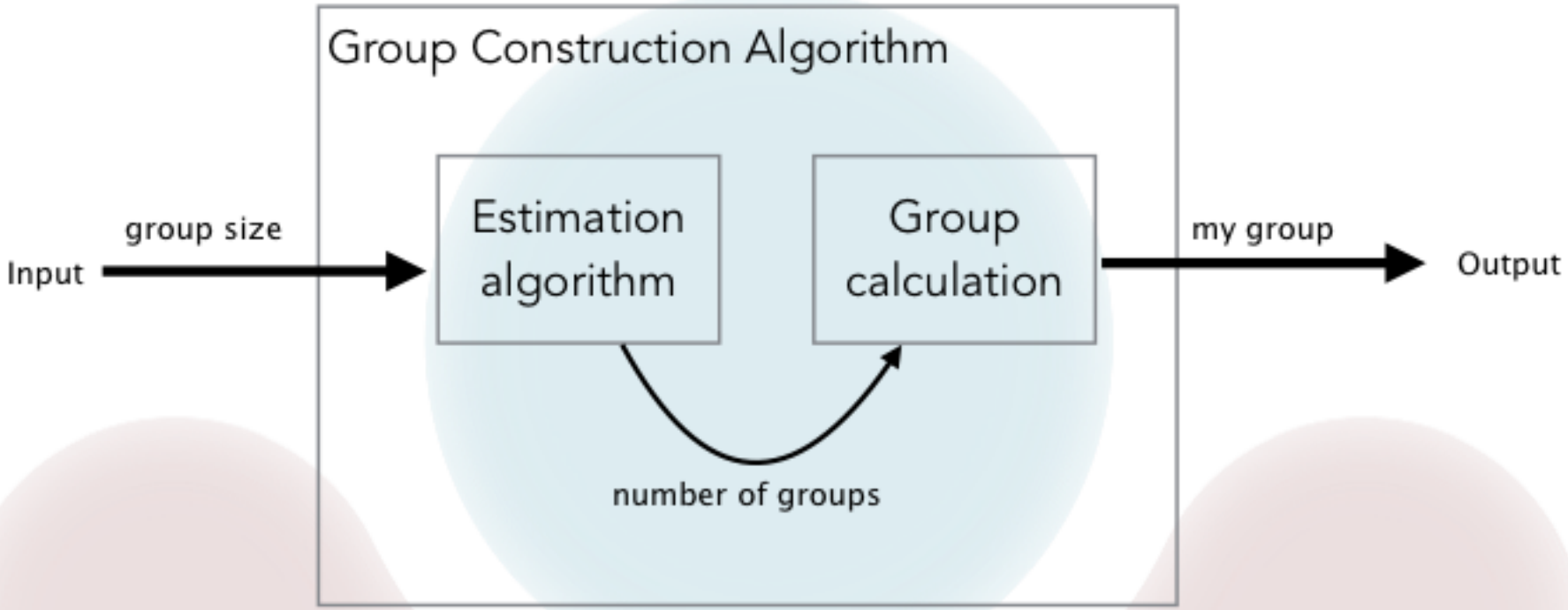


HASLab

Group construction

Group construction

- Because the total number of nodes is unknown, each node **estimates the number of groups** needed to **satisfy** the desired **group size**
- From such estimation calculates the group it belongs to



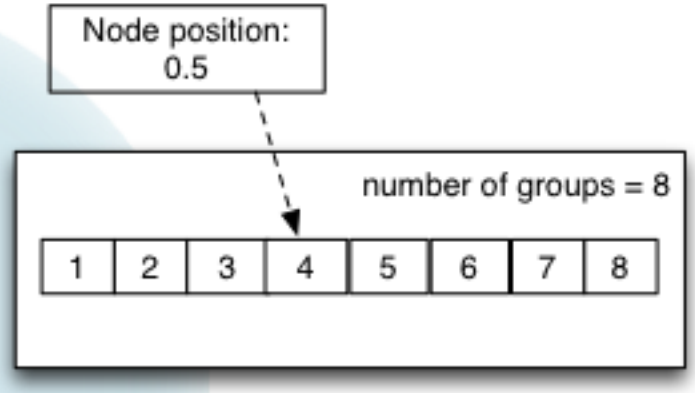
HASLab

Group construction

Group construction

Group calculation:

- **Each node has access to a number generator** that outputs number in the interval]0,1] **uniformly random across the entire system.**
- Each node, at startup, generates one of these numbers and stores it in a variable called **'position'**.
- Node position is **generated only once**, locally and never changes while the node is active.



group ← ceil(position * number of groups)

How to estimate 'number of groups'?

Group construction

Group size: user defined and constant

DataFlasks Node
group view = { }
Number of groups = 1

number of groups

1

Periodically

delivers some references for other nodes **alongside their 'position'**

PSS

HASLab

Group construction

number of groups

1

DataFlasks Node
group view = { a, b, … }
Number of groups = 1

Node A adds to its group all the received references because in this case everyone belongs to its group.

if (size of group view > group size):
  number of groups = number of groups * 2

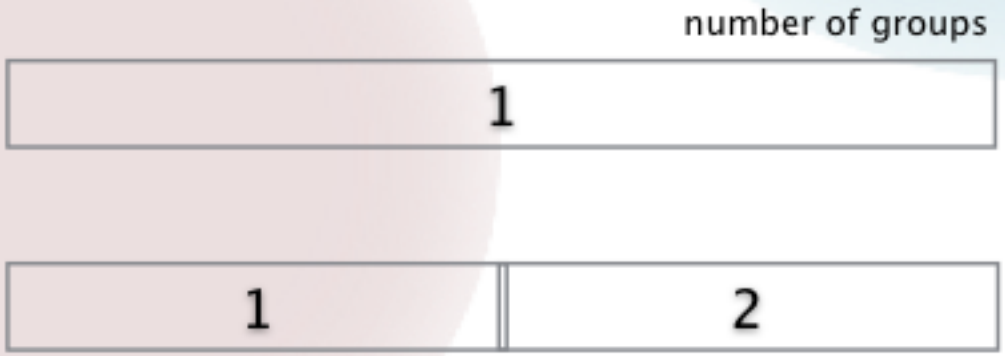
Split operation.

The size of 'group view' is the **local** estimation of the size of the group

HASLab

Group construction

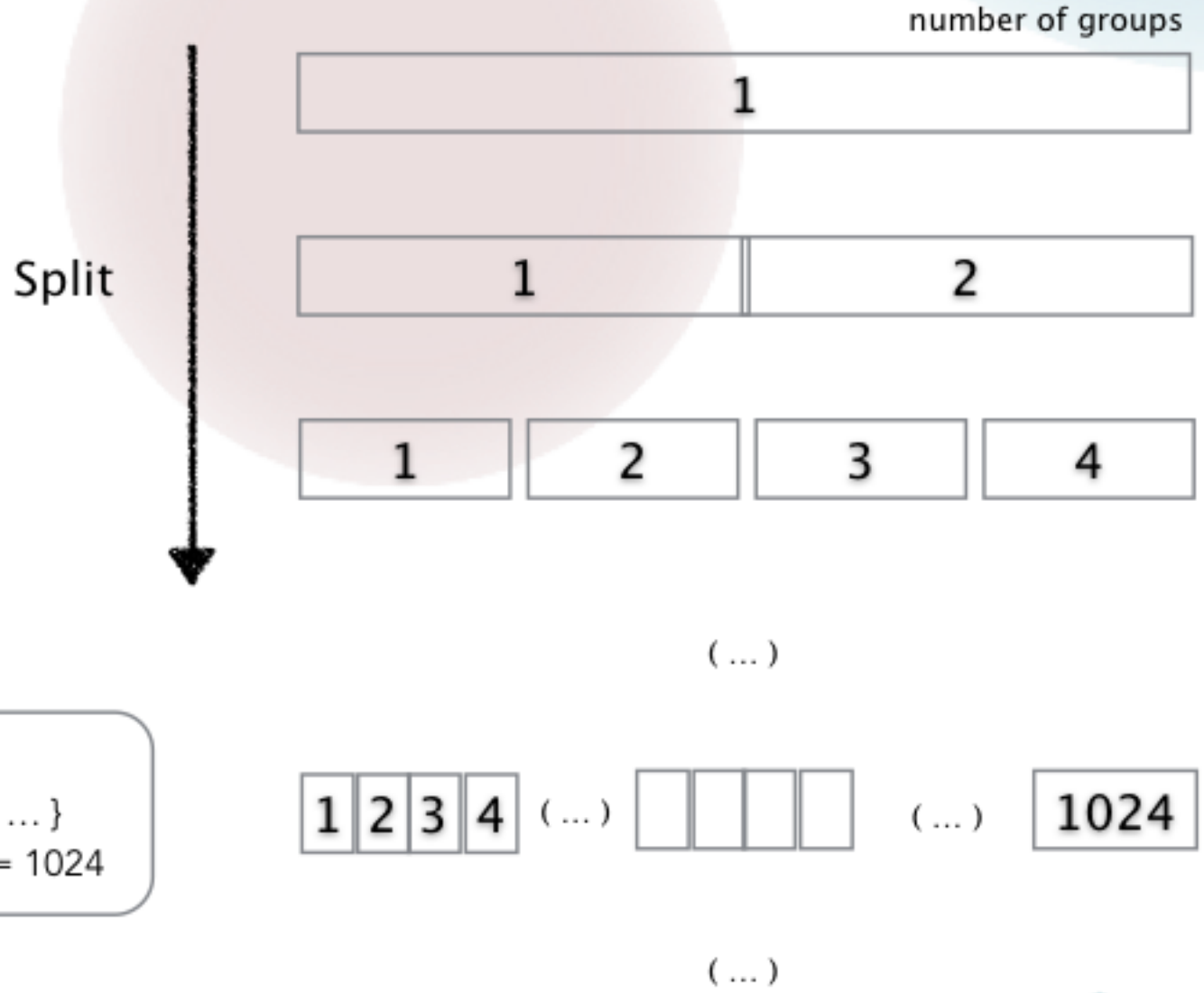
DataFlasks Node
group view = { a, b, … }
Number of groups = 2



The node's 'number of groups' estimate is now 2 groups.

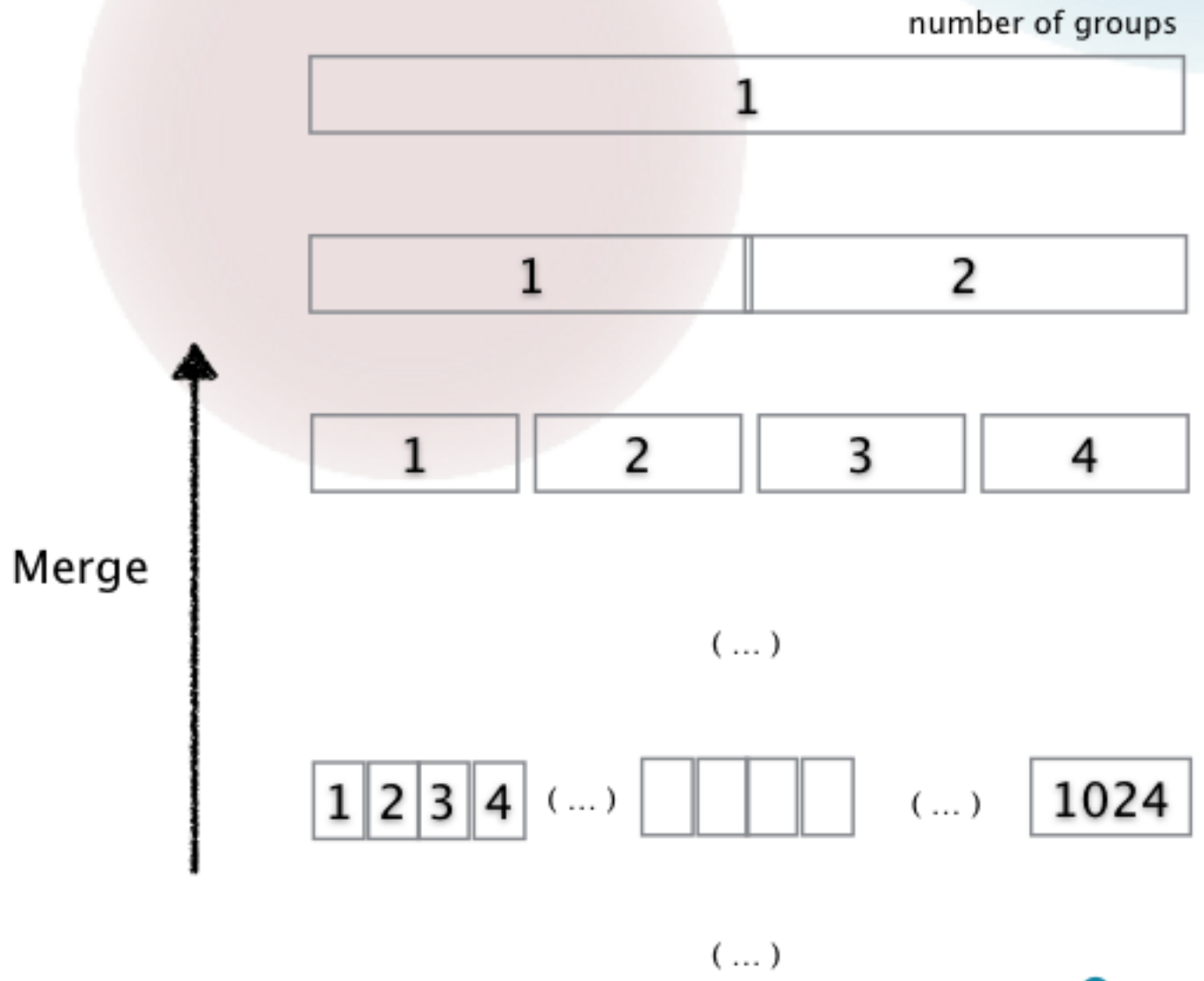
HASLab

Group construction



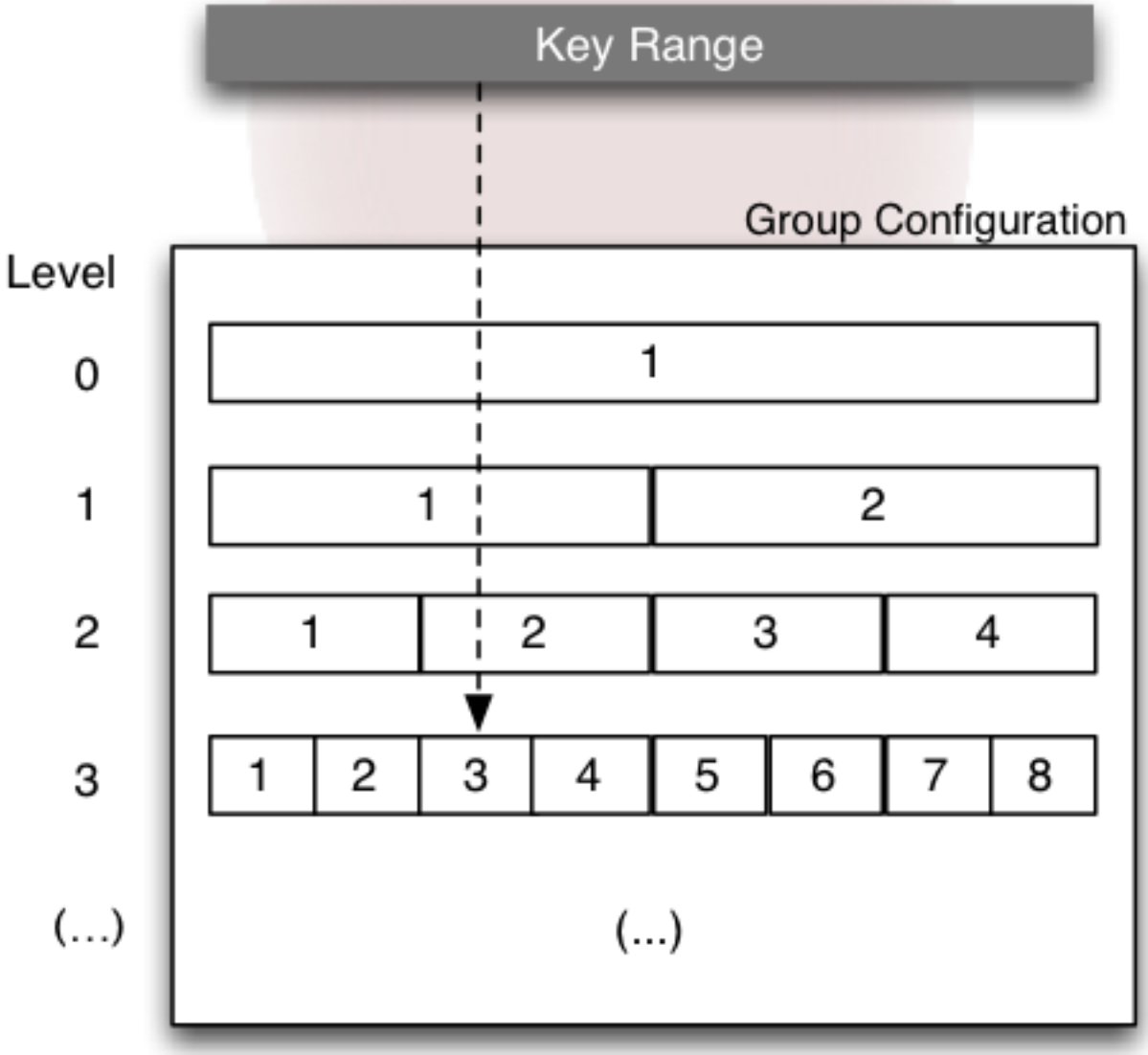
DataFlasks Node
group view = { a, b, ... }
Number of groups = 1024

Group construction



HASLab

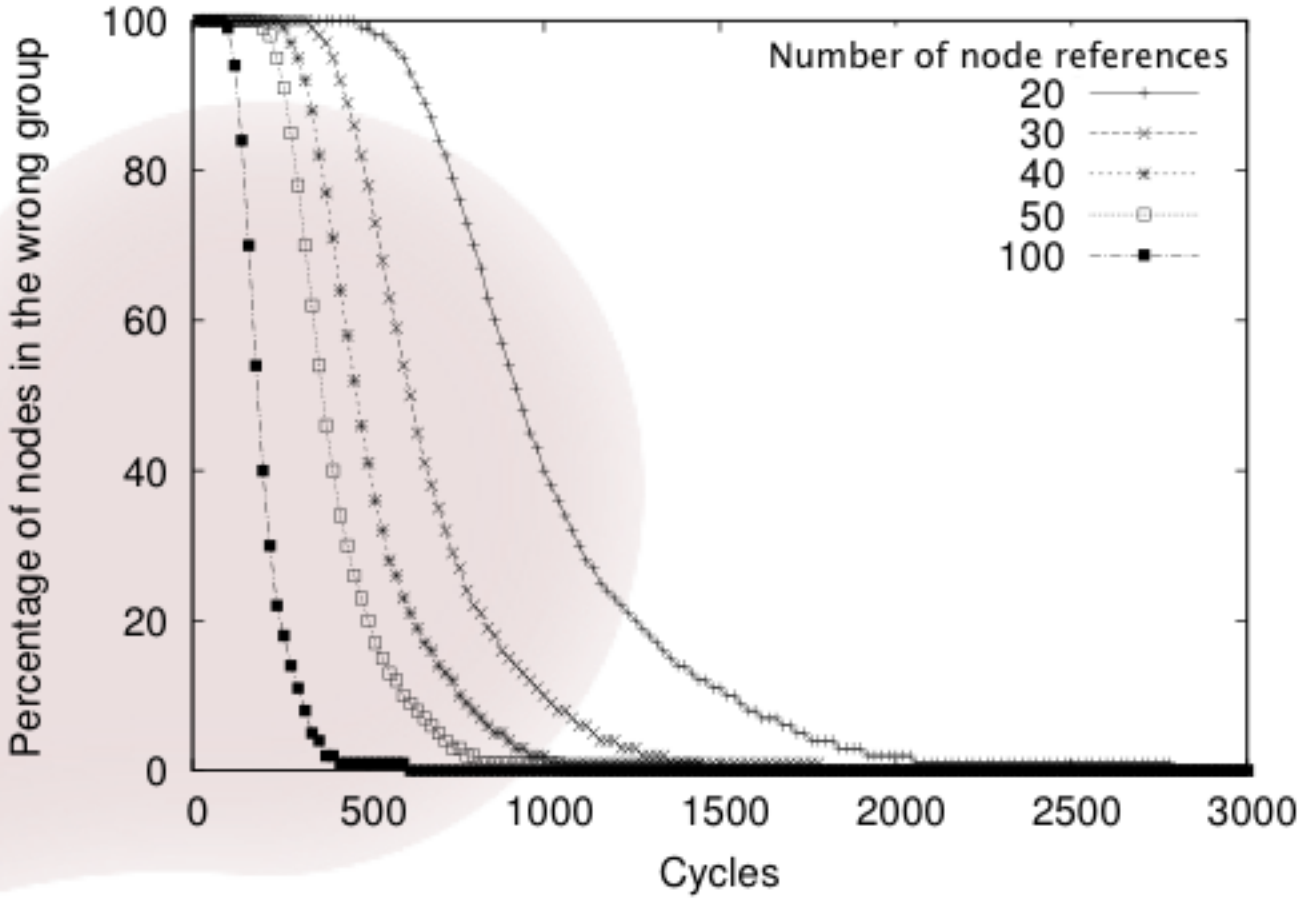
Group construction: why powers of 2?



HASLab

Group construction: convergence

~15K Nodes with variable number of node **references received each cycle from the PSS.**



HASLab

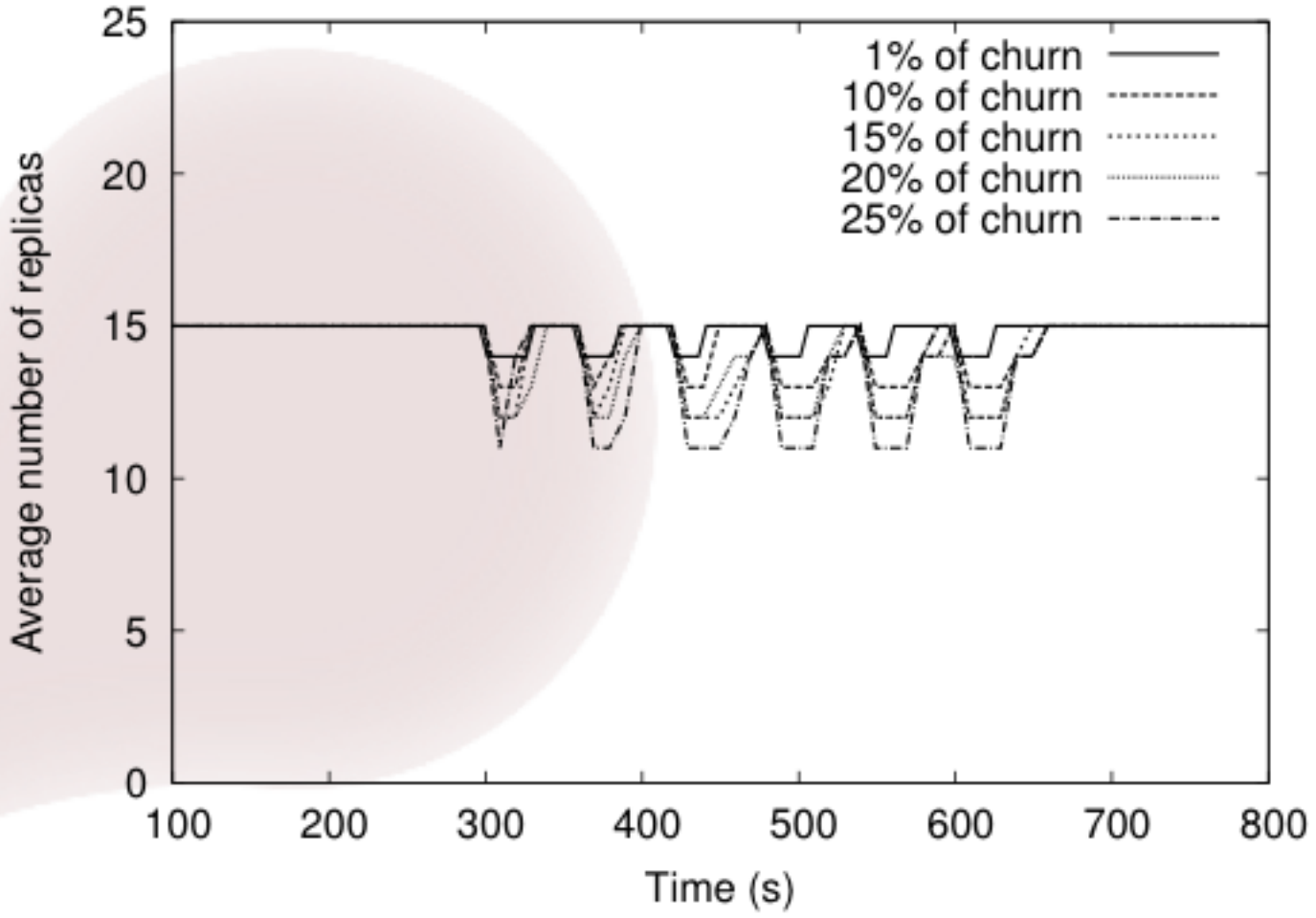
Proof of concept

- We have implemented a complete DataFlasks prototype
  - It is available at *github.com/fmaia/dataflasks*
- Evaluation on the Minha platform, which allows us to run multiple nodes in a single machine
  - Each of the nodes runs **real, ready for deployment code**

HASLab

Proof of concept

1K Nodes with variable churn percentage.



HASLab

Conclusion

- DataFlasks, an **epidemic data store for massive scale**:
  - **Radically** different from previous approaches
  - Entirely based on **gossip** protocols
  - Inherently **scalable and resilient** even in environments with **very high dynamism**
- A **novel group construction** epidemic algorithm
- A new distributed systems slicing protocol: **Slead**
- Both are independent of the main design and usable in other contexts and applications

HASLab

Results

- Core results:
  - **Slead: Low-Memory, Steady Distributed Systems Slicing. DAIS, 2012**

Francisco Maia, Miguel Matos, Etienne Rivière and Rui Oliveira

- **Slicing as a Distributed Systems Primitive. LADC, 2013**
  Francisco Maia, Miguel Matos, Etienne Rivière and Rui Oliveira
- **DataFlasks: an Epidemic Dependable Key-Value Substrate. DCDV, 2013**
  Francisco Maia, Miguel Matos, Ricardo Vilaça, José Pereira, Rui Oliveira and Etienne Rivière
- **DataFlasks: Epidemic Store for Massive Scale Systems. SRDS, 2014**
  Francisco Maia, Miguel Matos, Ricardo Vilaça, José Pereira, Rui Oliveira and Etienne Rivière

- Other results:
  - **Scalable Transactions in the Cloud: Partitioning Revisited. DOA, 2010**
    Francisco Maia, José Enrique Armendáriz-Iñigo, M. Idoia Ruiz-Fuertes and Rui Oliveira
  - **About the Feasibility of Transactional Support in Cloud Computing. EDCC, 2010**
    Francisco Maia, Rui Oliveira and José Enrique Armendáriz-Iñigo
  - **Worldwide Consensus. DAIS, 2011**
    Francisco Maia, Miguel Matos, José Pereira and Rui Oliveira
  - **MeT: Workload Aware Elasticity for NoSQL. Eurosys, 2013**
    Francisco Cruz, Francisco Maia, João Paulo, Miguel Matos, Ricardo Vilaça, José Pereira and Rui Oliveira
  - **Autonomous Multi-dimensional Slicing for Large-Scale Distributed Systems. DAIS 2014**
    Mathieu Pasquet, Francisco Maia, Etienne Rivière and Valerio Schiavoni
  - **Workload-aware Table Splitting for NoSQL. SAC 2014**
    Francisco Cruz, Francisco Maia, Rui Oliveira and Ricardo Vilaça
  - **Practical Evaluation of Large Scale Applications. DAIS 2015 (To appear)**
    Tiago Jorge, Francisco Maia, Miguel Matos, José Pereira and Rui Oliveira

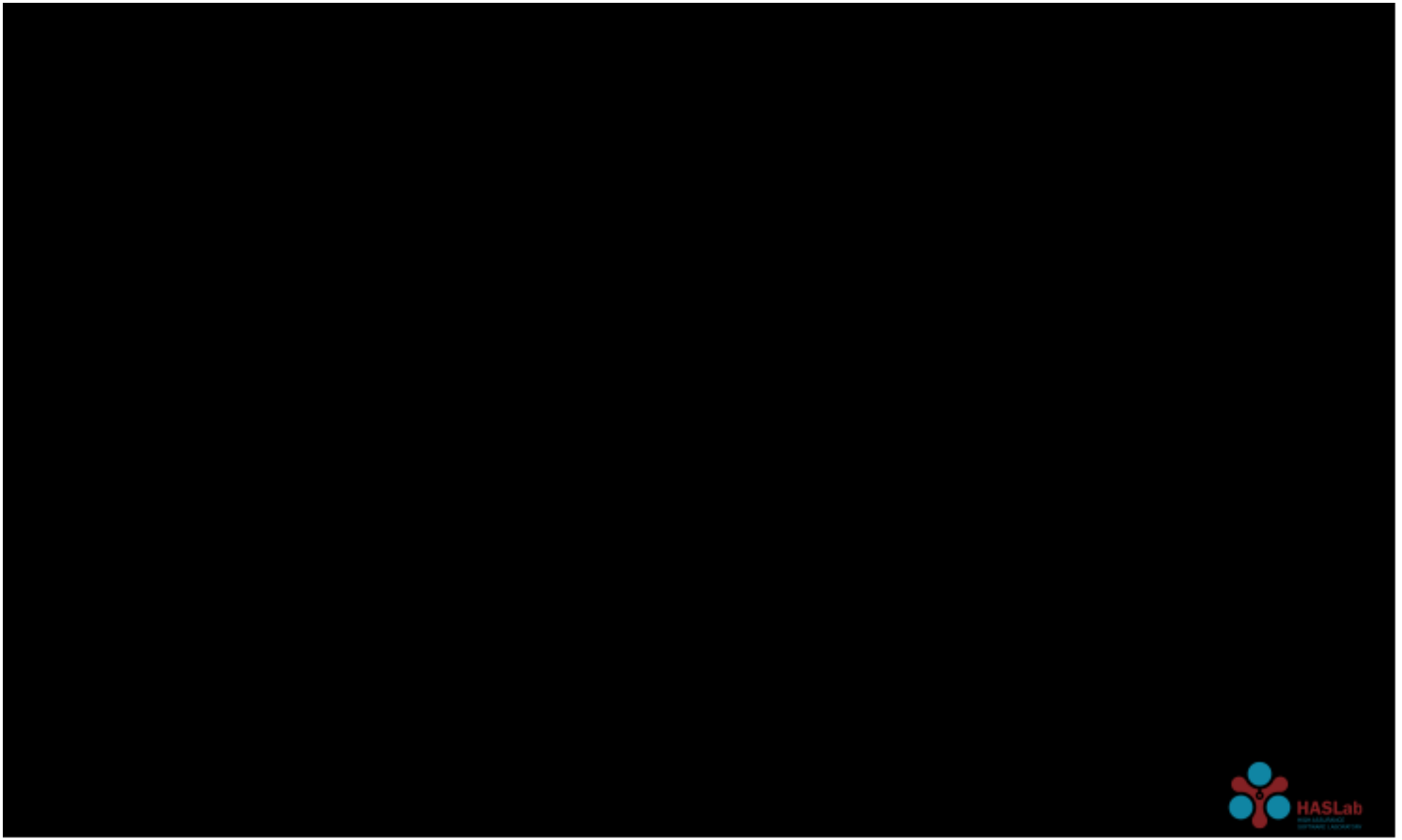
HASLab

HASLab
HIGH-ASSURANCE

Epidemic store for massive scale systems

Francisco Maia
Advisor: Rui Oliveira

Braga, 30.04.2015


HASLab