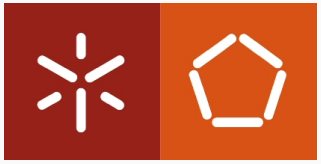


Efficient Synchronization of State-based CRDTs

Vitor Enes
Carlos Baquero (Universidade do Minho)
João Leitão (Universidade Nova de Lisboa)

Tuesday 3rd October, 2017

Outline

1. Data Replication

2. Conflict-free Replicated Data Types
   - State-based CRDTs
   - Efficient Synchronization
   - Results

Data Replication

Data Replication

- Improves **availability**
  - available even when some replicas are unreachable
- Improves **performance**
  - *reduced latency*: geo-distributed replicas give us low-latency links
  - *increased throughput*: multiple replicas serving data

Data Replication

- Improves **availability**
  - available even when some replicas are unreachable
- Improves **performance**
  - *reduced latency*: geo-distributed replicas give us low-latency links
  - *increased throughput*: multiple replicas serving data
- Several consistency models, e.g.:
  - **Strong Consistency**:
    - illusion of a single-copy by synchronizing replicas on each update
    - very successful within the data center but
    - wide-area networks increase latency of requests and decrease system's throughput
  - **Eventual Consistency**:
    - synchronize replicas in the background
    - replicas may diverge

Conflict-free Replicated Data Types


akka


TRIFORK.
...think software


TOMTOM


bet365


basho


riak


Lasp

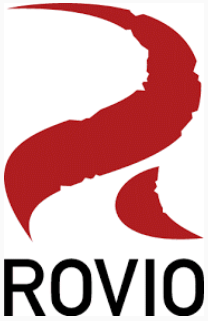

AntidoteDB


SOUNDCLOUD


PROJECTFIFO


HELIUM


Tapjoy

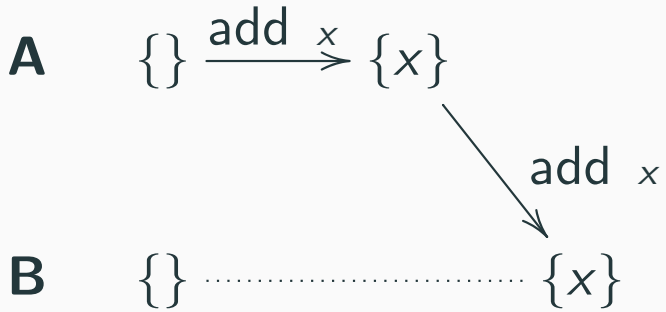

ROVIO


ANGRY BIRDS

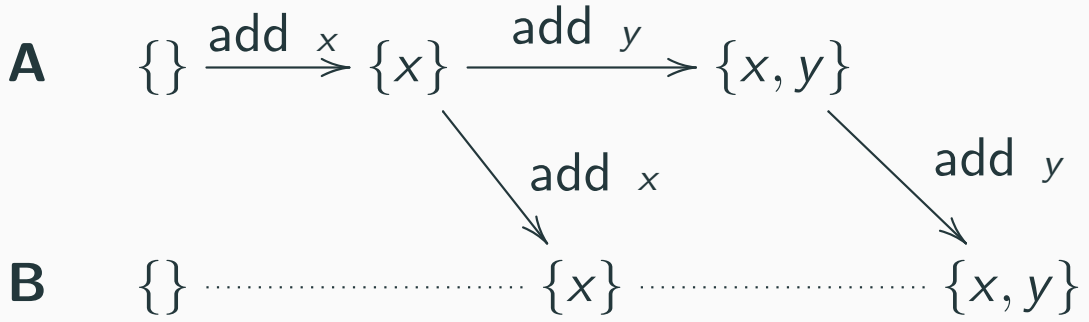

Riot
GAMES


LEAGUE of LEGENDS

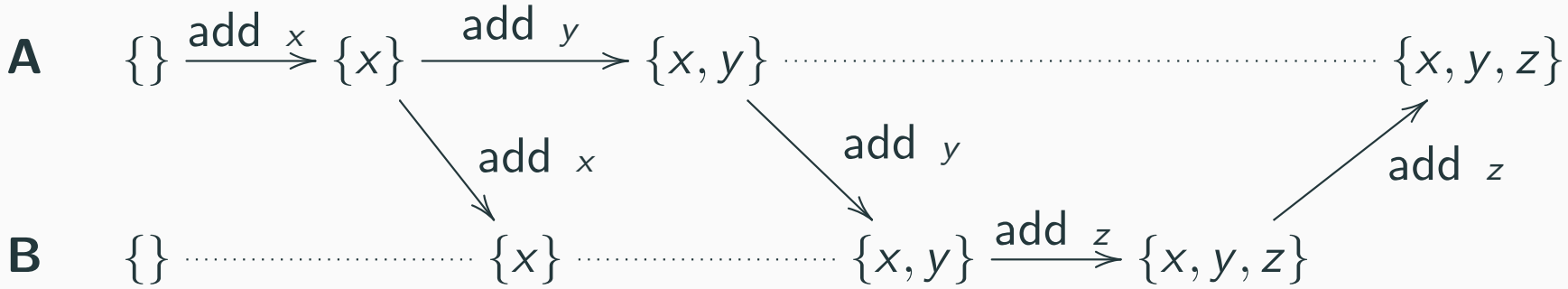
Operation-based



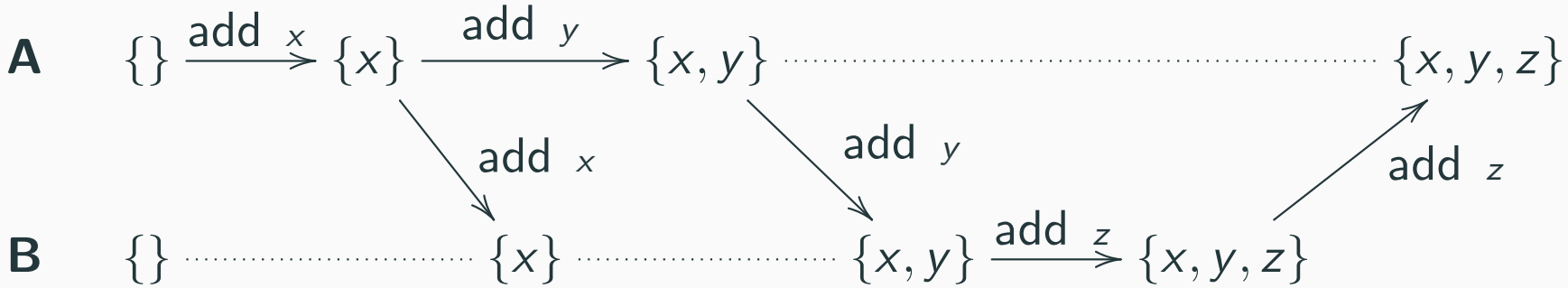
Operation-based



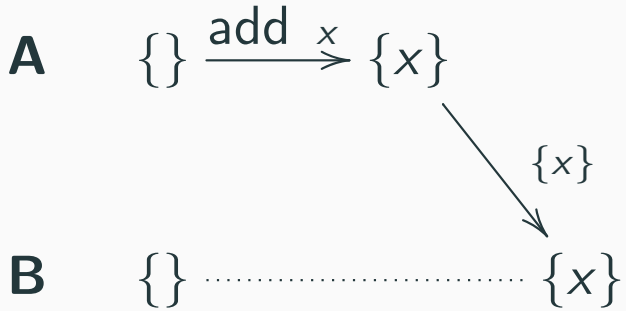
Operation-based



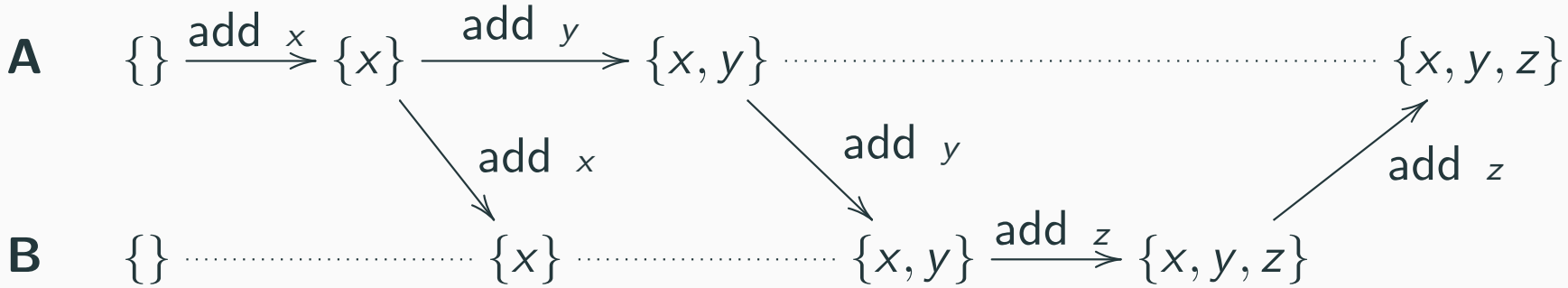
Operation-based



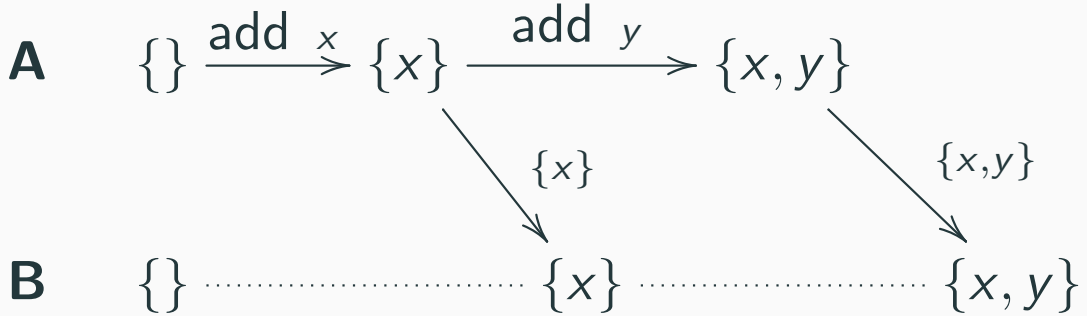
State-based



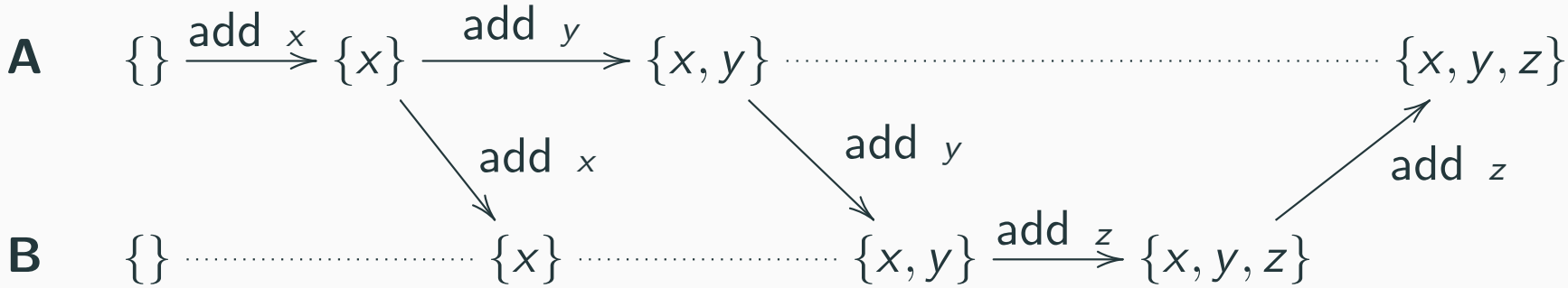
Operation-based



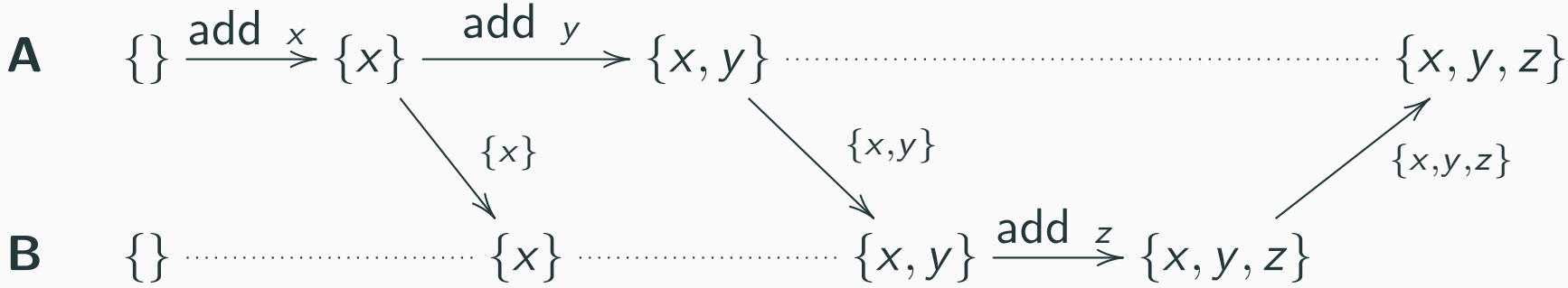
State-based



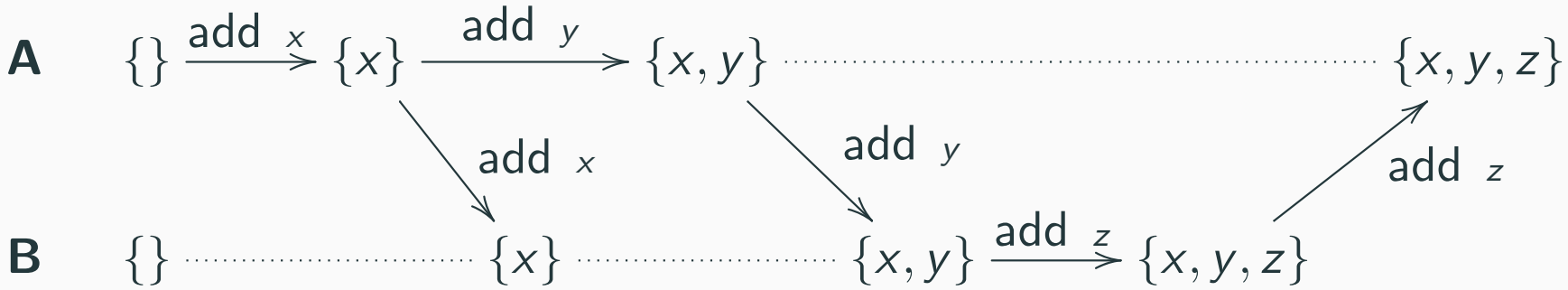
Operation-based



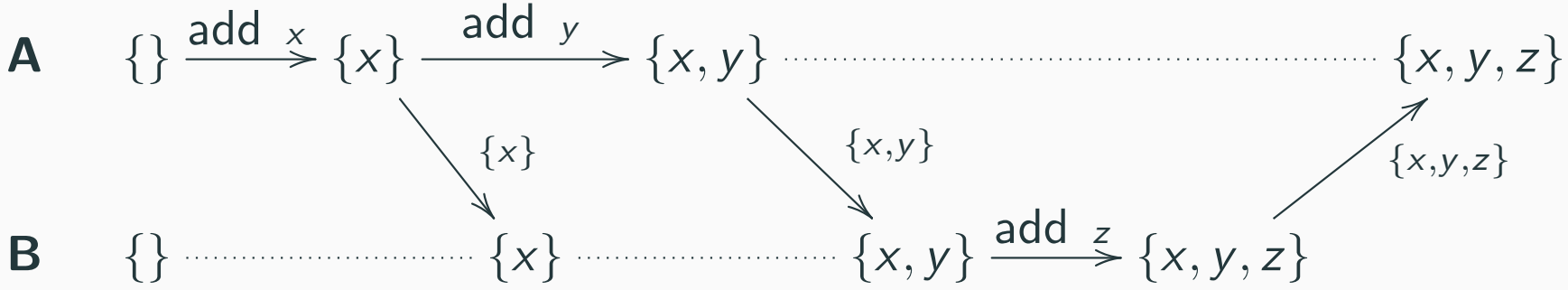
State-based



Operation-based



State-based



$ $ full state transmission $ $

How can we avoid replica A sending the full state to replica B?

How can we avoid replica A sending the full state to replica B?

Two fundamental problems

1. **A knows something about the state of B:**
   - **Goal: A** to compute a delta (δ) of minimum size to be sent to **B**

How can we avoid replica A sending the full state to replica B?

Two fundamental problems

1. **A knows something about the state of B:**
   - **Goal: A** to compute a delta (δ) of minimum size to be sent to **B**
   - Techniques:
     - *avoid sending acknowledged entries in δ-buffer* (**Classic Delta-based CRDTs**)
     - ...
     - ...

How can we avoid replica A sending the full state to replica B?

Two fundamental problems

1. **A knows something about the state of B:**
   - **Goal: A** to compute a delta (δ) of minimum size to be sent to **B**
   - Techniques:
     - *avoid sending acknowledged entries in δ-buffer* (**Classic Delta-based CRDTs**)
     - ...
     - ...
2. **A knows nothing:**
   - **Goal:** design a protocol that minimizes state transmission

How can we avoid replica A sending the full state to replica B?

Two fundamental problems

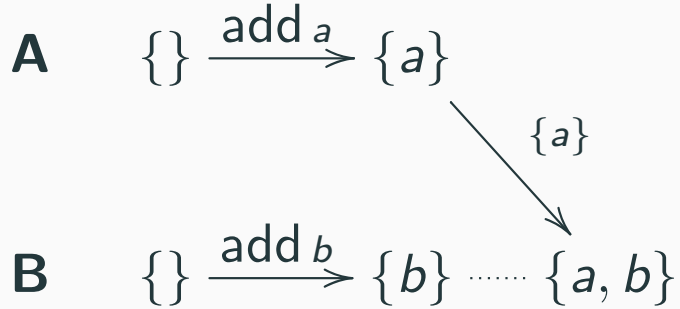
1. **A knows something about the state of B:**
   - **Goal: A** to compute a delta (δ) of minimum size to be sent to **B**
   - Techniques:
     - *avoid sending acknowledged entries in δ-buffer* (**Classic Delta-based CRDTs**)
     - ...
     - ...
2. **A knows nothing:**
   - **Goal:** design a protocol that minimizes state transmission
   - Techniques:
     - ...
     - ...

Solving the first problem

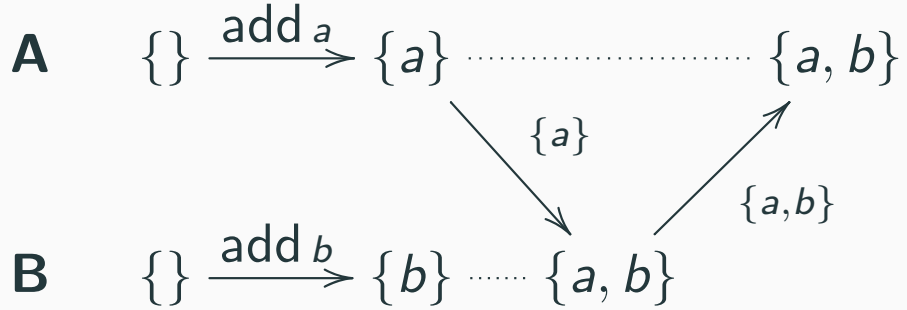
A $\{\} \xrightarrow{\text{add } a} \{a\}$

B $\{\} \xrightarrow{\text{add } b} \{b\}$

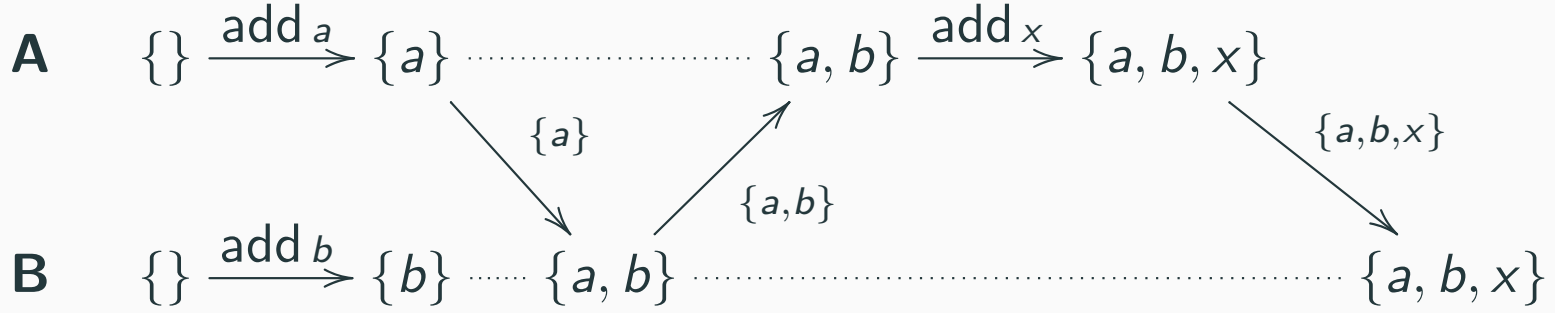
Solving the first problem



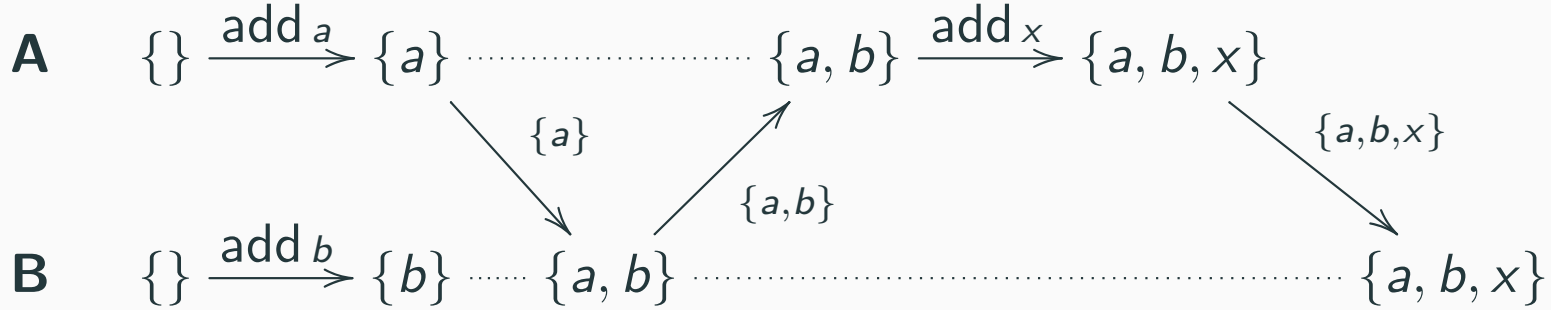
Solving the first problem



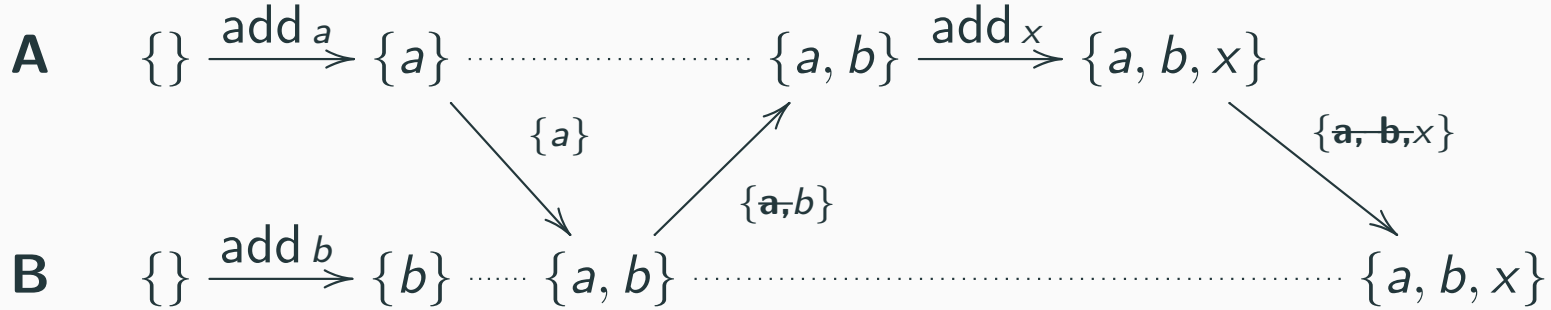
Solving the first problem



Solving the first problem



Avoid back-propagation of δ-groups



How can we avoid replica A sending the full state to replica B?

Two fundamental problems

1. **A knows something about the state of B:**
   - **Goal: A** to compute a delta (δ) of minimum size to be sent to **B**
   - Techniques:
     - *avoid sending acknowledged entries in δ-buffer* (**Classic Delta-based CRDTs**)
     - **avoid back-propagation of δ-groups** (**BP**)
     - ...
2. **A knows nothing:**
   - **Goal:** design a protocol that minimizes state transmission
   - Techniques:
     - ...
     - ...

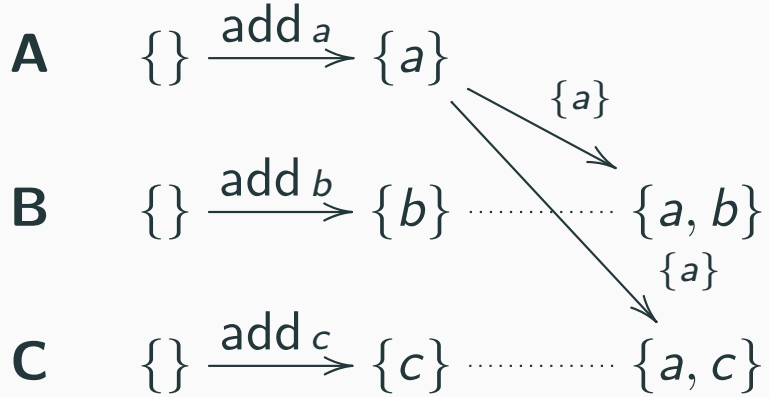
Solving the first problem

A $\{\} \xrightarrow{\text{add } a} \{a\}$

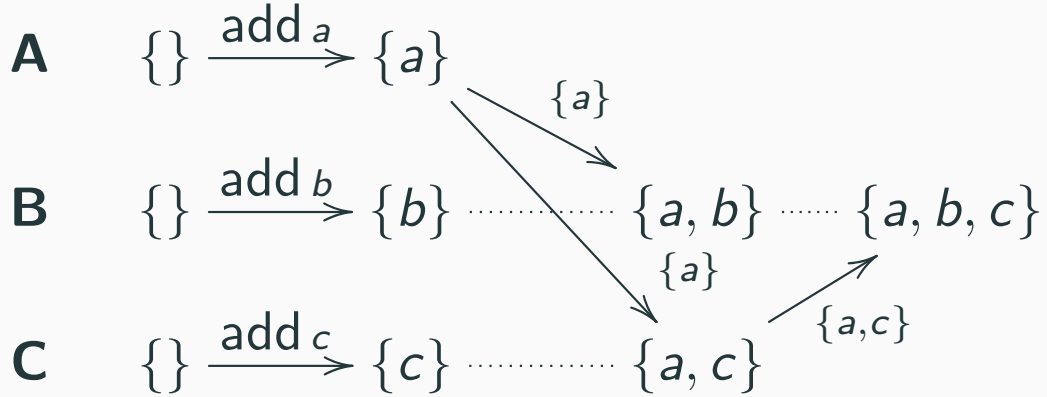
B $\{\} \xrightarrow{\text{add } b} \{b\}$

C $\{\} \xrightarrow{\text{add } c} \{c\}$

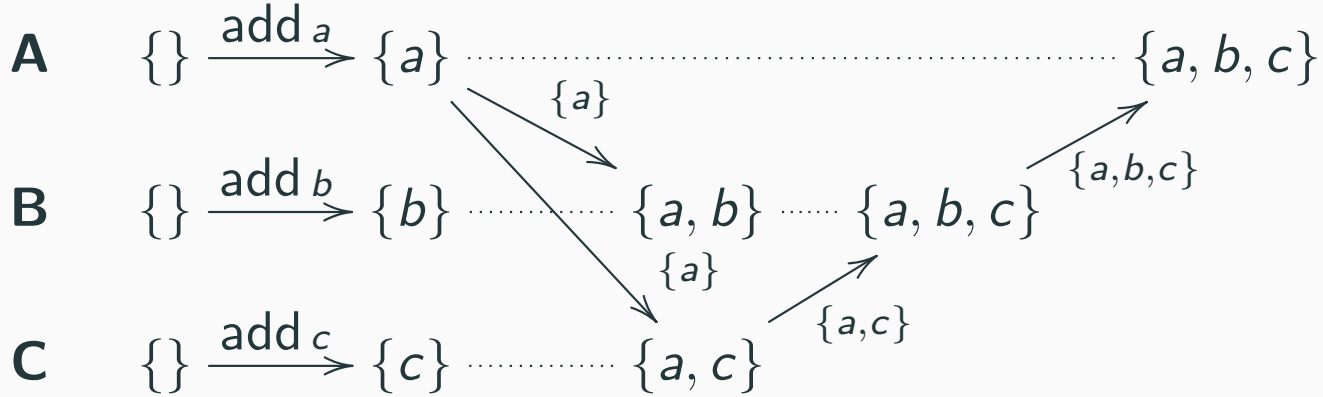
Solving the first problem



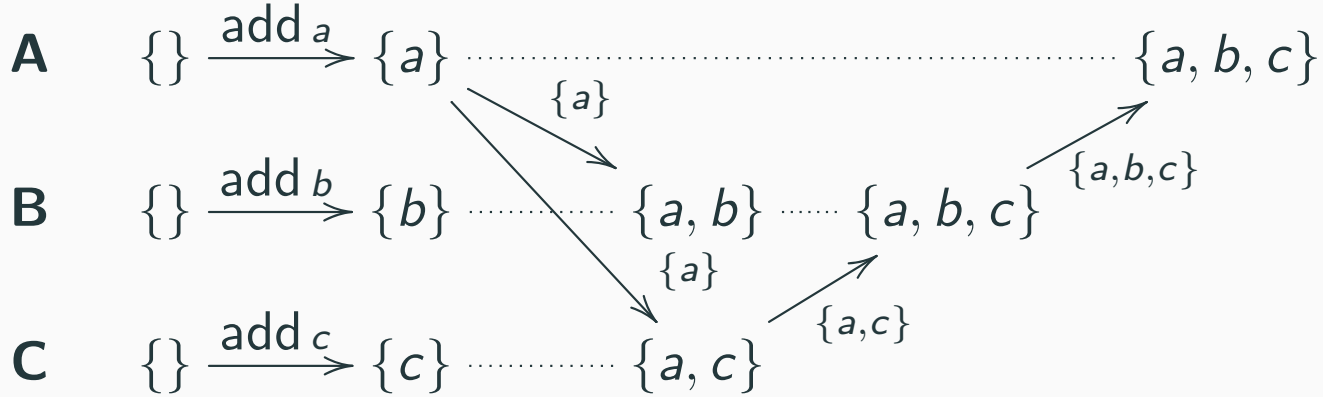
Solving the first problem



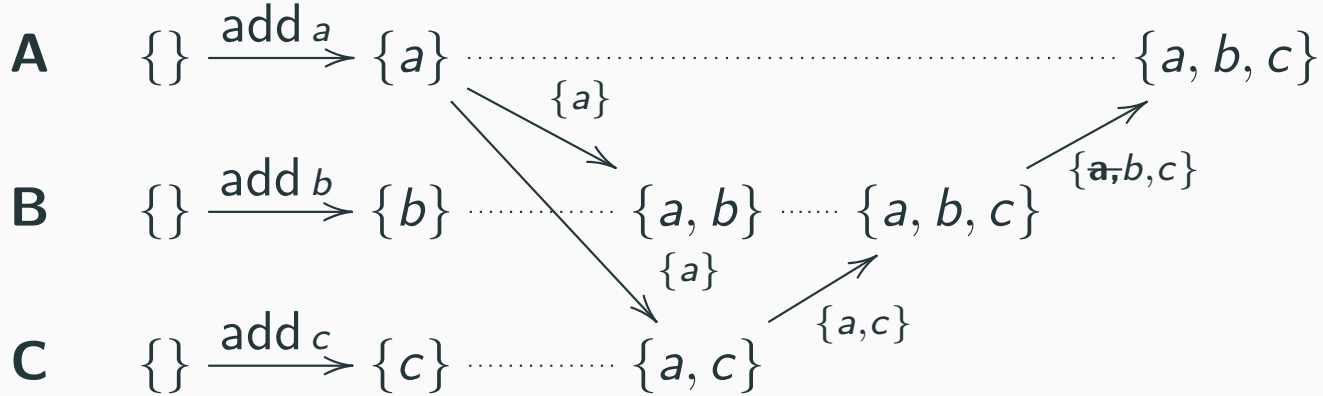
Solving the first problem



Solving the first problem



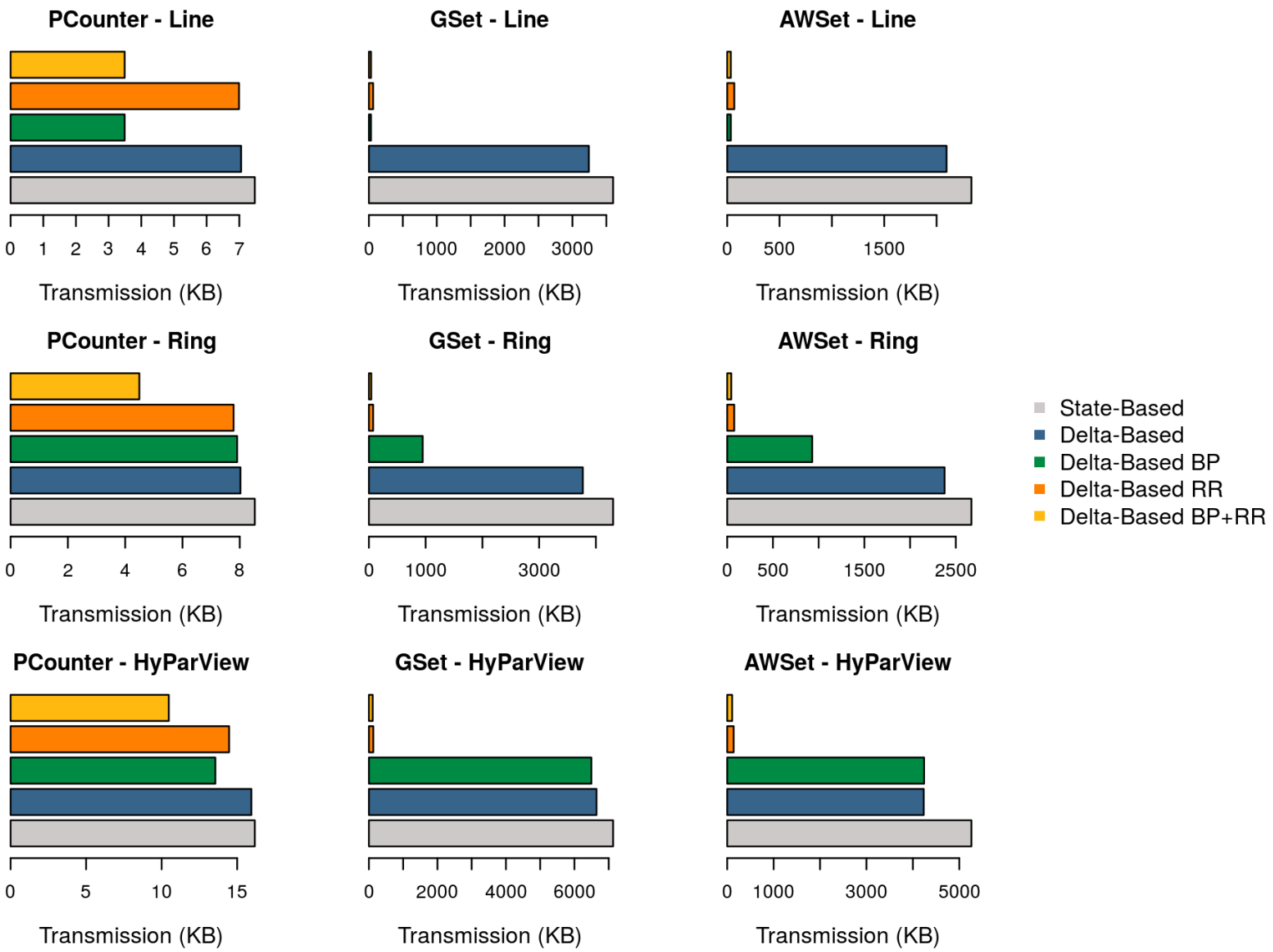
Remove redundant state in δ-groups



How can we avoid replica A sending the full state to replica B?

Two fundamental problems

1. **A knows something about the state of B:**
   - **Goal: A** to compute a delta (δ) of minimum size to be sent to **B**
   - Techniques:
     - *avoid sending acknowledged entries in δ-buffer* (**Classic Delta-based CRDTs**)
     - *avoid back-propagation of δ-groups* (**BP**)
     - **remove redundant state in δ-groups** (**RR**)
2. **A knows nothing:**
   - **Goal:** design a protocol that minimizes state transmission
   - Techniques:
     - ...
     - ...


PCounter - Line
0 1 2 3 4 5 6 7
Transmission (KB)
GSet - Line
0 1000 2000 3000
Transmission (KB)
AWSet - Line
0 500 1500
Transmission (KB)
PCounter - Ring
0 2 4 6 8
Transmission (KB)
GSet - Ring
0 1000 3000
Transmission (KB)
AWSet - Ring
0 500 1500 2500
Transmission (KB)
PCounter - HyParView
0 5 10 15
Transmission (KB)
GSet - HyParView
0 2000 4000 6000
Transmission (KB)
AWSet - HyParView
0 1000 3000 5000
Transmission (KB)
State-Based
Delta-Based
Delta-Based BP
Delta-Based RR
Delta-Based BP+RR

Solving the second problem

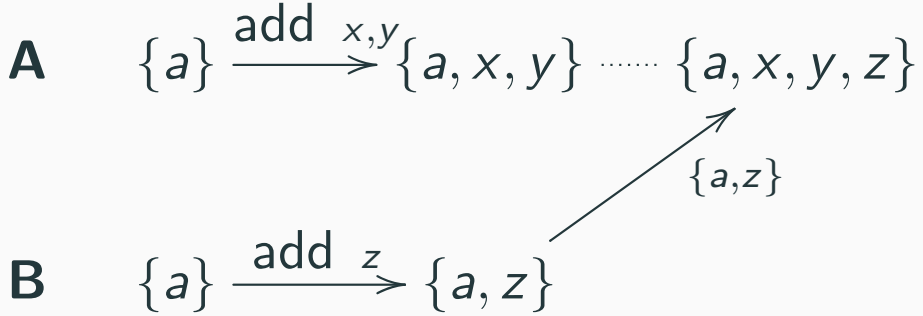
State-driven synchronization

A $\{a\} \xrightarrow{\text{add } x,y} \{a, x, y\}$

B $\{a\} \xrightarrow{\text{add } z} \{a, z\}$

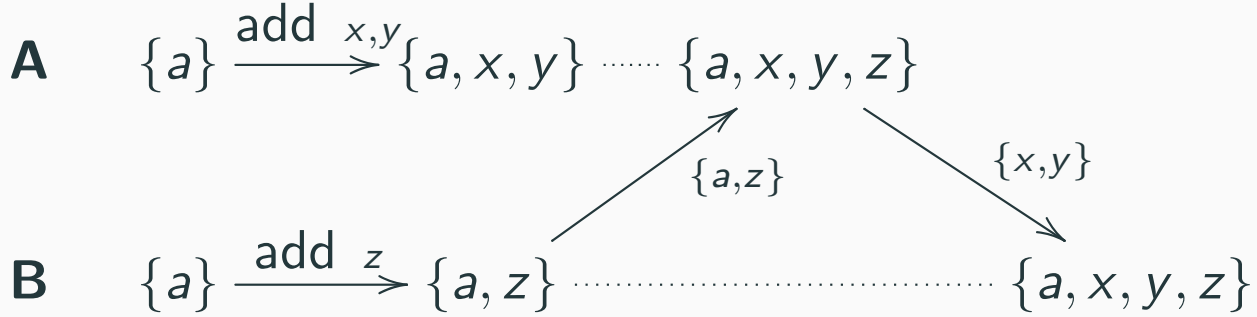
Solving the second problem

State-driven synchronization



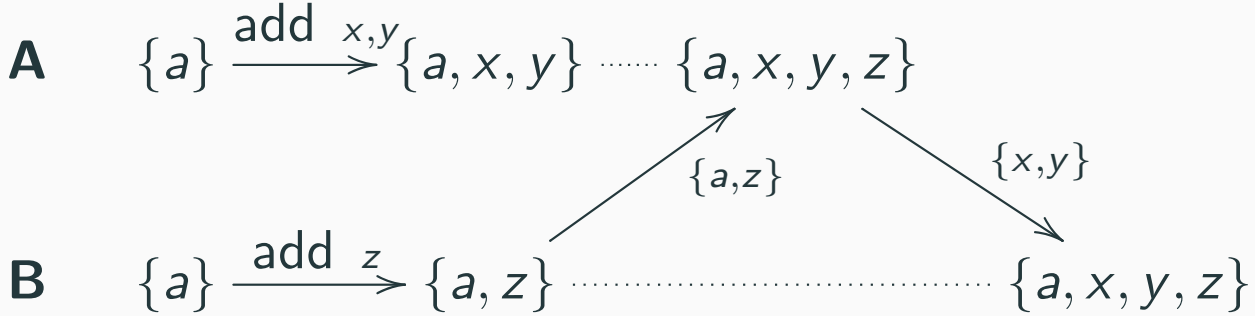
Solving the second problem

State-driven synchronization



Solving the second problem

State-driven synchronization



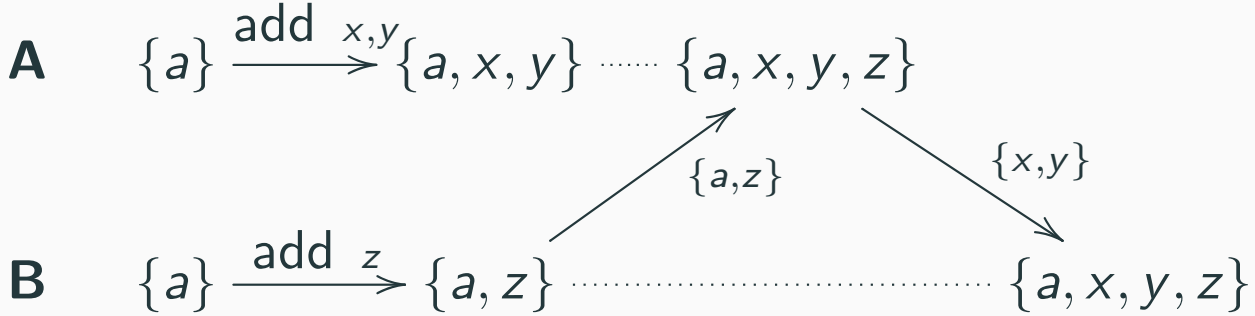
Digest-driven synchronization

A $\{a\} \xrightarrow{\text{add } x,y} \{a,x,y\}$

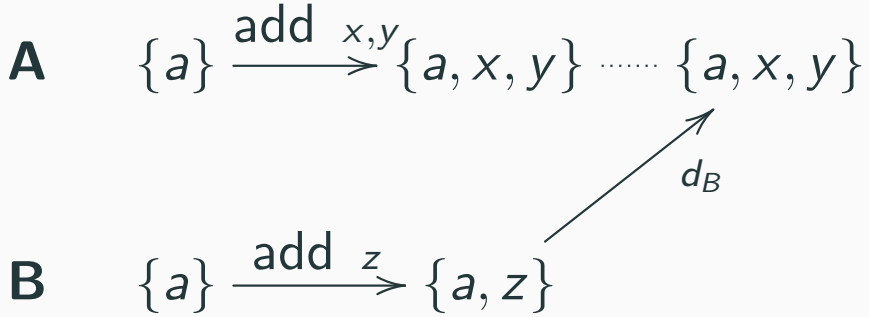
B $\{a\} \xrightarrow{\text{add } z} \{a,z\}$

Solving the second problem

State-driven synchronization

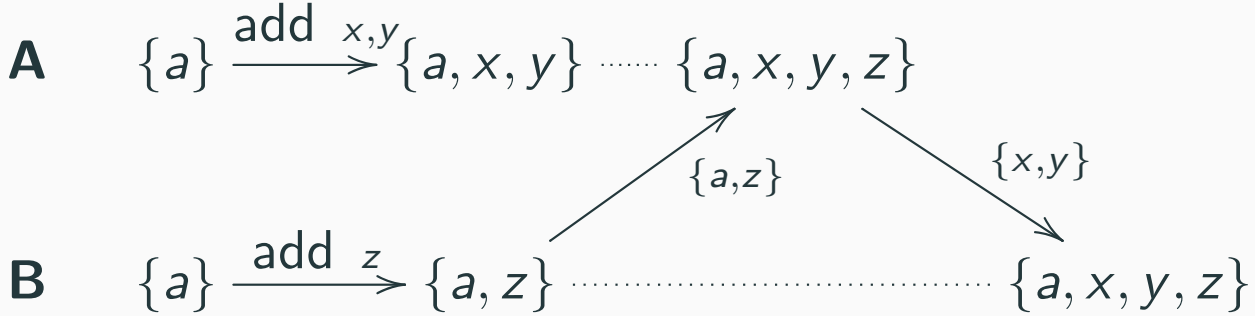


Digest-driven synchronization

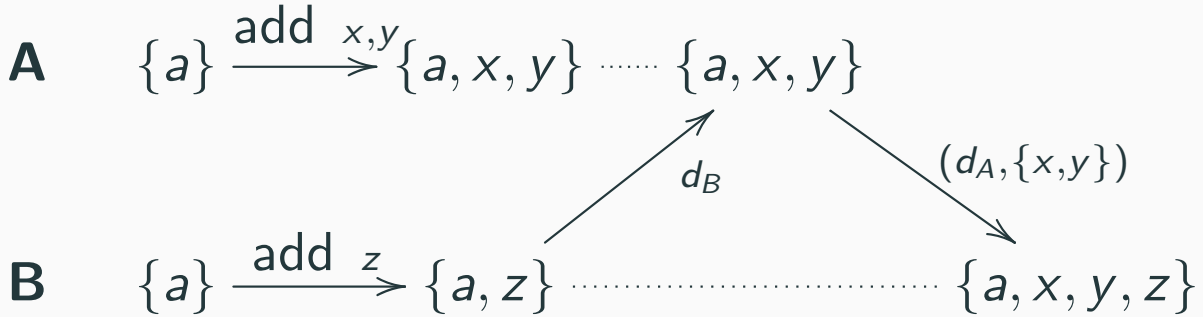


Solving the second problem

State-driven synchronization

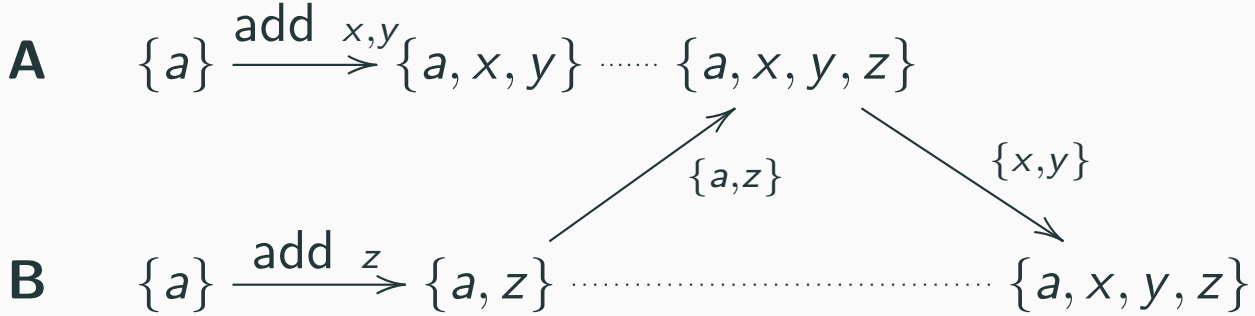


Digest-driven synchronization

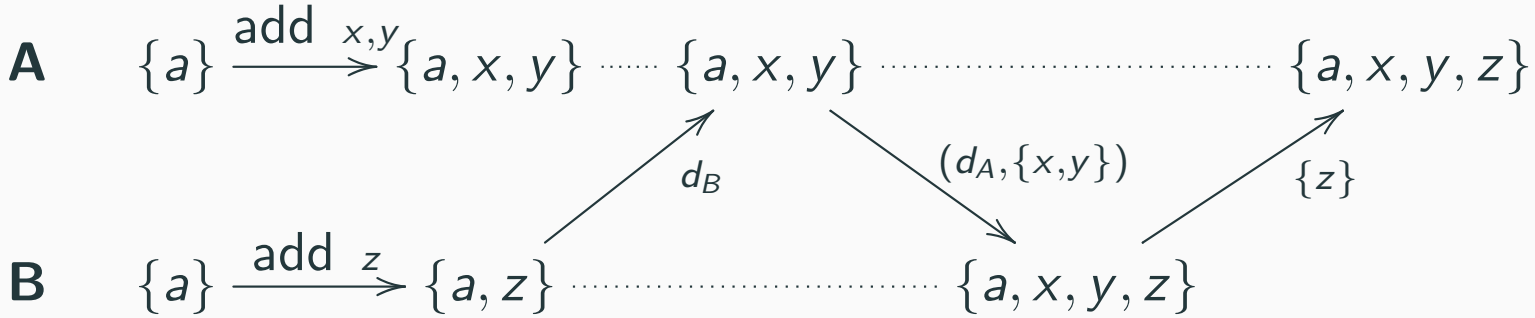


Solving the second problem

State-driven synchronization



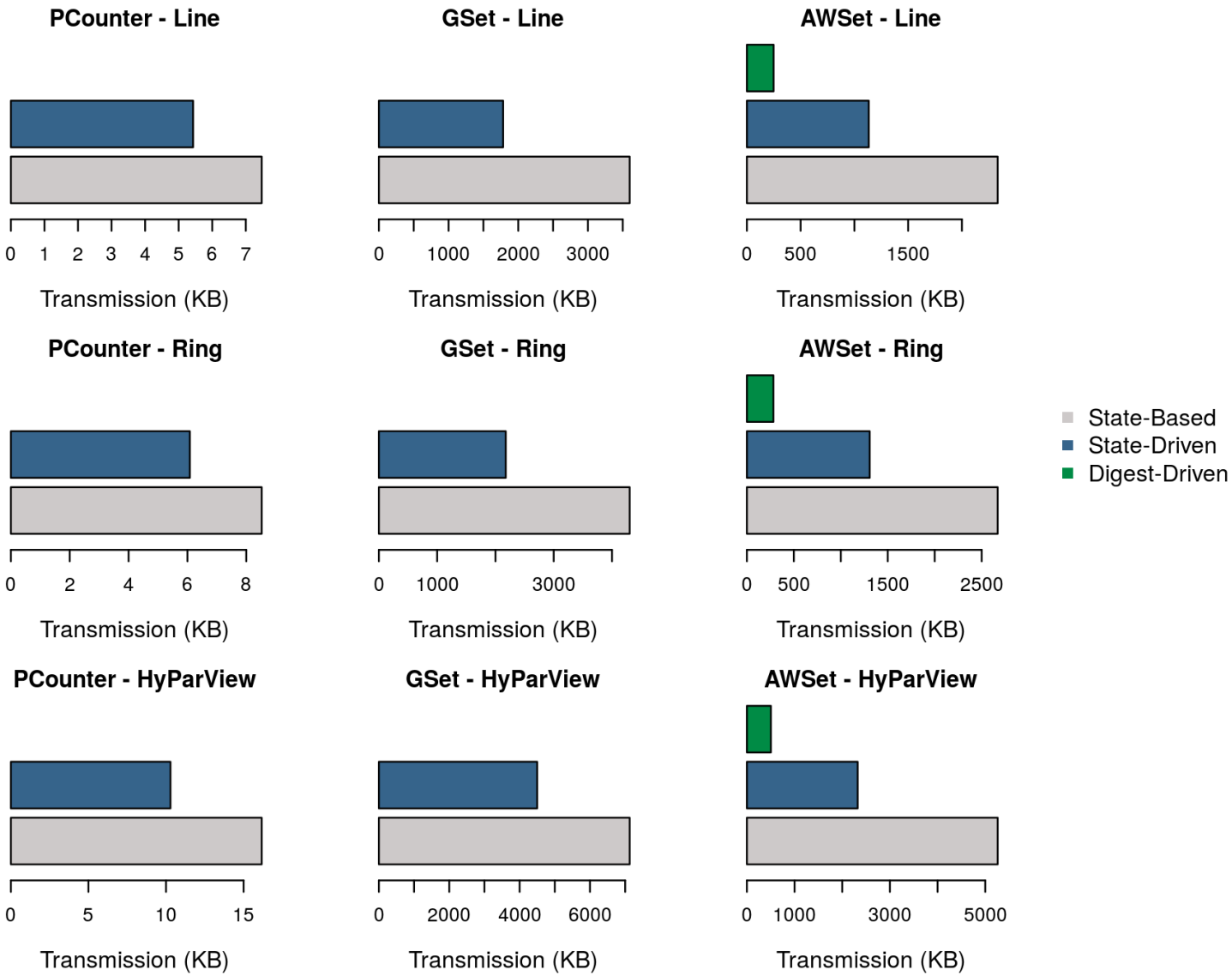
Digest-driven synchronization



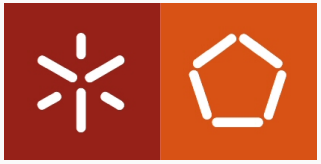
How can we avoid replica A sending the full state to replica B?

Two fundamental problems

1. **A knows something about the state of B:**
   - **Goal: A** to compute a delta (δ) of minimum size to be sent to **B**
   - Techniques:
     - *avoid sending acknowledged entries in δ-buffer* (**Classic Delta-based CRDTs**)
     - *avoid back-propagation of δ-groups* (**BP**)
     - *remove redundant state in δ-groups* (**RR**)
2. **A knows nothing:**
   - **Goal:** design a protocol that minimizes state transmission
   - Techniques:
     - **state-driven synchronization**
     - **digest-driven synchronization**


PCounter - Line
0 1 2 3 4 5 6 7
Transmission (KB)
GSet - Line
0 1000 2000 3000
Transmission (KB)
AWSet - Line
0 500 1500
Transmission (KB)
PCounter - Ring
0 2 4 6 8
Transmission (KB)
GSet - Ring
0 1000 3000
Transmission (KB)
AWSet - Ring
0 500 1500 2500
Transmission (KB)
PCounter - HyParView
0 5 10 15
Transmission (KB)
GSet - HyParView
0 2000 4000 6000
Transmission (KB)
AWSet - HyParView
0 1000 3000 5000
Transmission (KB)
State-Based
State-Driven
Digest-Driven

bit.ly/msc-presentation



Efficient Synchronization of State-based CRDTs

Vitor Enes
Carlos Baquero (Universidade do Minho)
João Leitão (Universidade Nova de Lisboa)

Tuesday 3rd October, 2017