Effective and Efficient Compromise Recovery for Weakly Consistent Replication

Prince Mahajan (UT Austin),
Ramakrishna Kotla, Cathy Marshall,
Venugopalan "Rama" Ramasubramanian, Tom Rodeheffer,
Doug Terry, Ted Wobber
(Microsoft Research Silicon Valley)

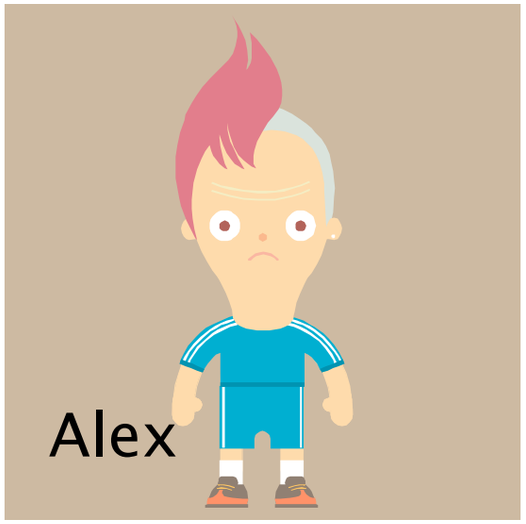
Scenario

Scenario


Sam

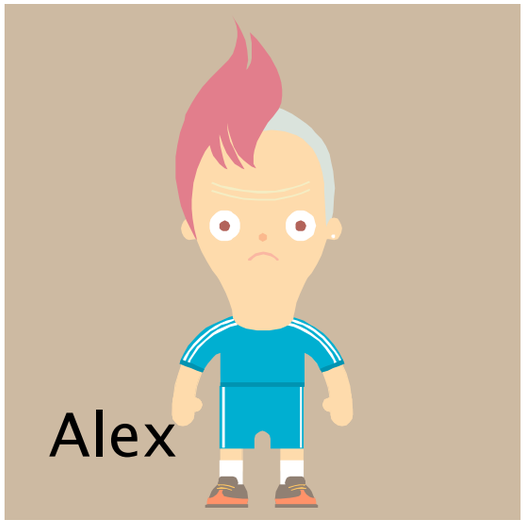

iPod
1:25 PM
Contacts
Notes
Calculator
Settings
Weather
Maps
YouTube
Videos
Clock
Stocks
Photos
Beta
NOW
NYTimes
Music
9

Scenario

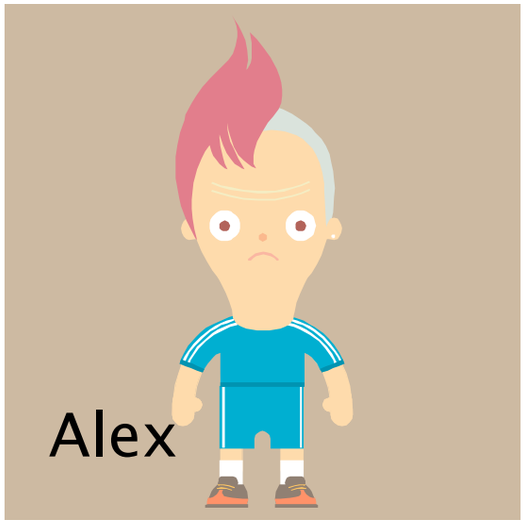

Alex

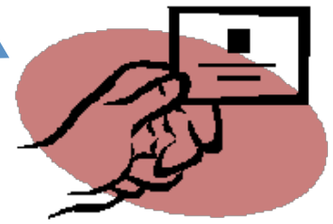

iPod
1:25 PM
Contacts
Notes
Calculator
Settings
Weather
Maps
YouTube
Videos
Clock
Stocks
Photos
Bank
NOW
NYTimes
Music
9

Scenario



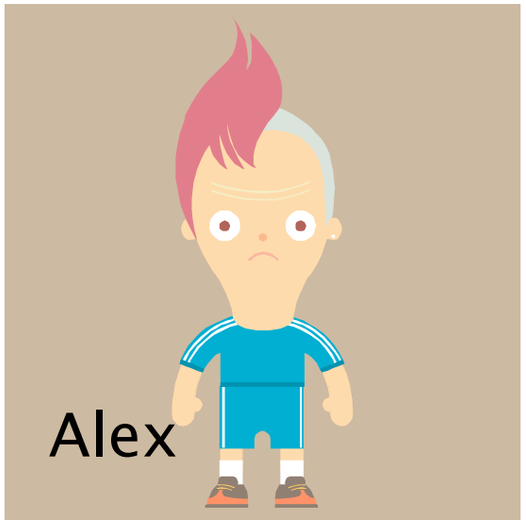
Scenario

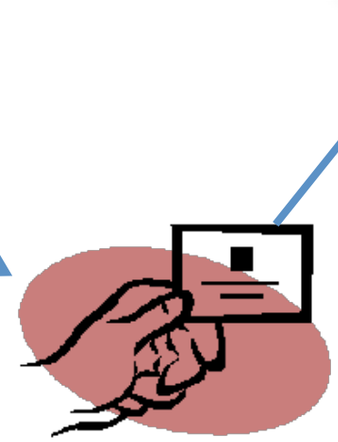




Corrupted
“contact”

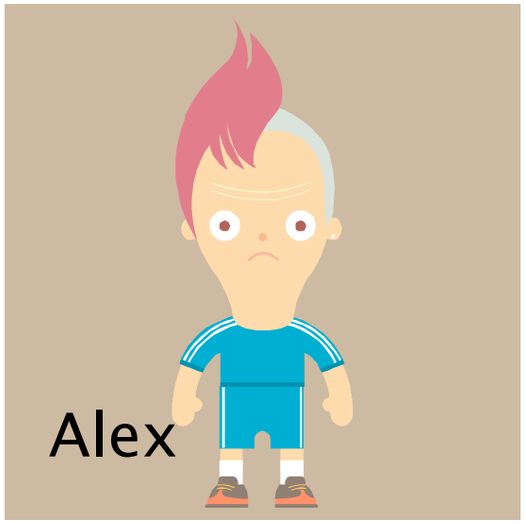
Scenario



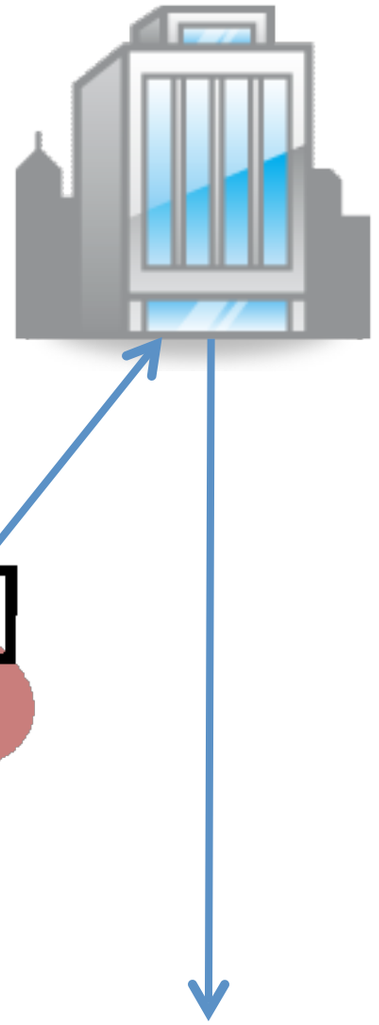


Corrupted
“contact”

Scenario



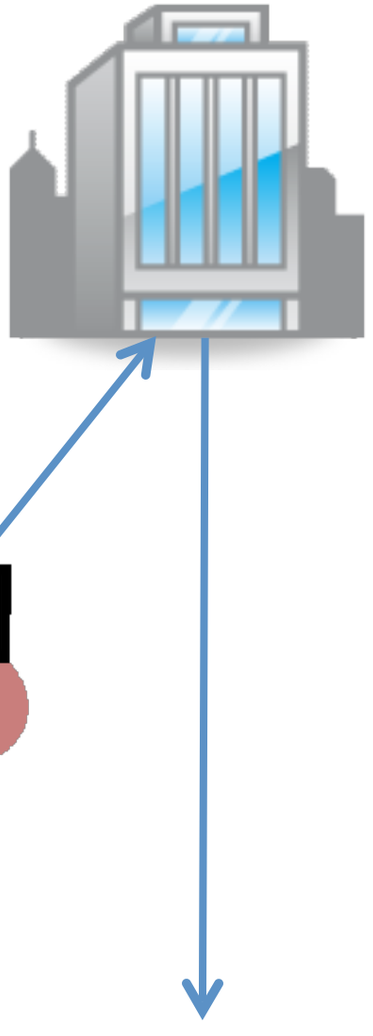
Corrupted
“contact”



Workgroup

Scenario

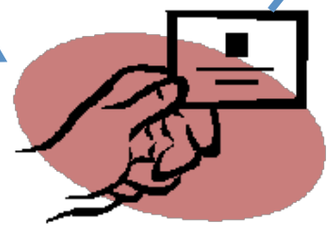
Sam



Corrupted
“contact”

Workgroup

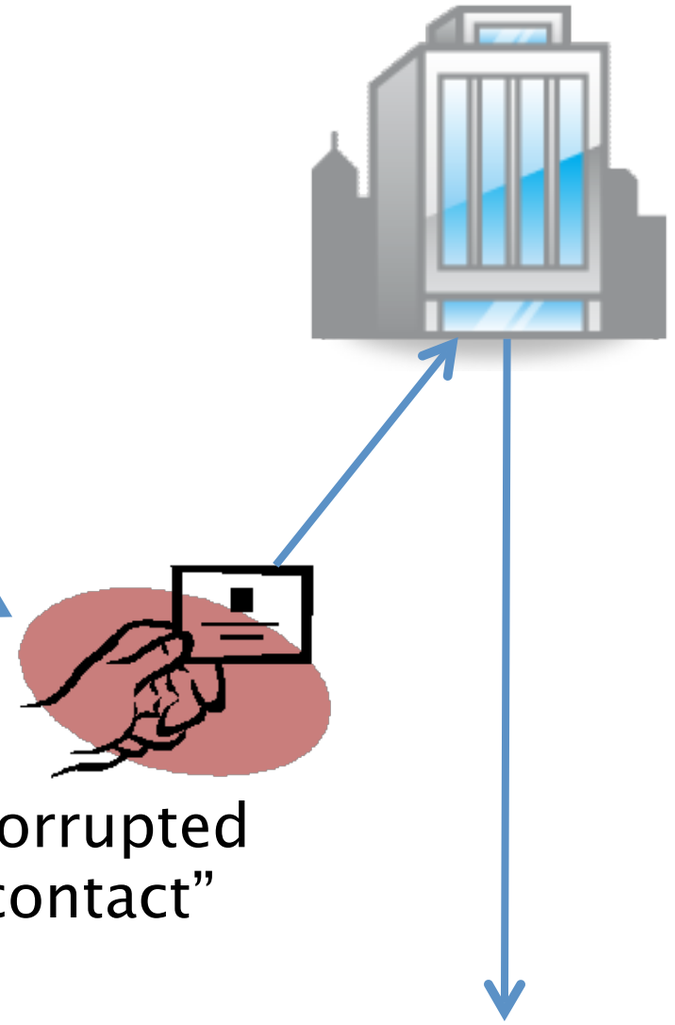
Scenario



Corrupted
“contact”

Workgroup

Scenario

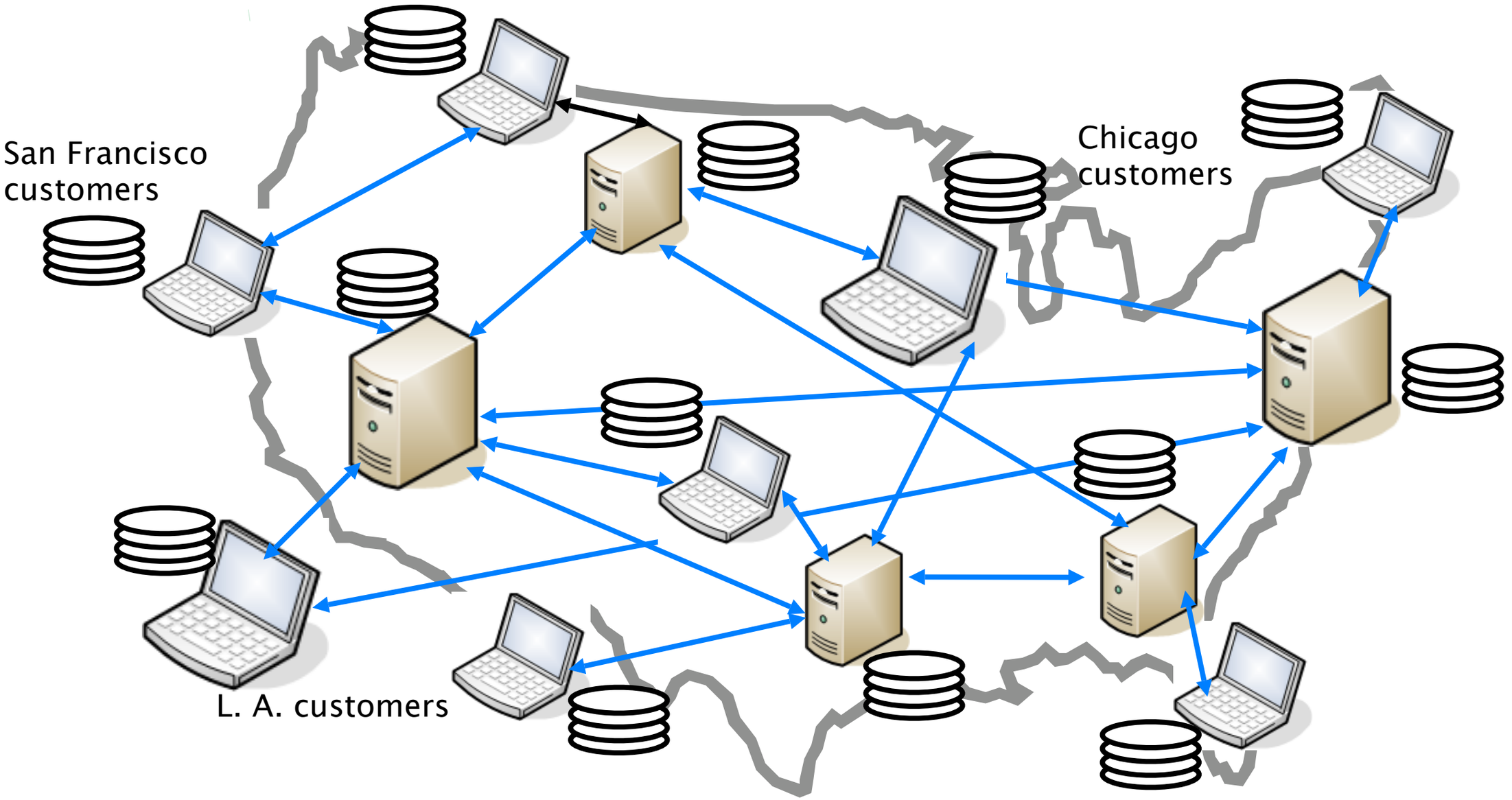


Corrupted
“contact”

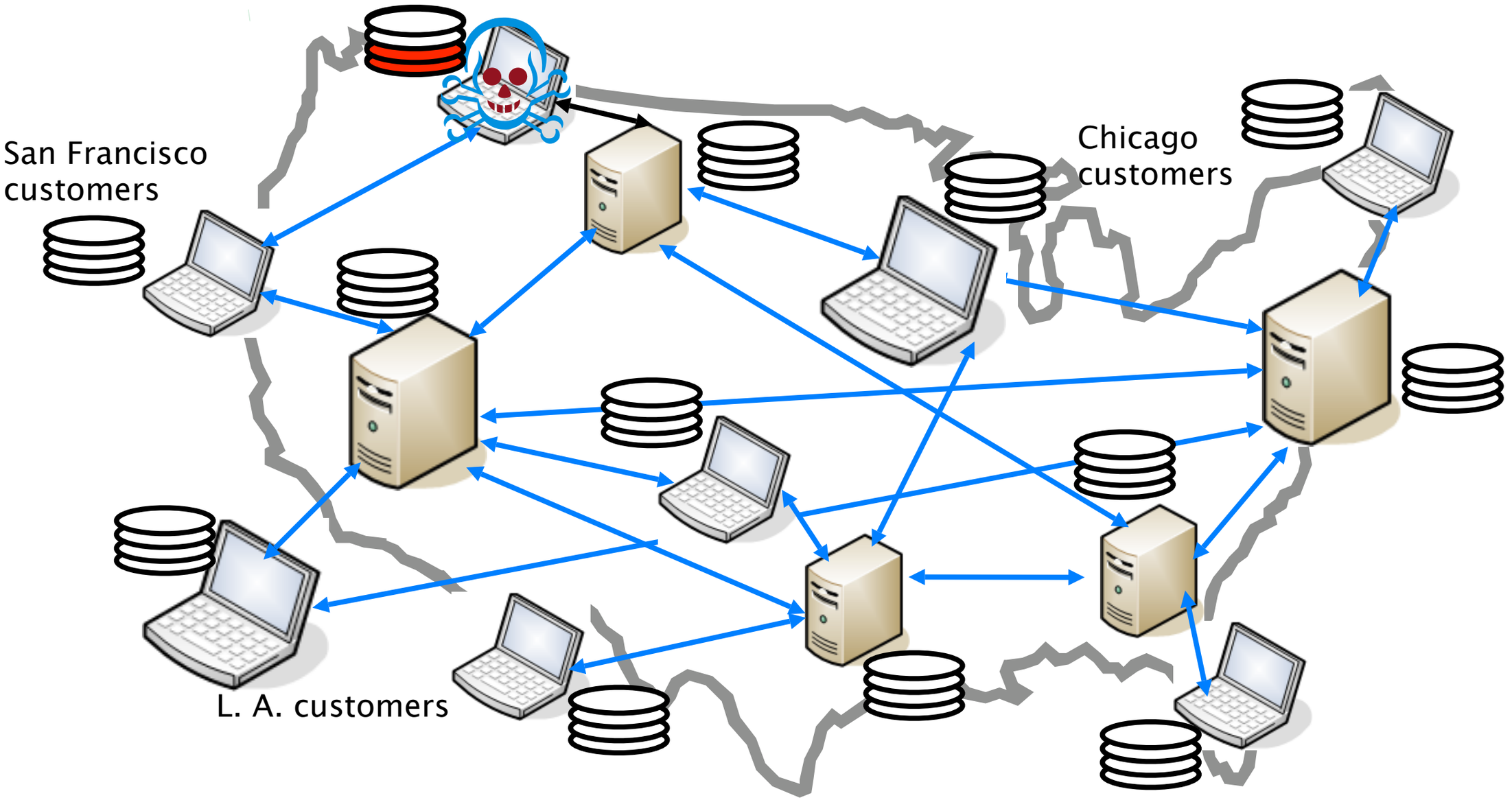
Workgroup

In replicated systems, even inappropriate updates propagate automatically.

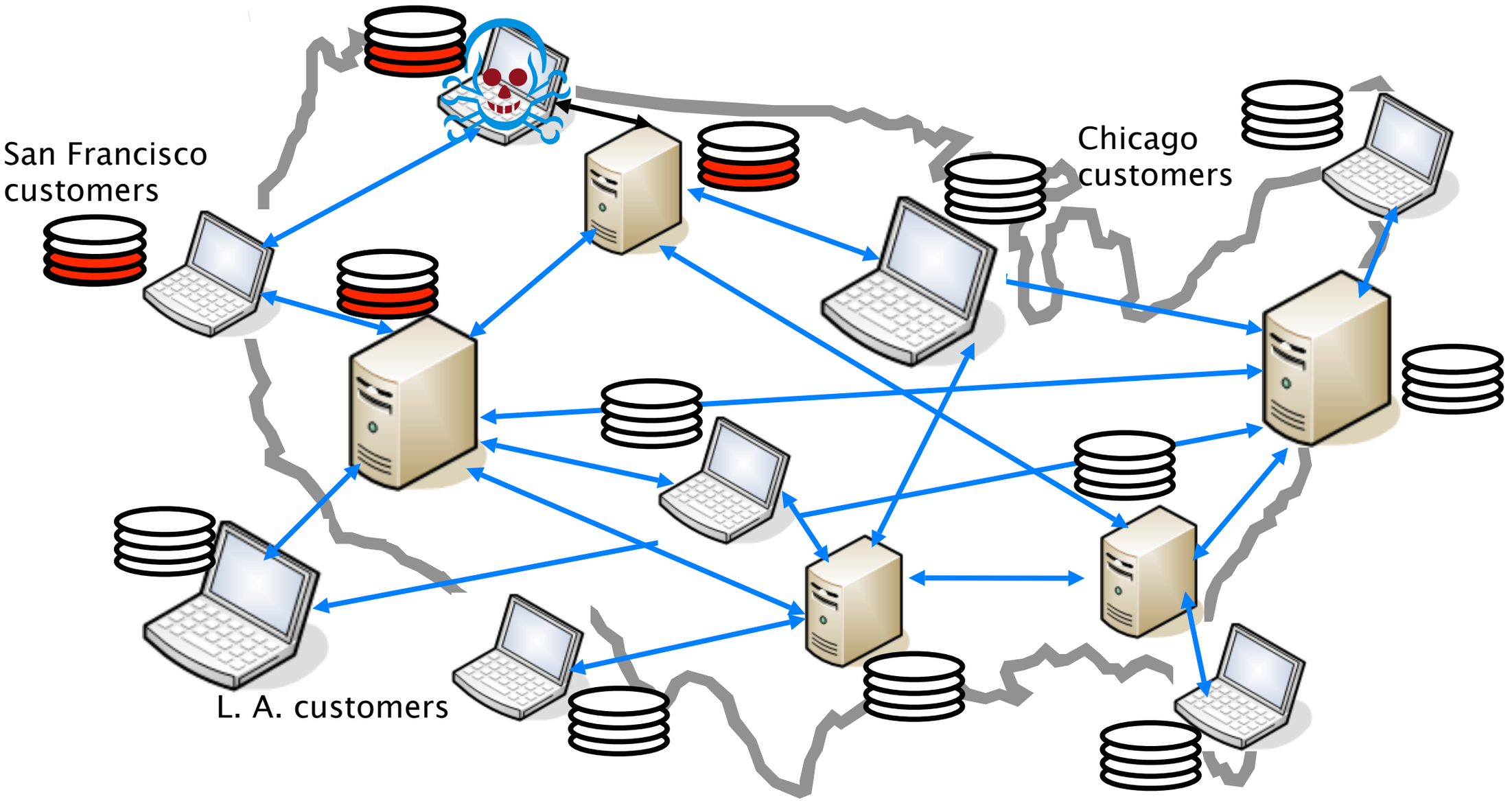
Another Scenario



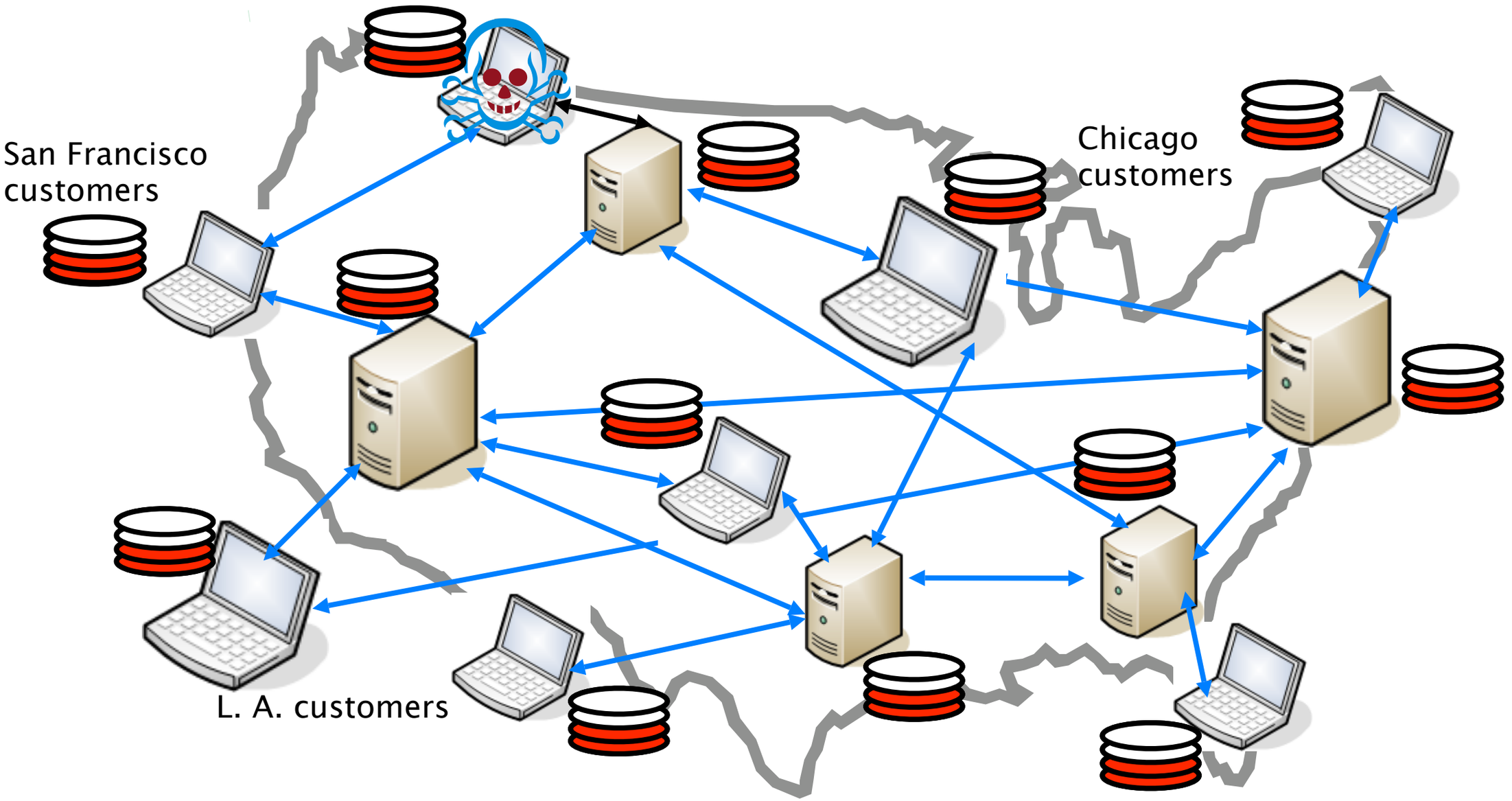
Another Scenario



Another Scenario



Another Scenario



Our Contributions

Polygraph: A framework that

- Extends weakly consistent replication
- Removes corrupted updates
- Recovers uncorrupted updates
- While being
  - Effective: Retain most uncorrupted updates
  - Efficient: Incur less bandwidth cost

Outline

- Motivation
- System Model
- Backup-based approach
- Polygraph: Effective and Efficient Recovery
- Results
- Conclusion

System Model

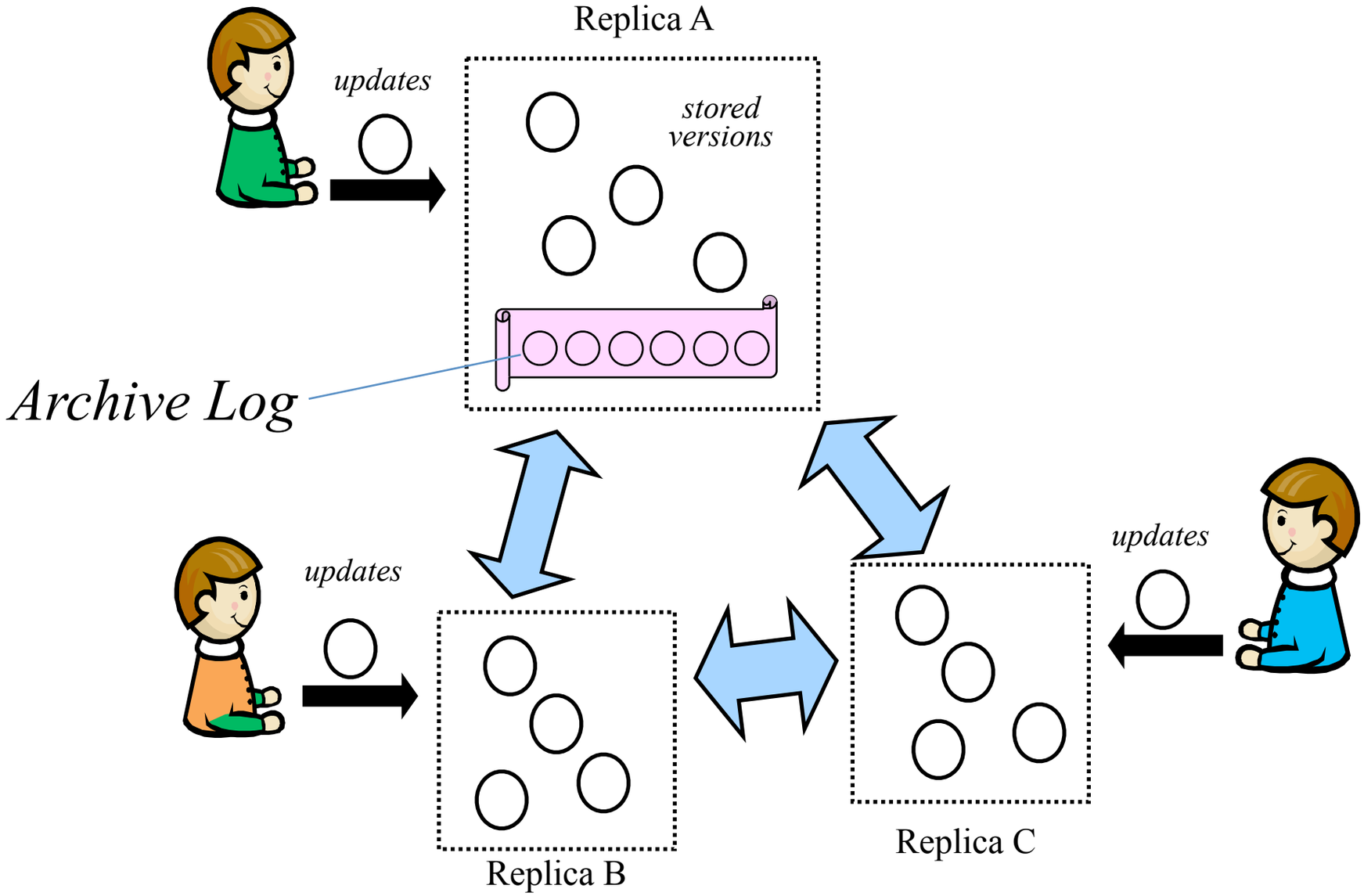
System Model

- Replicas can independently update items
- Each update produces a new version of item
- New versions propagate asynchronously

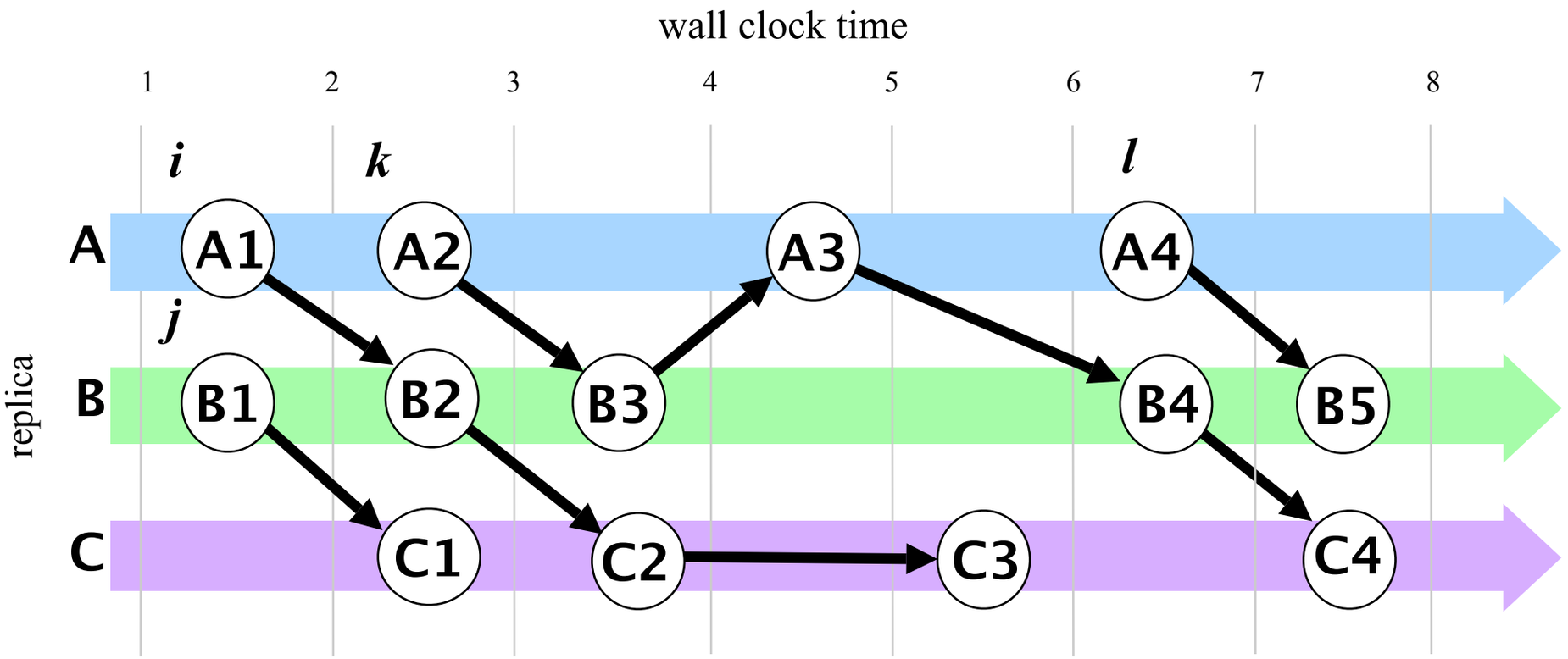
System Model

- Replicas can independently update items
- Each update produces a new version of item
- New versions propagate asynchronously
- Replicas retain the most recent version
- Archive replica logs all received versions

Example System



Update Timeline



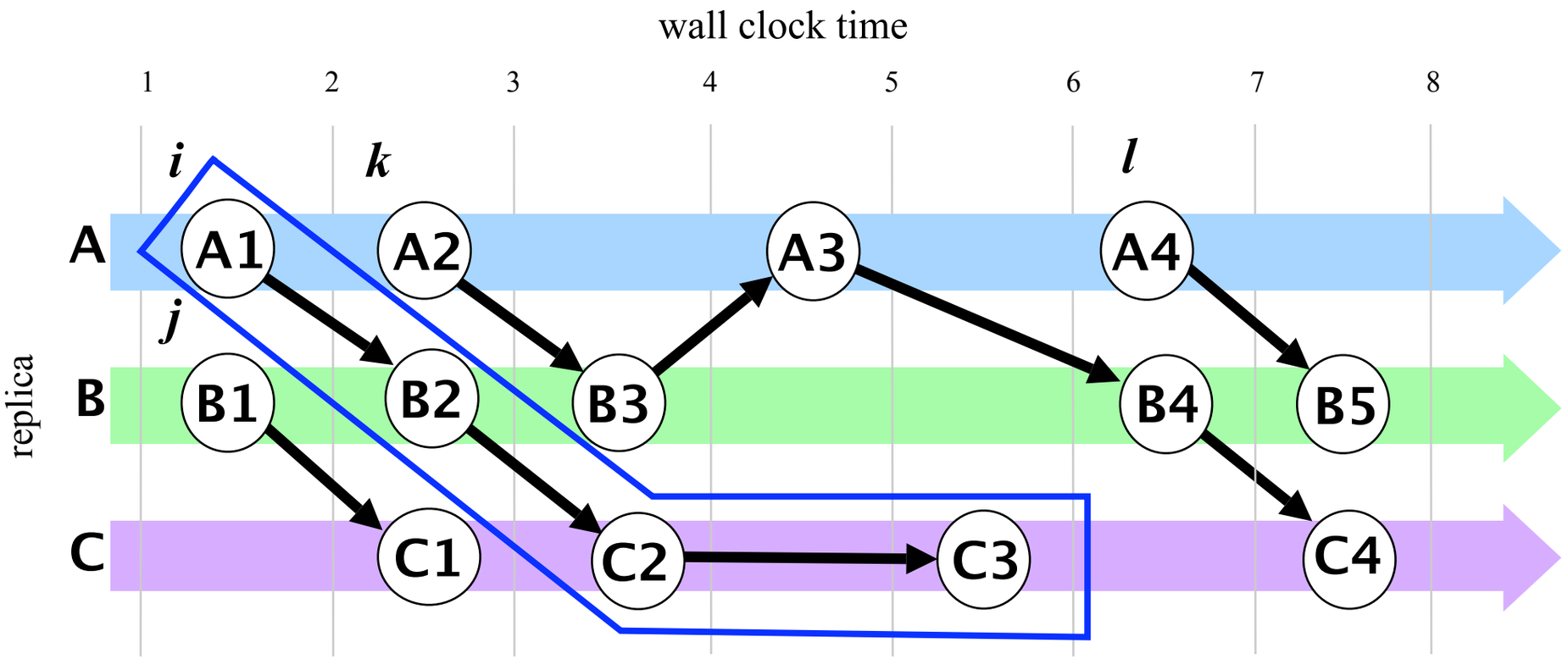
A, B, C replicas

A1, A2,...,C4 versions

i j k l items

influence

Update Timeline

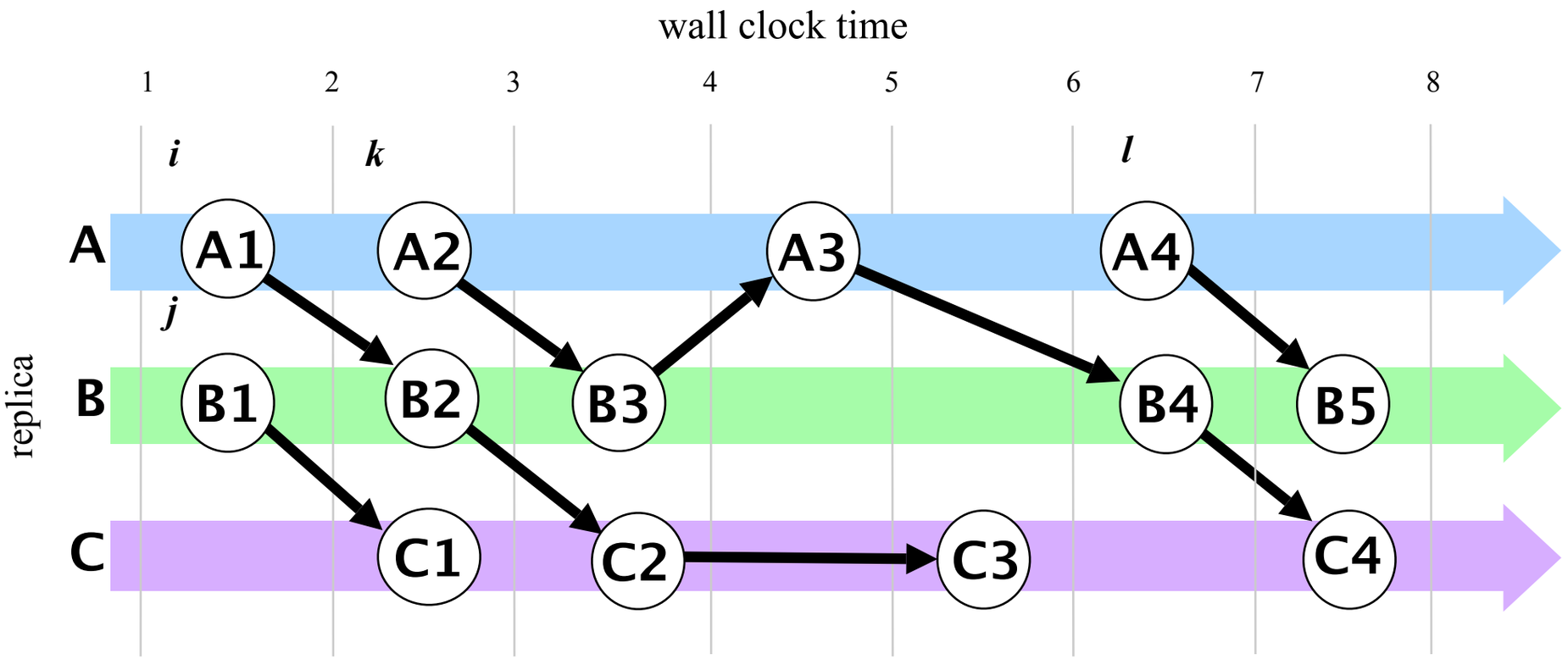


A, B, C	replicas	*i j k l*	items
A1, A2,...,C4	versions	→	influence

Threat Model

- Compromises can result from malice or misuse
- Corrupted versions
  - Versions injected by compromised replicas
  - Versions influenced by corrupted versions
- An external agent detects and reports compromises after the fact
- Archive and replication layer is not compromised

Update Timeline



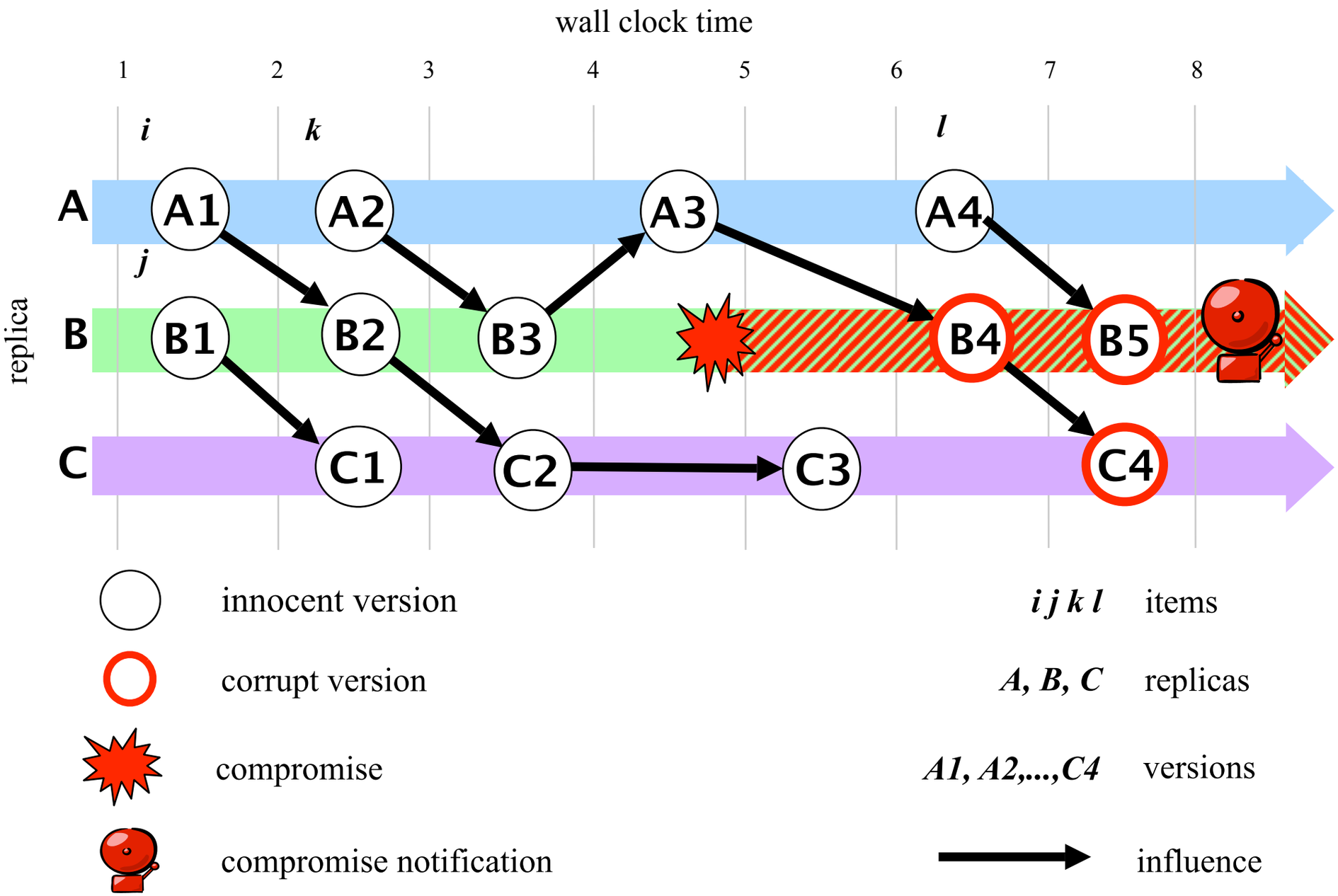
A, B, C replicas

A1, A2,...,C4 versions

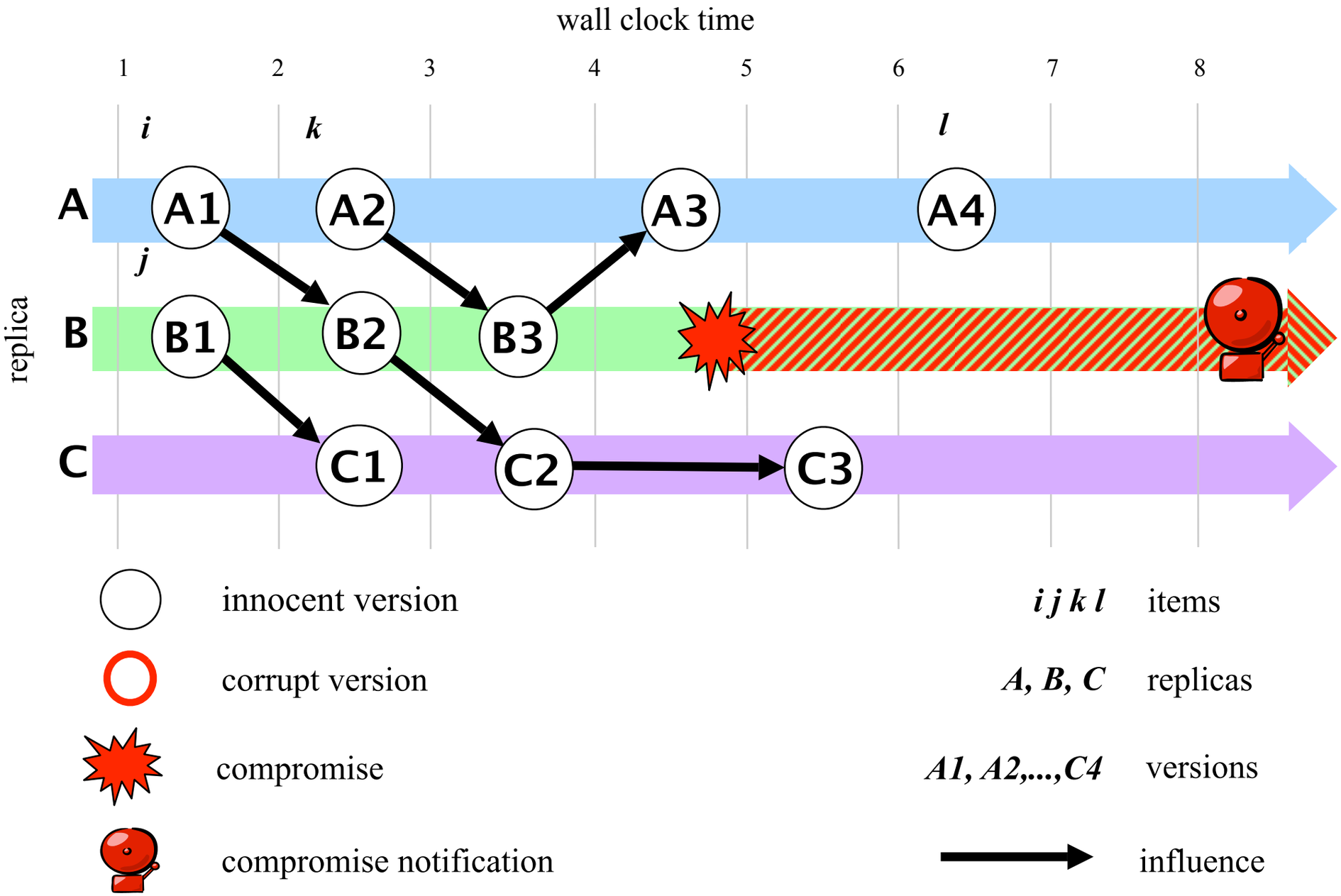
i j k l items

influence

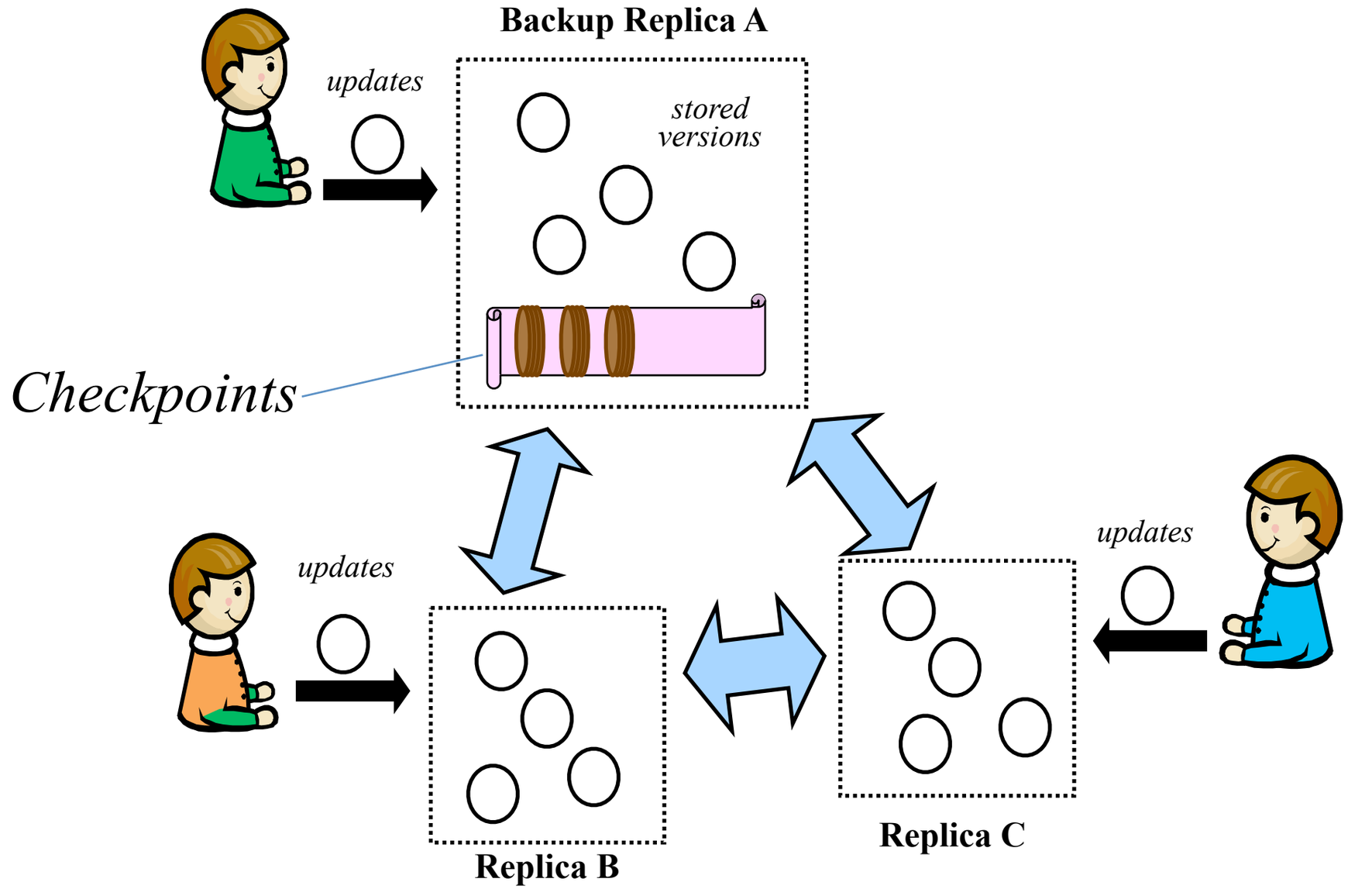
Update Timeline (with compromise)



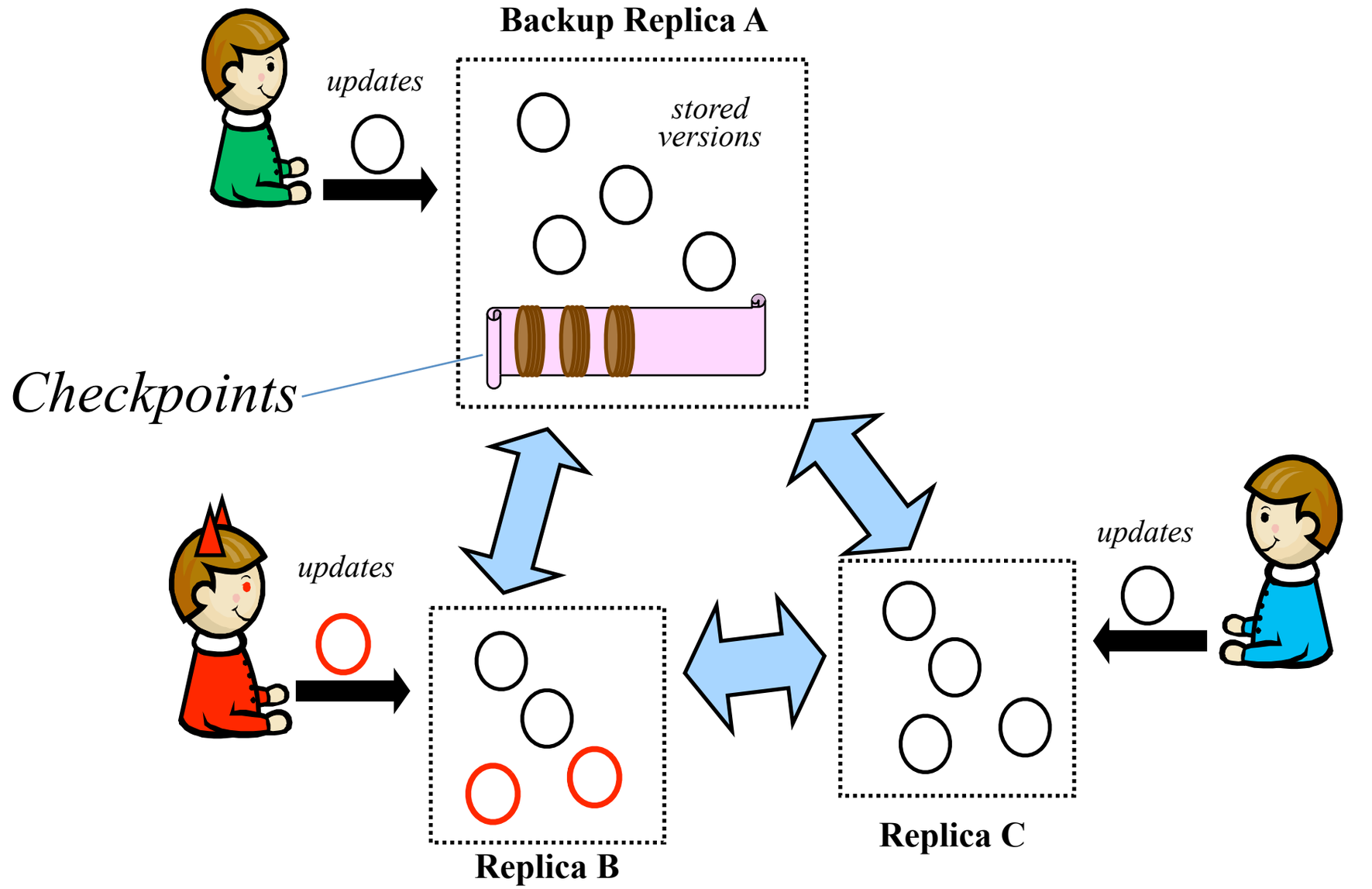
Update Timeline (after recovery)



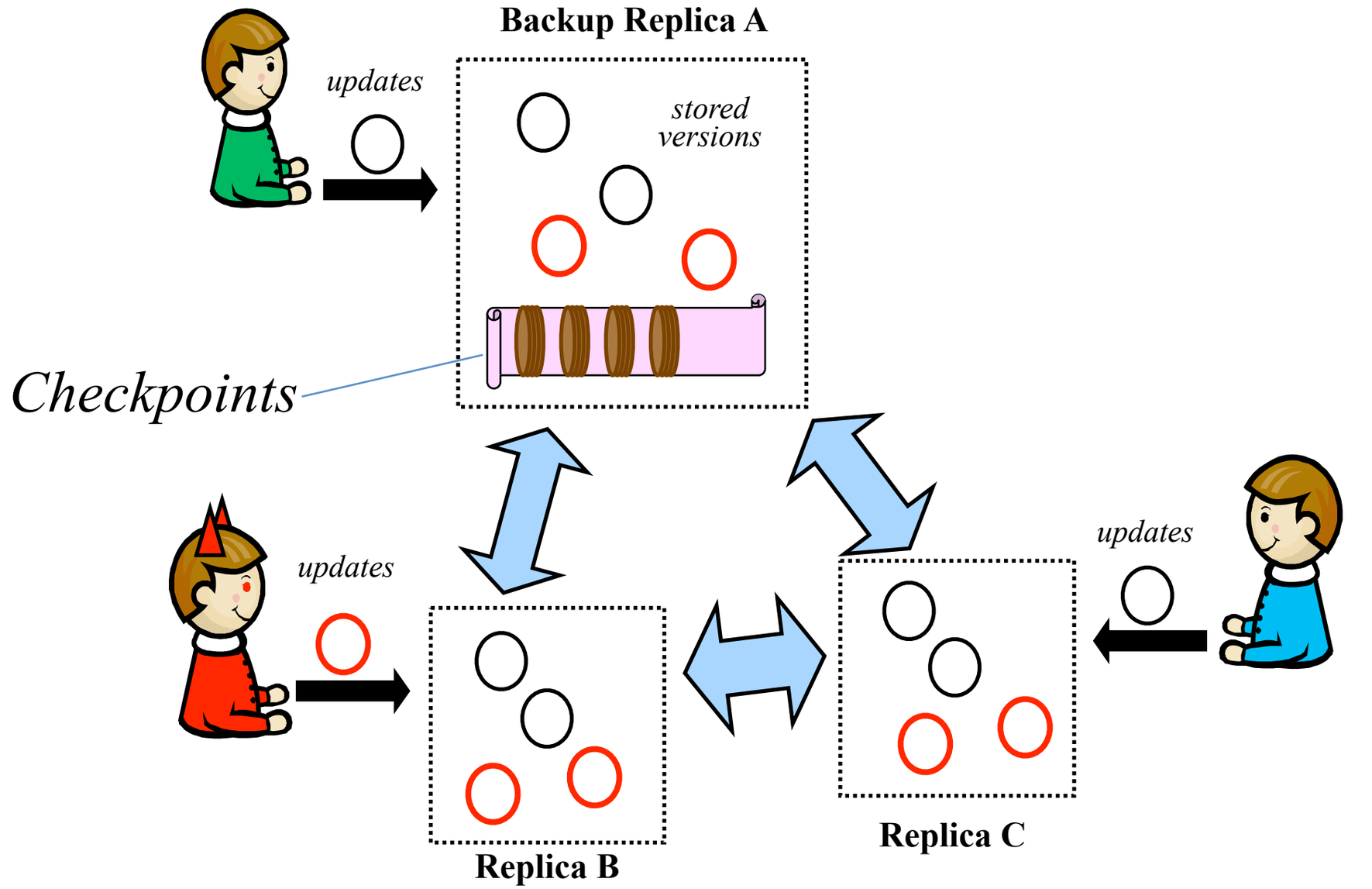
Backup-based Approach



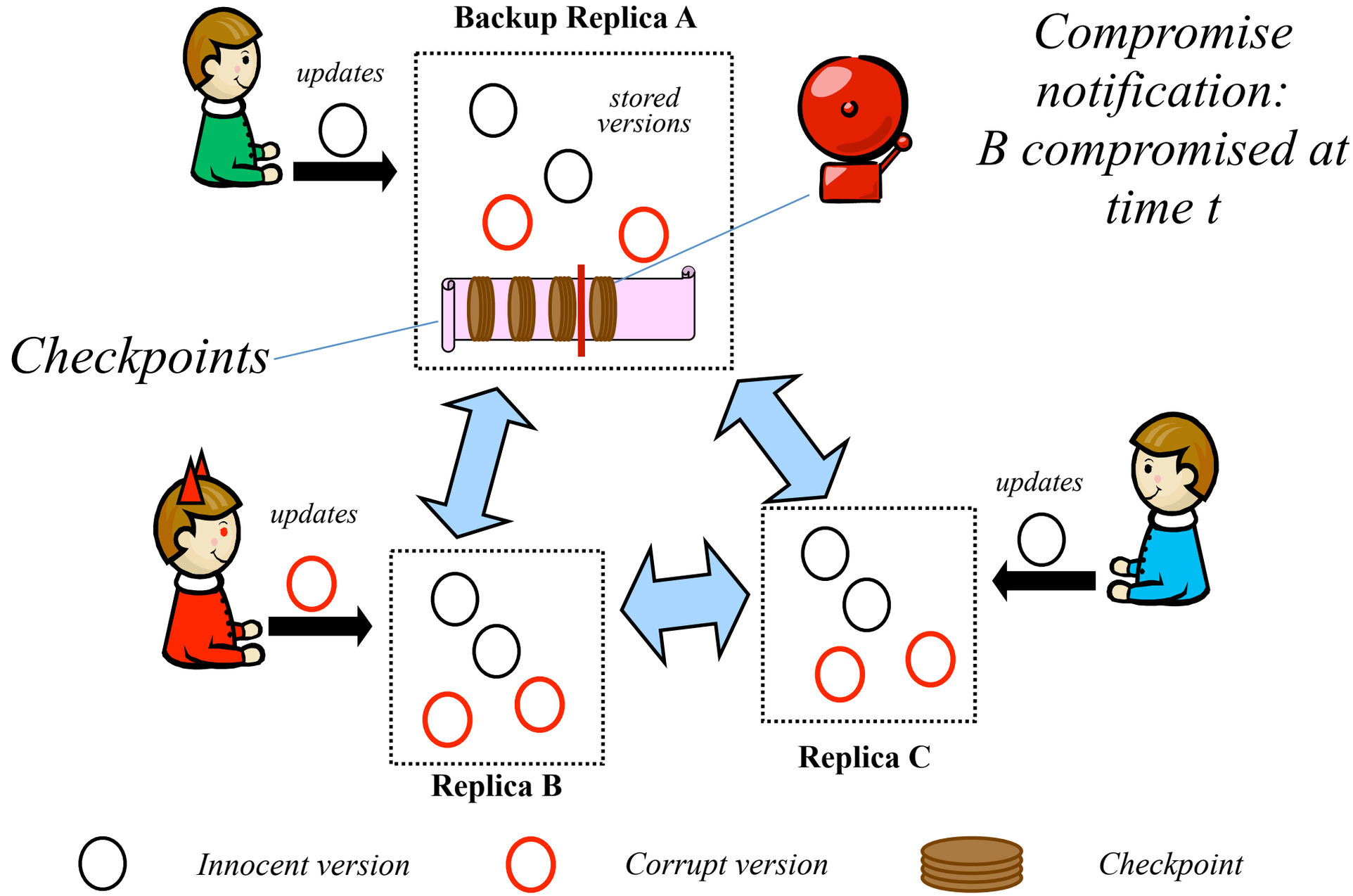
Backup-based Approach



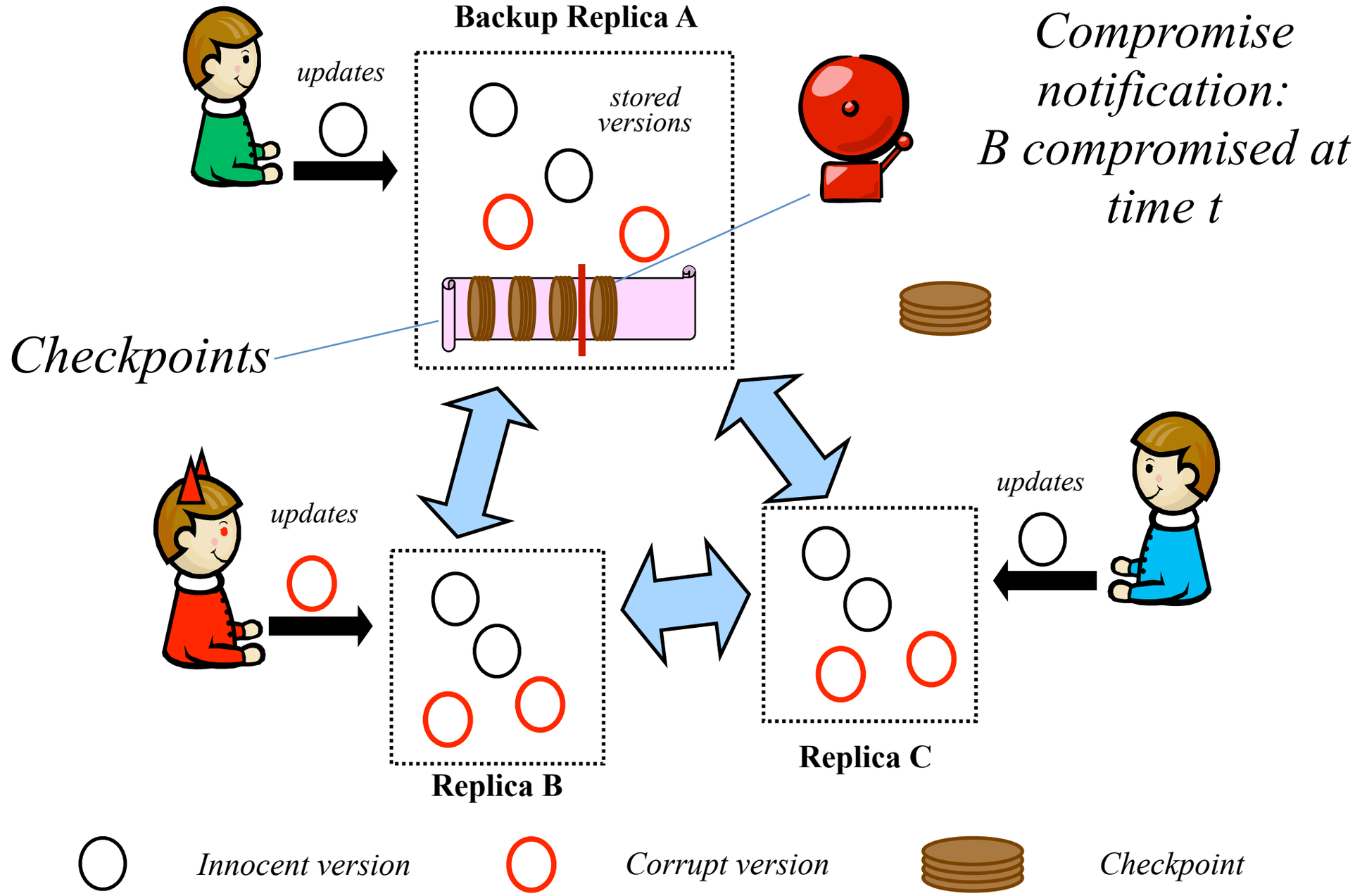
Backup-based Approach



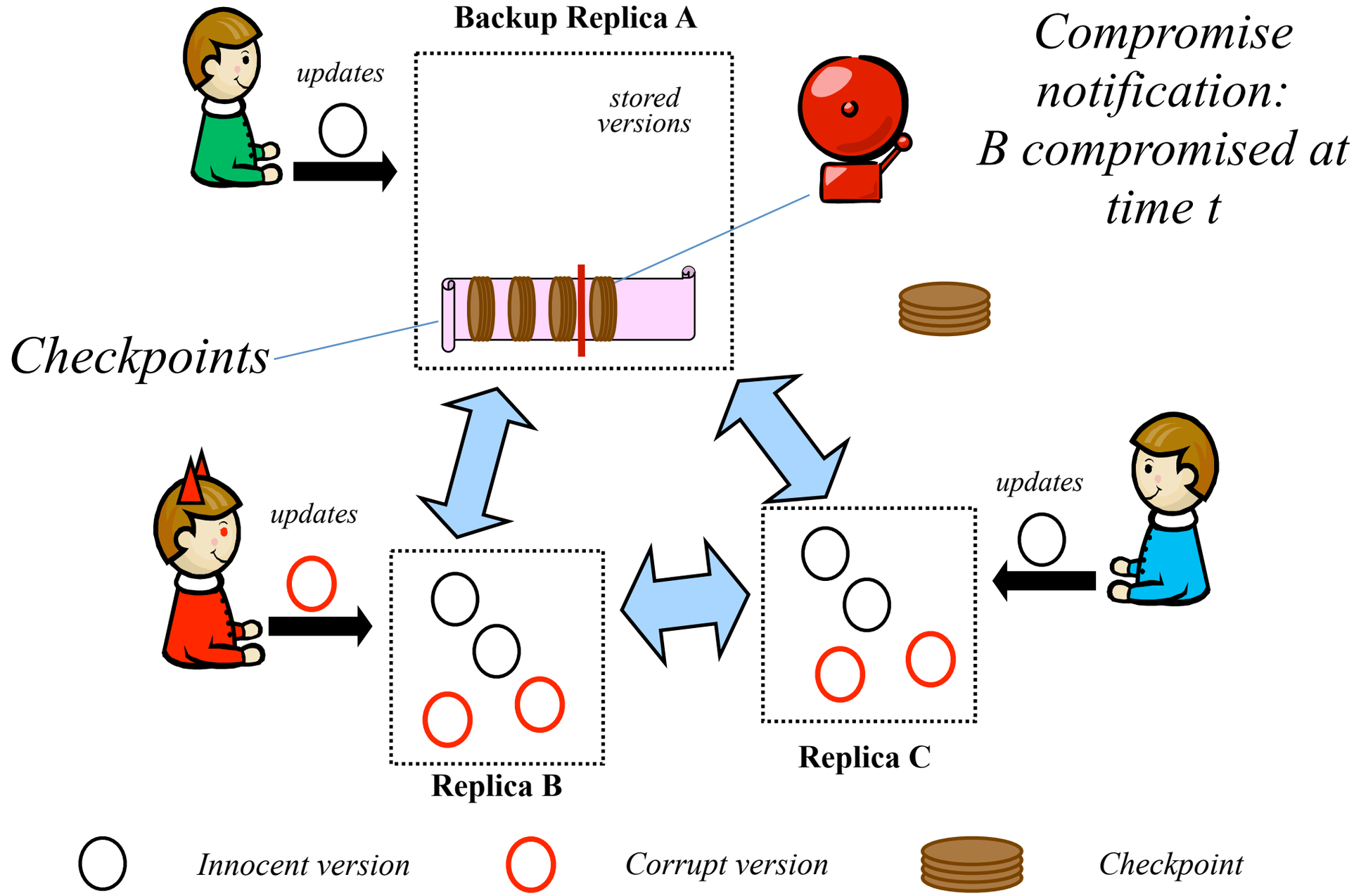
Backup-based Approach



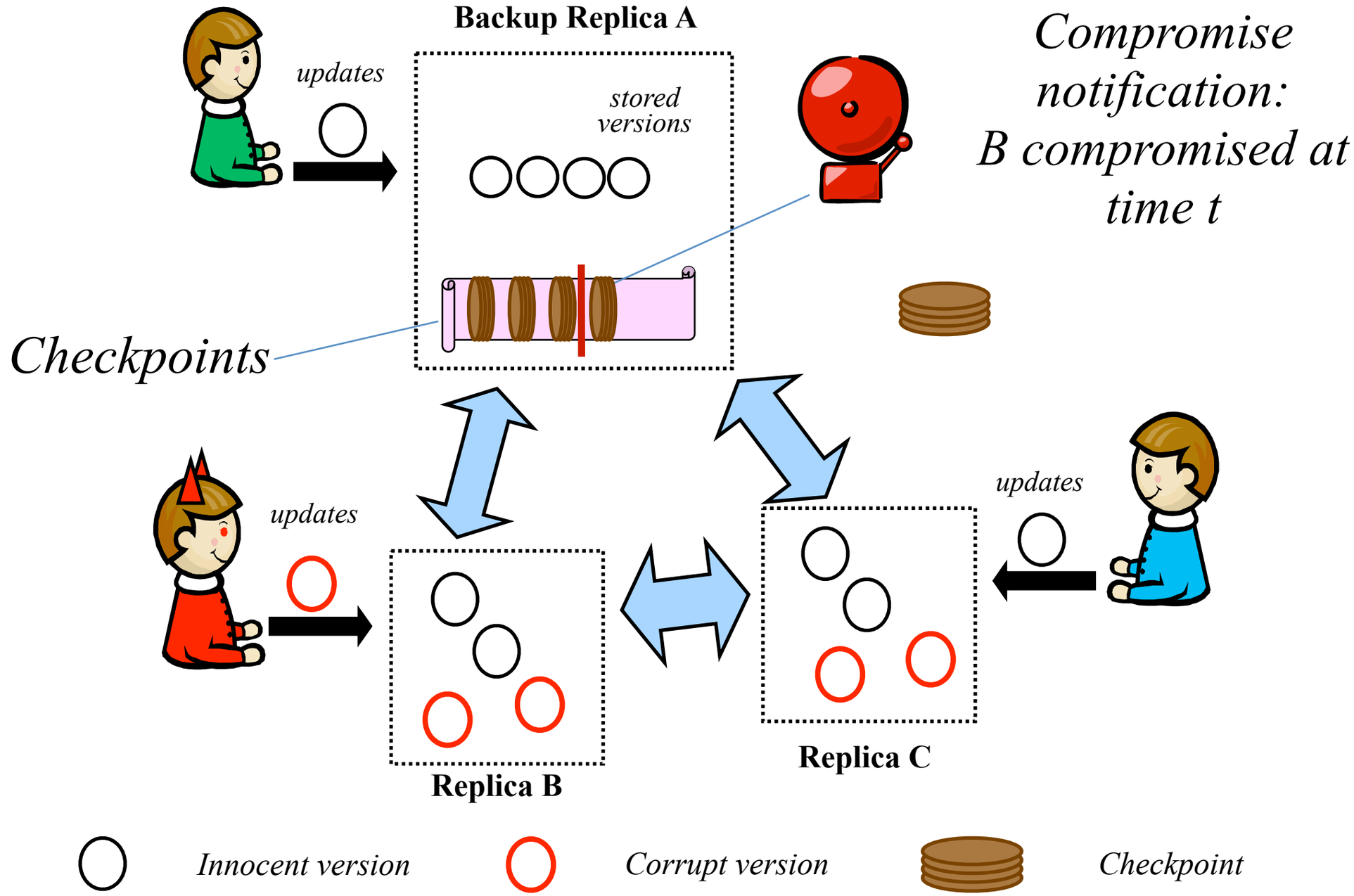
Backup-based Approach



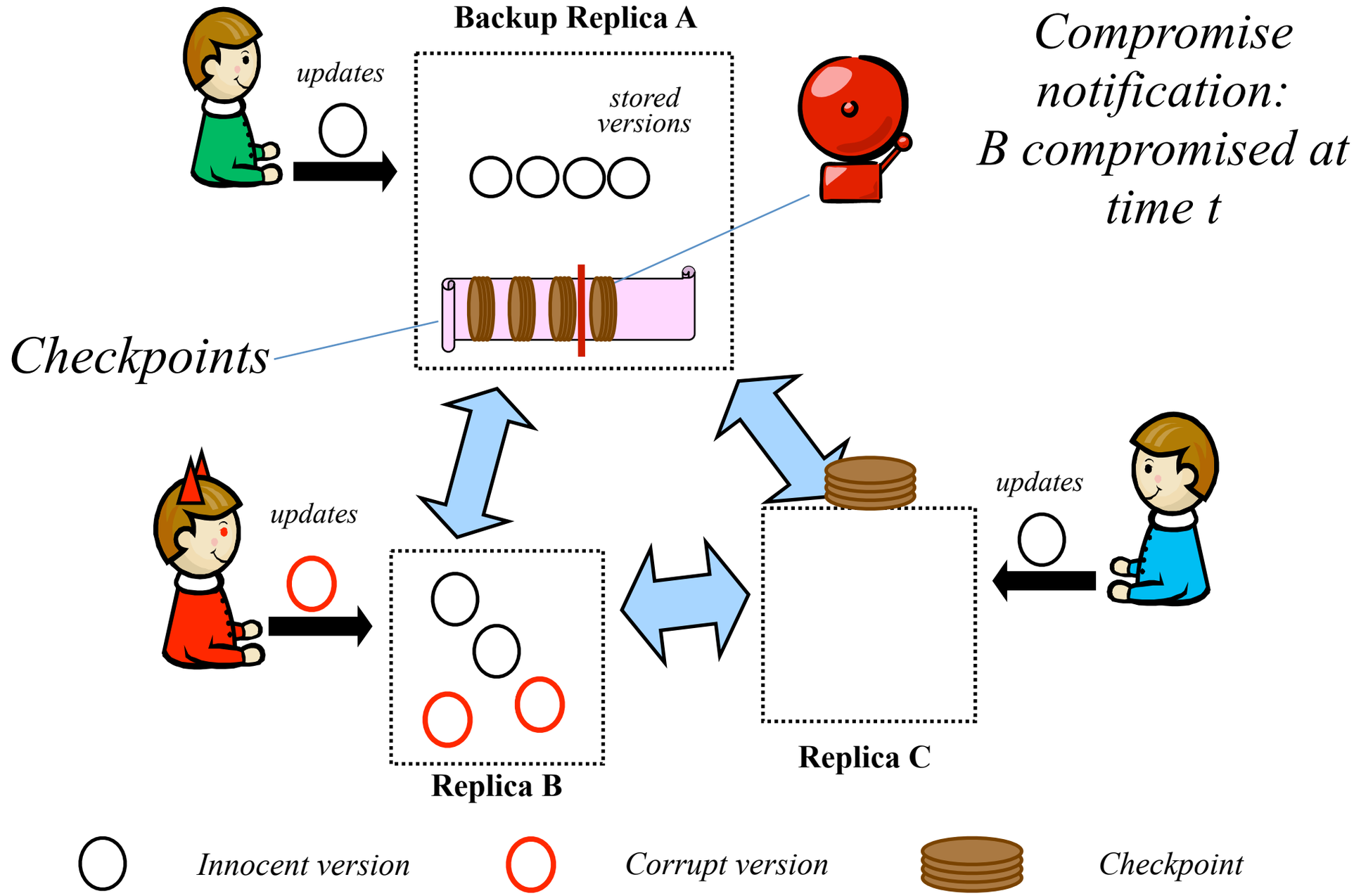
Backup-based Approach



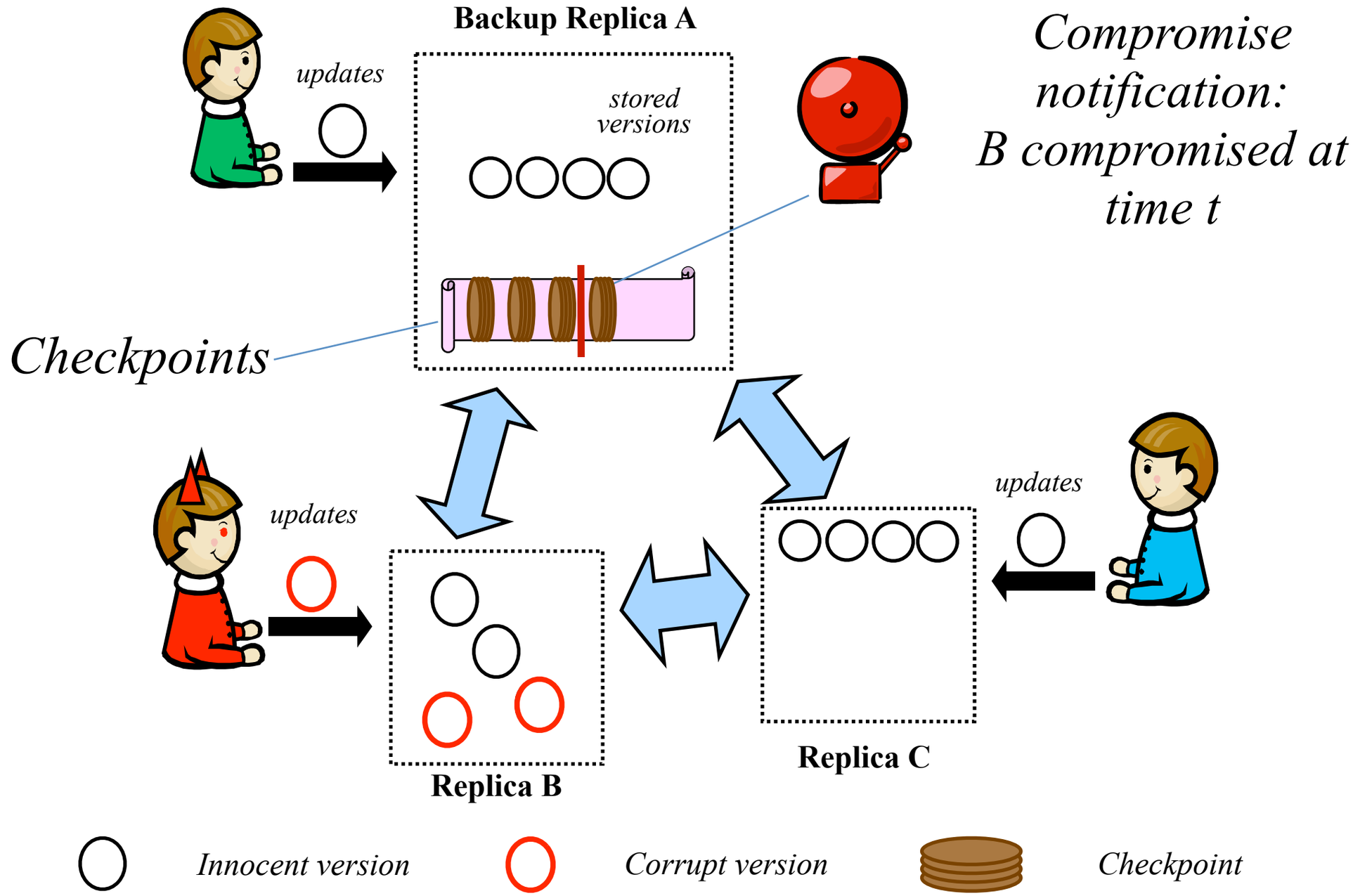
Backup-based Approach



Backup-based Approach

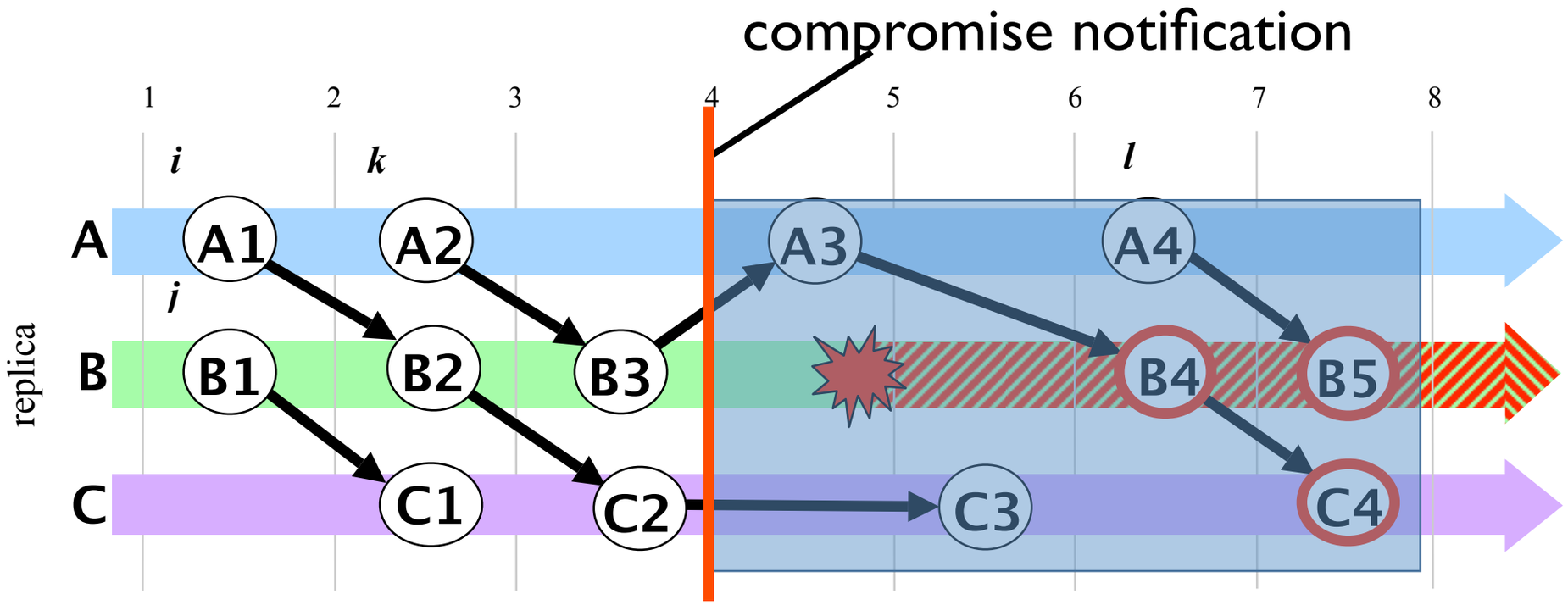


Backup-based Approach



Drawbacks of Backup-based Approach

- Inefficient: Re-propagation from backup to replicas
- Ineffective: Updates subsequent to checkpoint are lost



Polygraph: Key Ideas

Polygraph: Key Ideas

- Innocent version identification
  - Effectiveness: innocent versions created post-compromise are recovered

Polygraph: Key Ideas

- Innocent version identification
  - Effectiveness: innocent versions created post-compromise are recovered
- Replica-local retention

Polygraph: Key Ideas

- Innocent version identification
  - Effectiveness: innocent versions created post-compromise are recovered
- Replica-local retention
  - Replicas retain innocent versions

Polygraph: Key Ideas

- Innocent version identification
  - Effectiveness: innocent versions created post-compromise are recovered
- Replica-local retention
  - Replicas retain innocent versions
  - Effectiveness: newer versions recovered

Polygraph: Key Ideas

- Innocent version identification
  - Effectiveness: innocent versions created post-compromise are recovered
- Replica-local retention
  - Replicas retain innocent versions
  - Effectiveness: newer versions recovered
  - Efficiency: retained versions save bandwidth

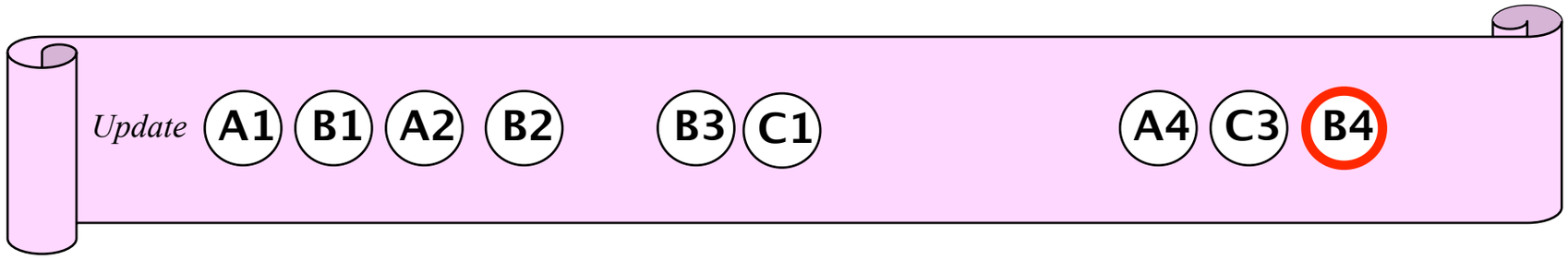
Innocent Versions

Version is innocent if it is

- Generated before compromise, or
- Not influenced by any corrupt version from the compromised replica

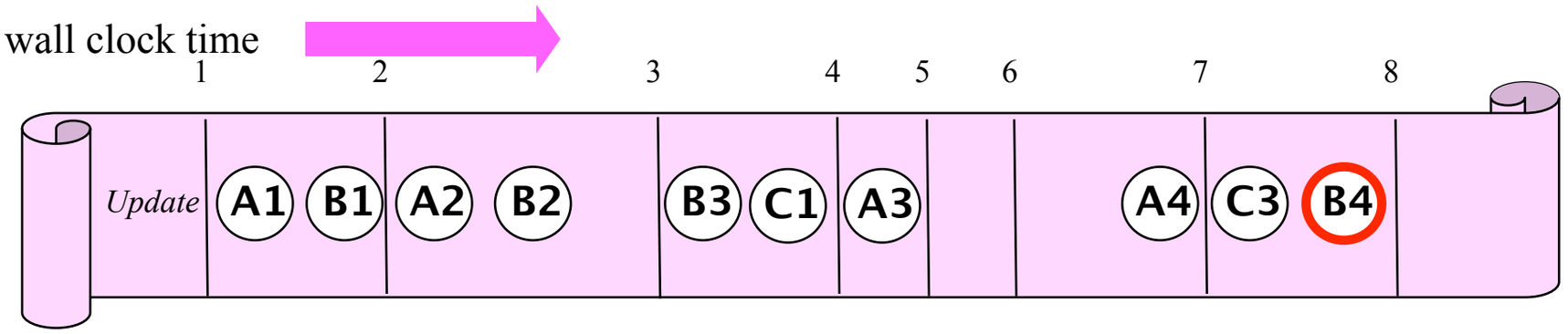
Is a version generated before compromise?

Archive Log

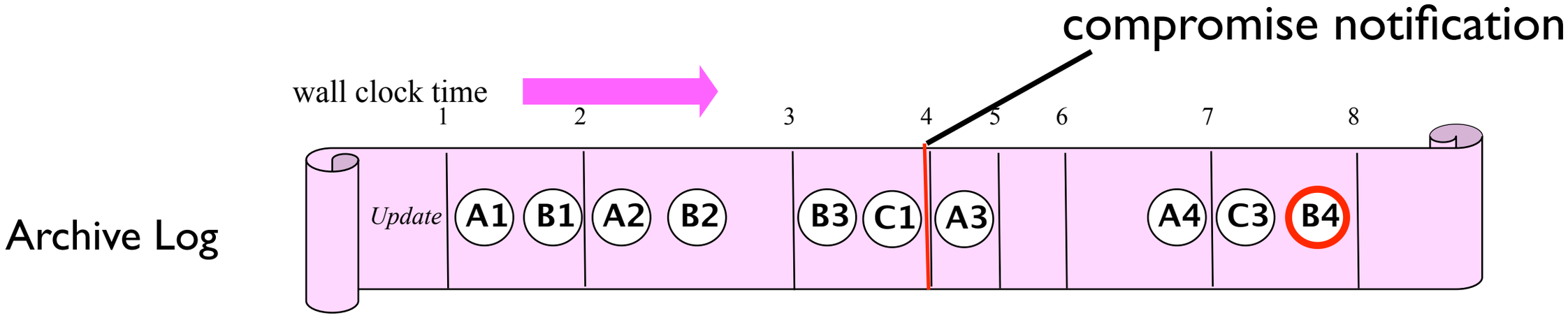


Is a version generated before compromise?

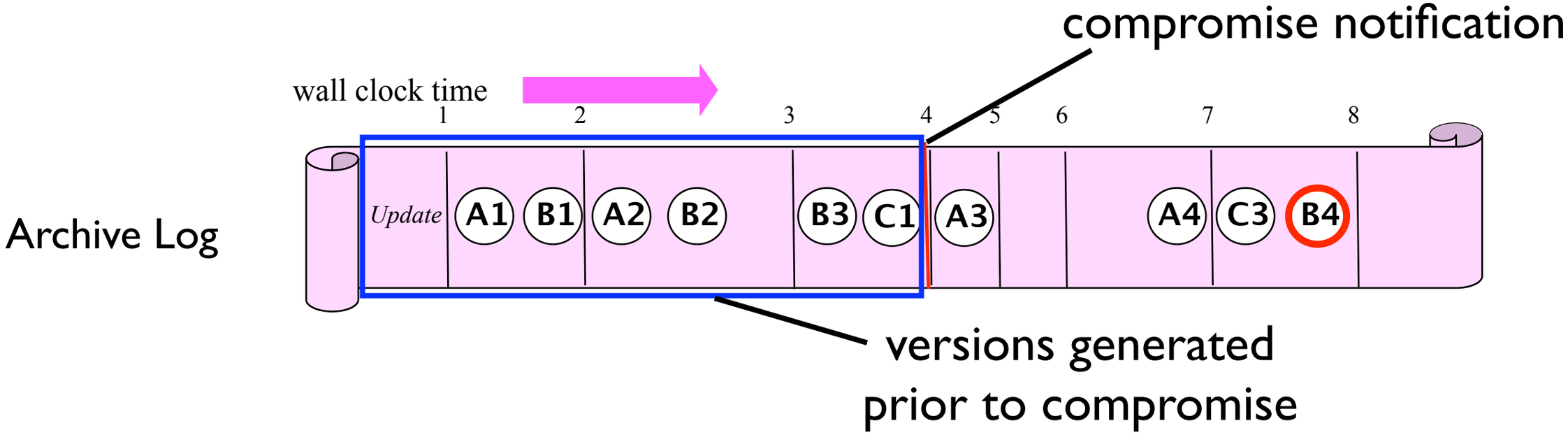
Archive Log



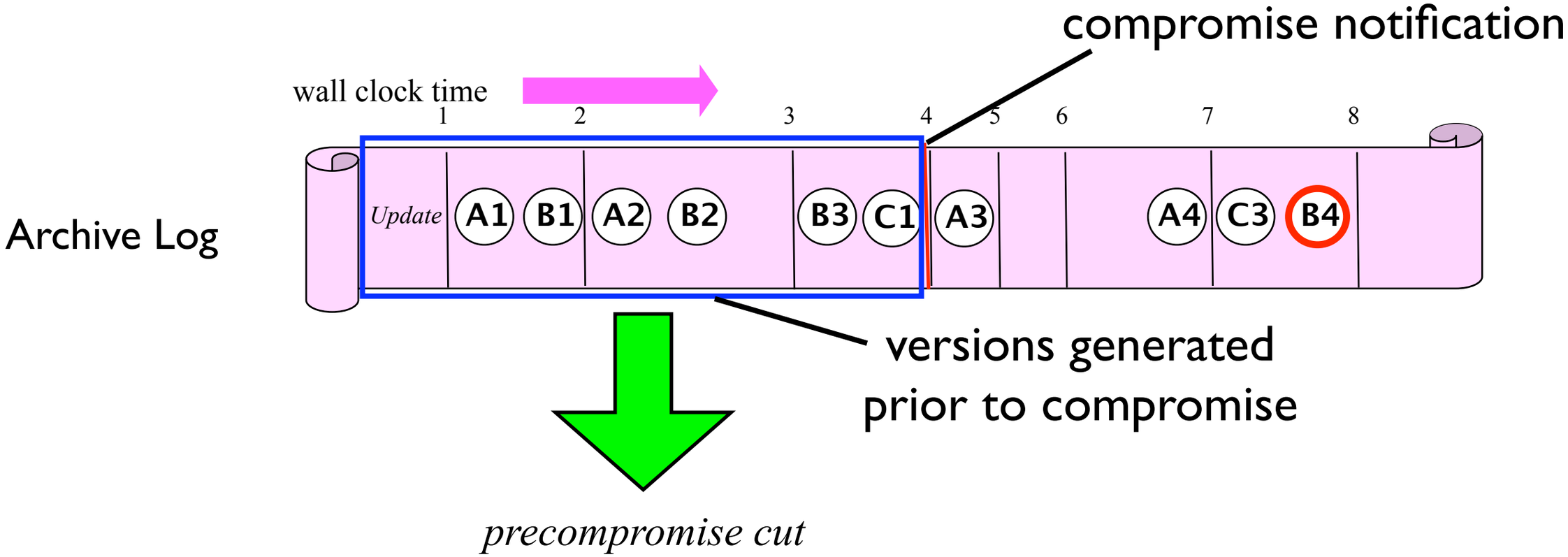
Is a version generated before compromise?



Is a version generated before compromise?



Is a version generated before compromise?



Precompromise cut summarizes versions archived prior to compromise

Is a version ***v*** influenced by any corrupt version from the compromised replica?

Is a version ***v*** influenced by any corrupt version from the compromised replica?

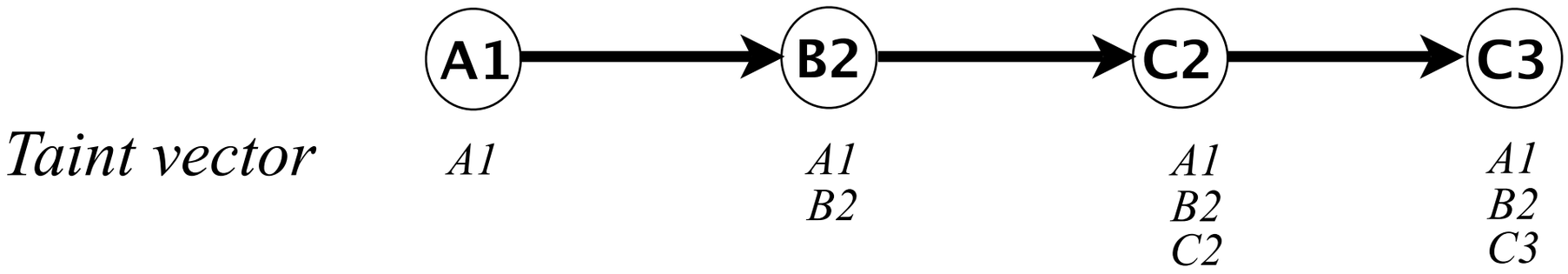
- Sufficient to check: is the most recent version from the compromised replica that influenced ***v*** is corrupt?

Is a version ***v*** influenced by any corrupt version from the compromised replica?

- Sufficient to check: is the most recent version from the compromised replica that influenced ***v*** is corrupt?
- Each version has a *taint vector*
- Taint vector of a version ***v*** tracks the most recent version from each replica that has influenced ***v***

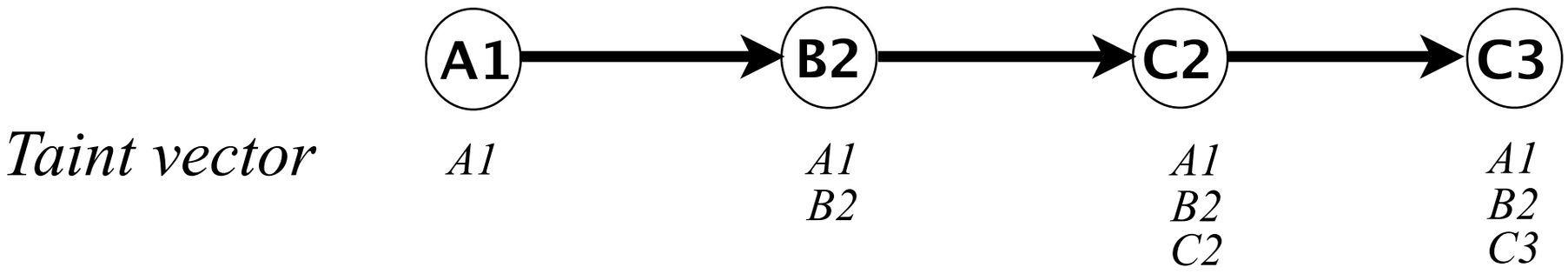
Is a version ***v*** influenced by any corrupt version from the compromised replica?

- Sufficient to check: is the most recent version from the compromised replica that influenced ***v*** is corrupt?
- Each version has a *taint vector*
- Taint vector of a version ***v*** tracks the most recent version from each replica that has influenced ***v***



Is a version ***v*** influenced by any corrupt version from the compromised replica?

- Sufficient to check: is the most recent version from the compromised replica that influenced ***v*** is corrupt?
- Each version has a *taint vector*
- Taint vector of a version ***v*** tracks the most recent version from each replica that has influenced ***v***



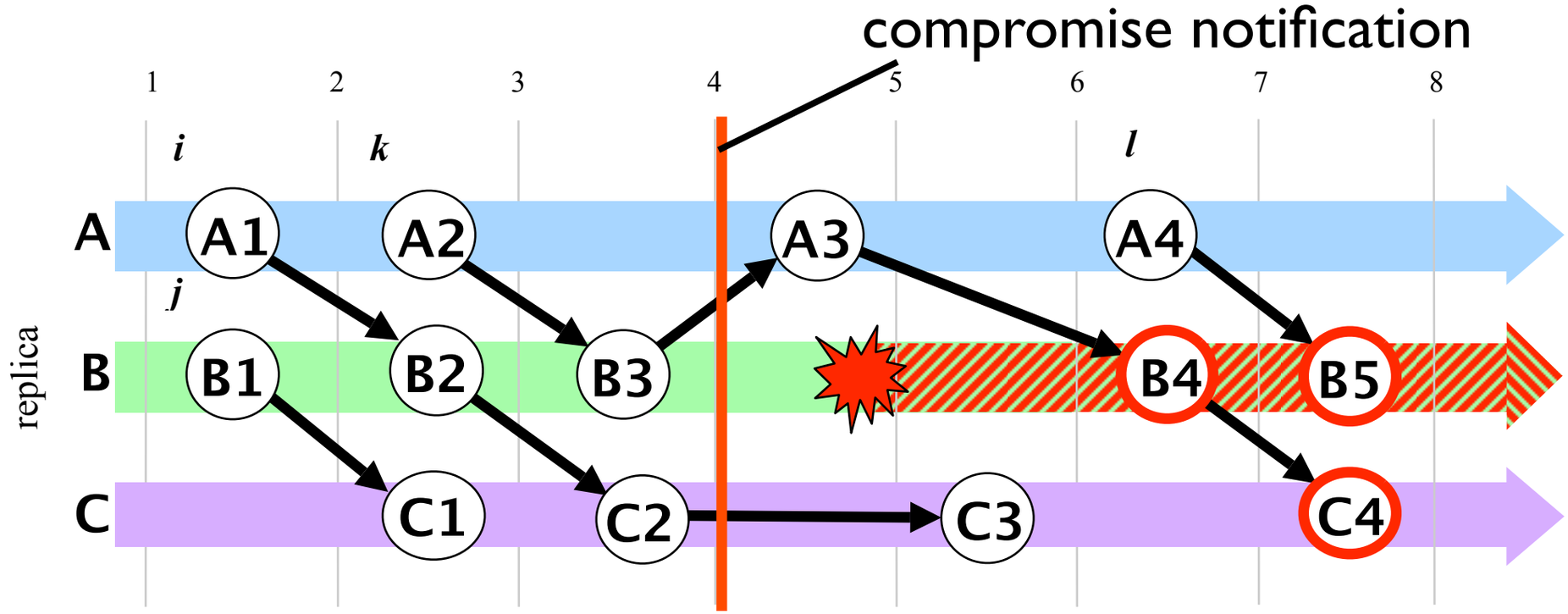
A version ***v*** is innocent if the influencing version from the compromised replica in ***v***'s taint vector is innocent

Innocent Version Identification

Can identify innocent versions received after compromise

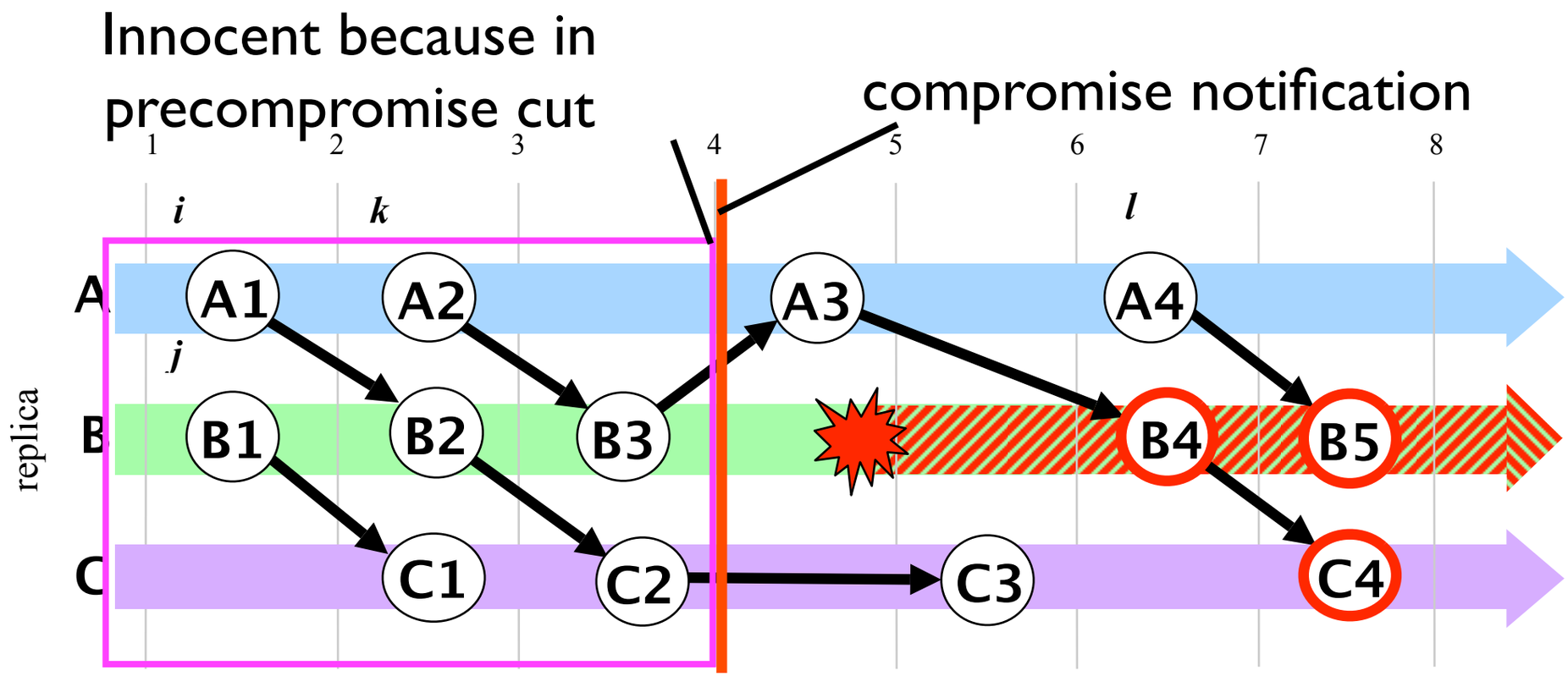
Innocent Version Identification

Can identify innocent versions received after compromise



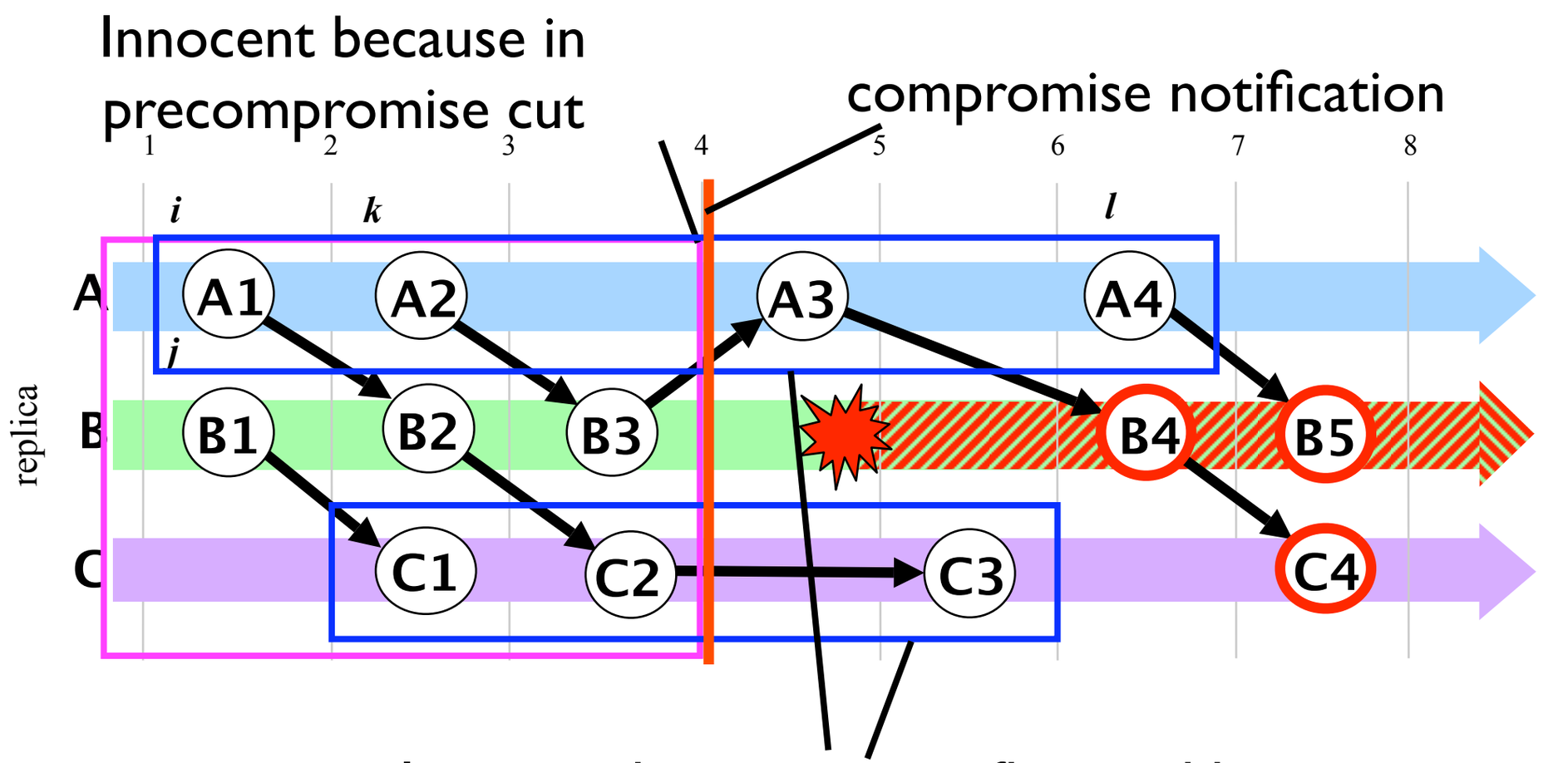
Innocent Version Identification

Can identify innocent versions received after compromise



Innocent Version Identification

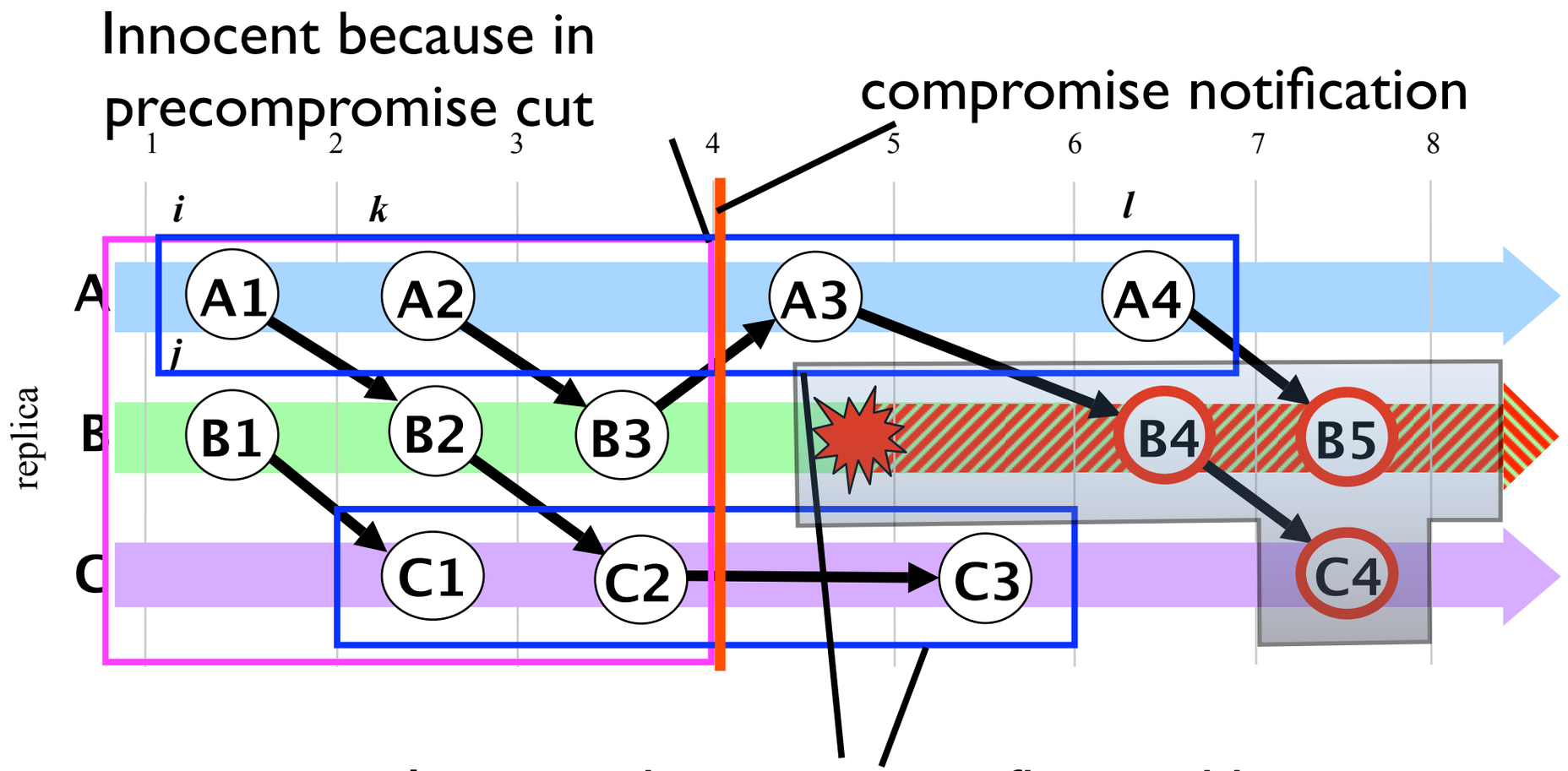
Can identify innocent versions received after compromise



Innocent because not influenced by any corrupt version from the compromised replica

Innocent Version Identification

Can identify innocent versions received after compromise

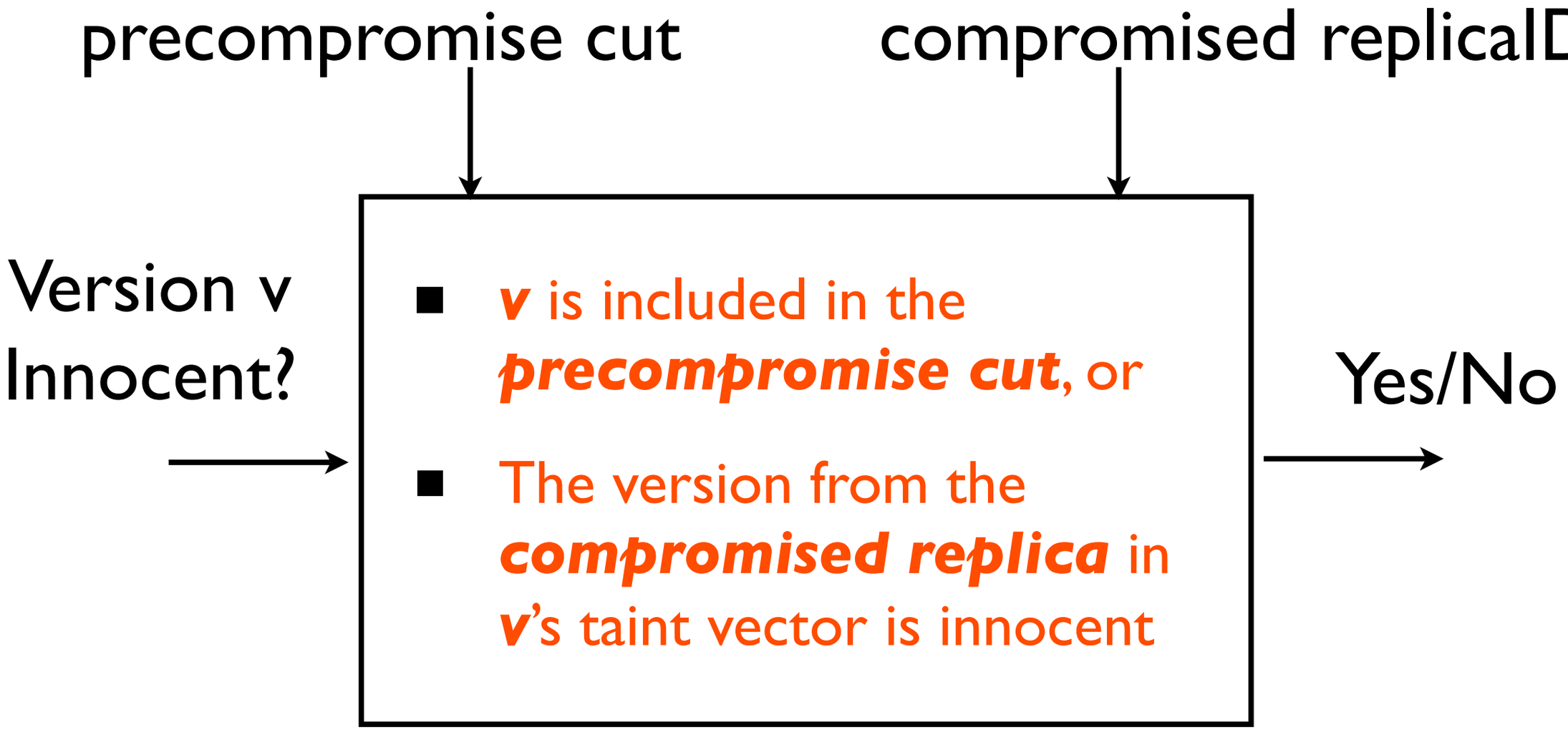


Innocent because not influenced by any corrupt version from the compromised replica

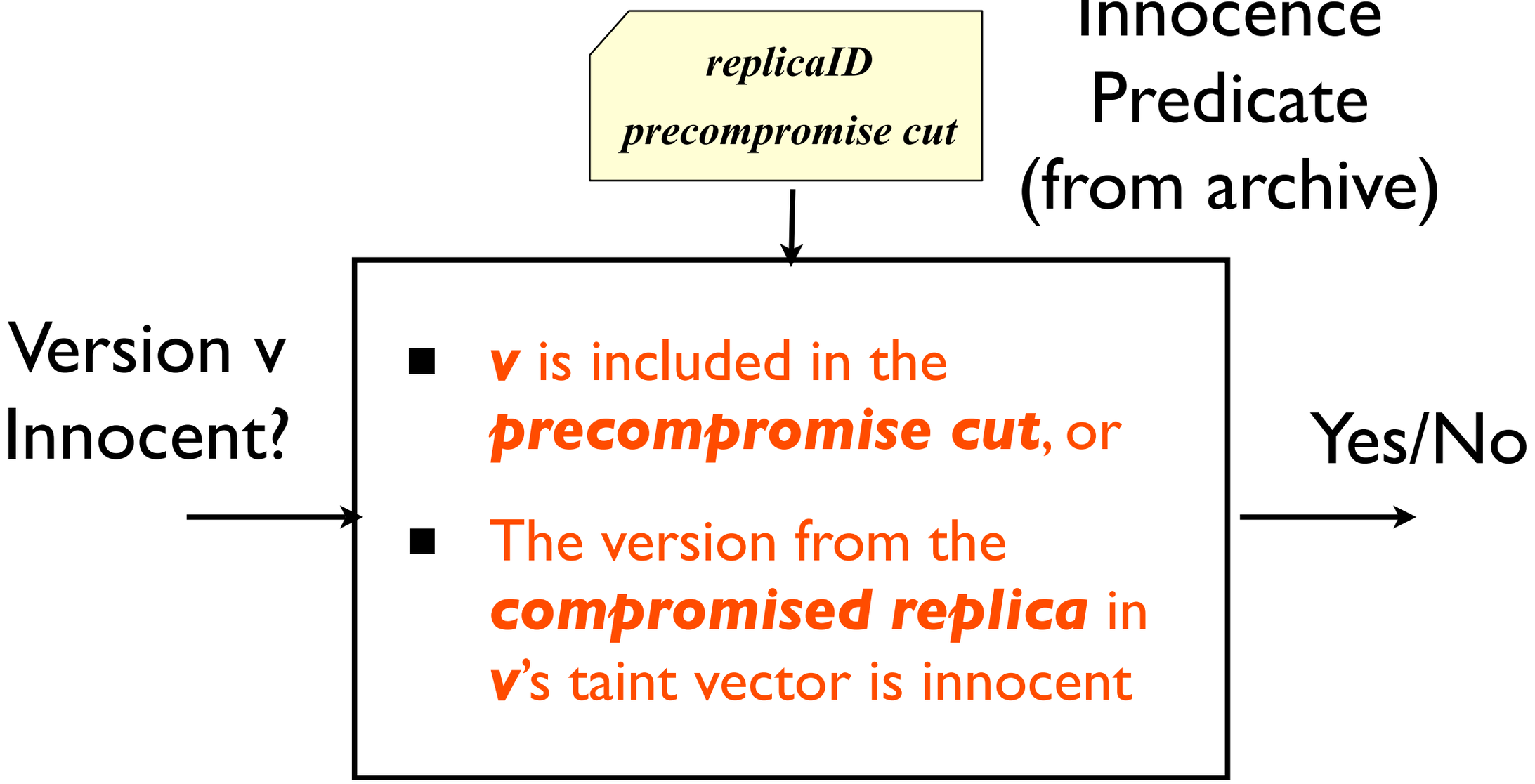
Replica-local Retention

- Replicas retain innocent versions
- Key mechanism: *innocence predicate*

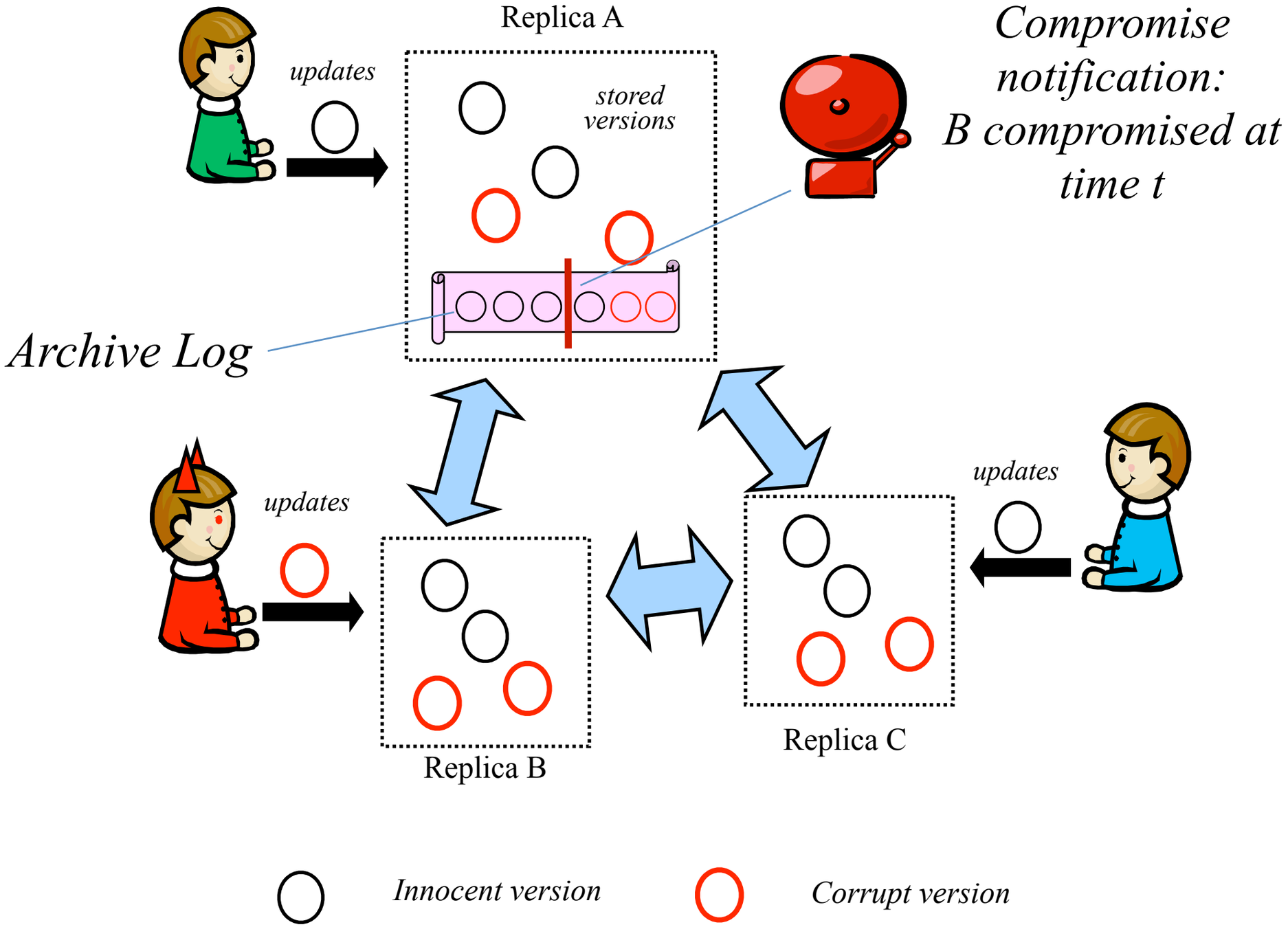
Innocent Version Identification at Archive



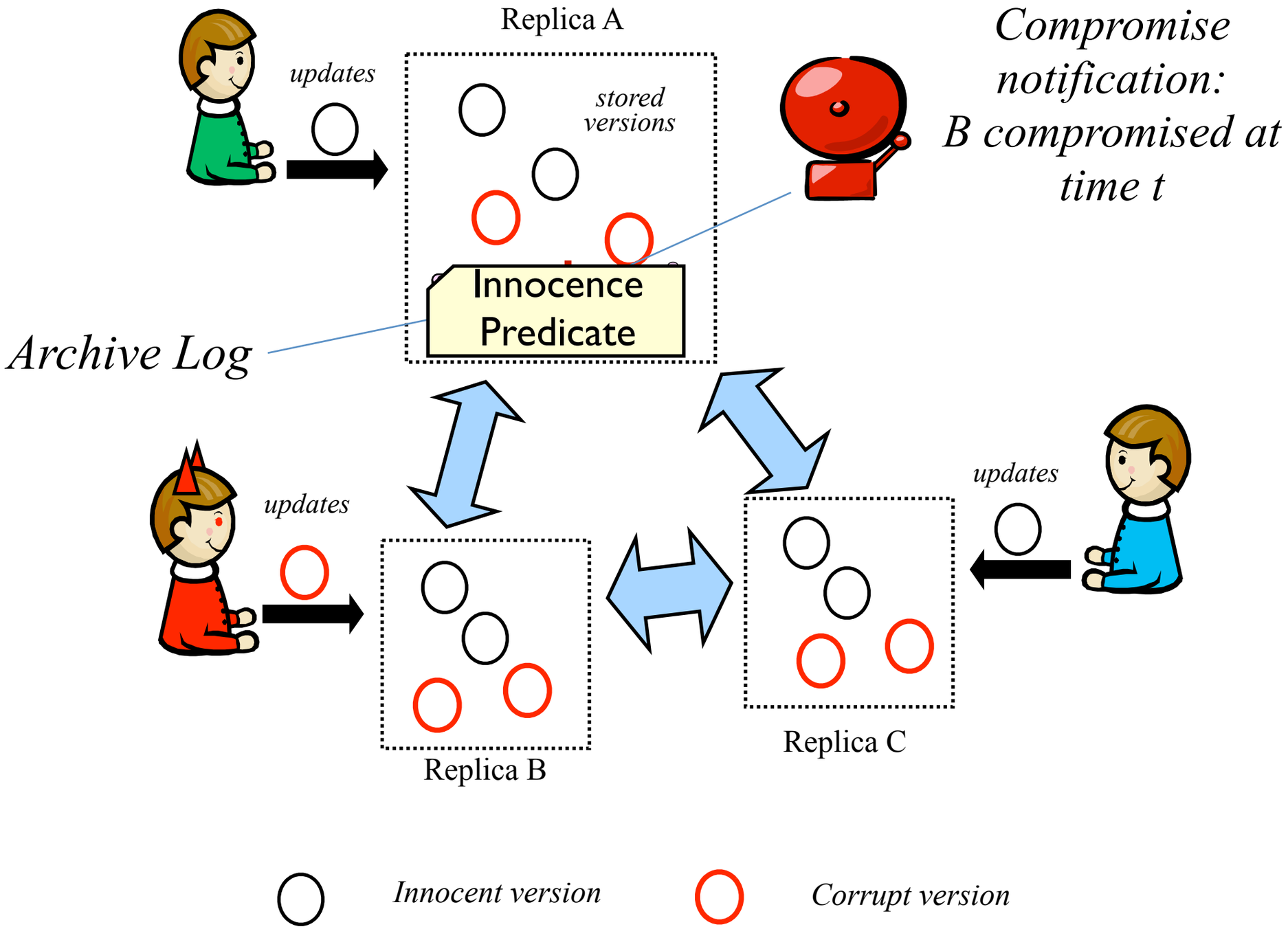
Innocence Predicate: Innocent Version Identification at Replica



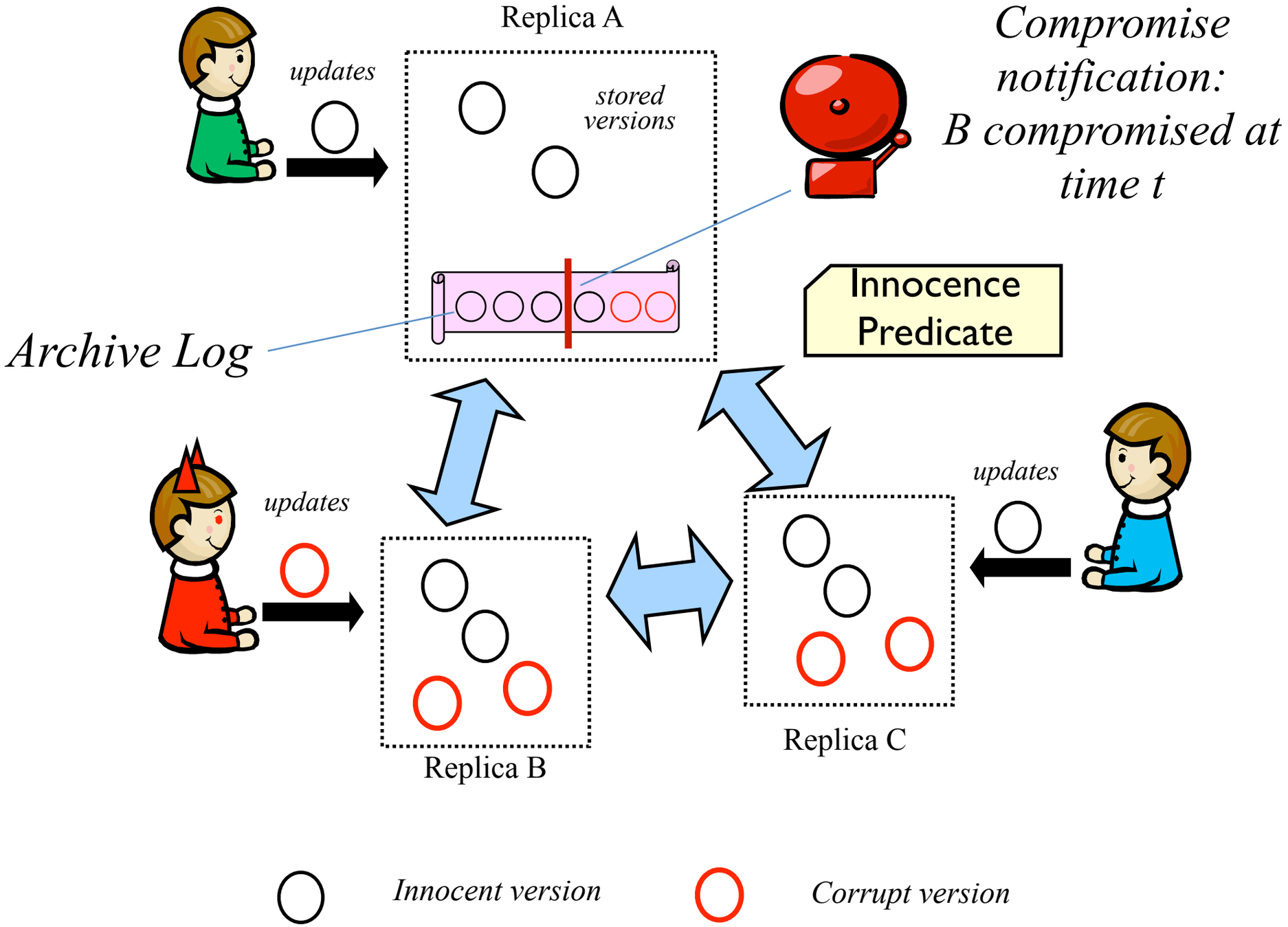
Replica-local Retention



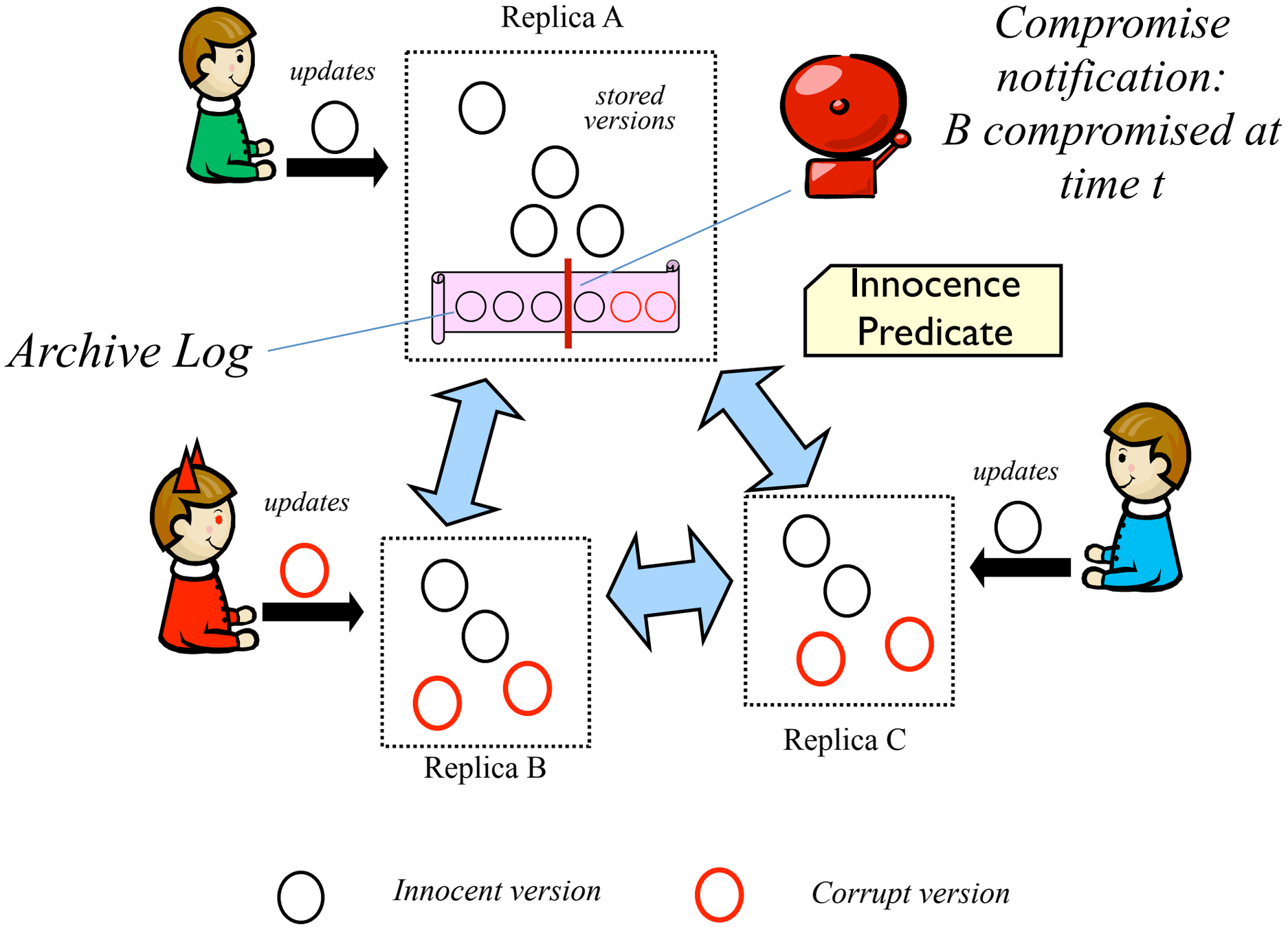
Replica-local Retention



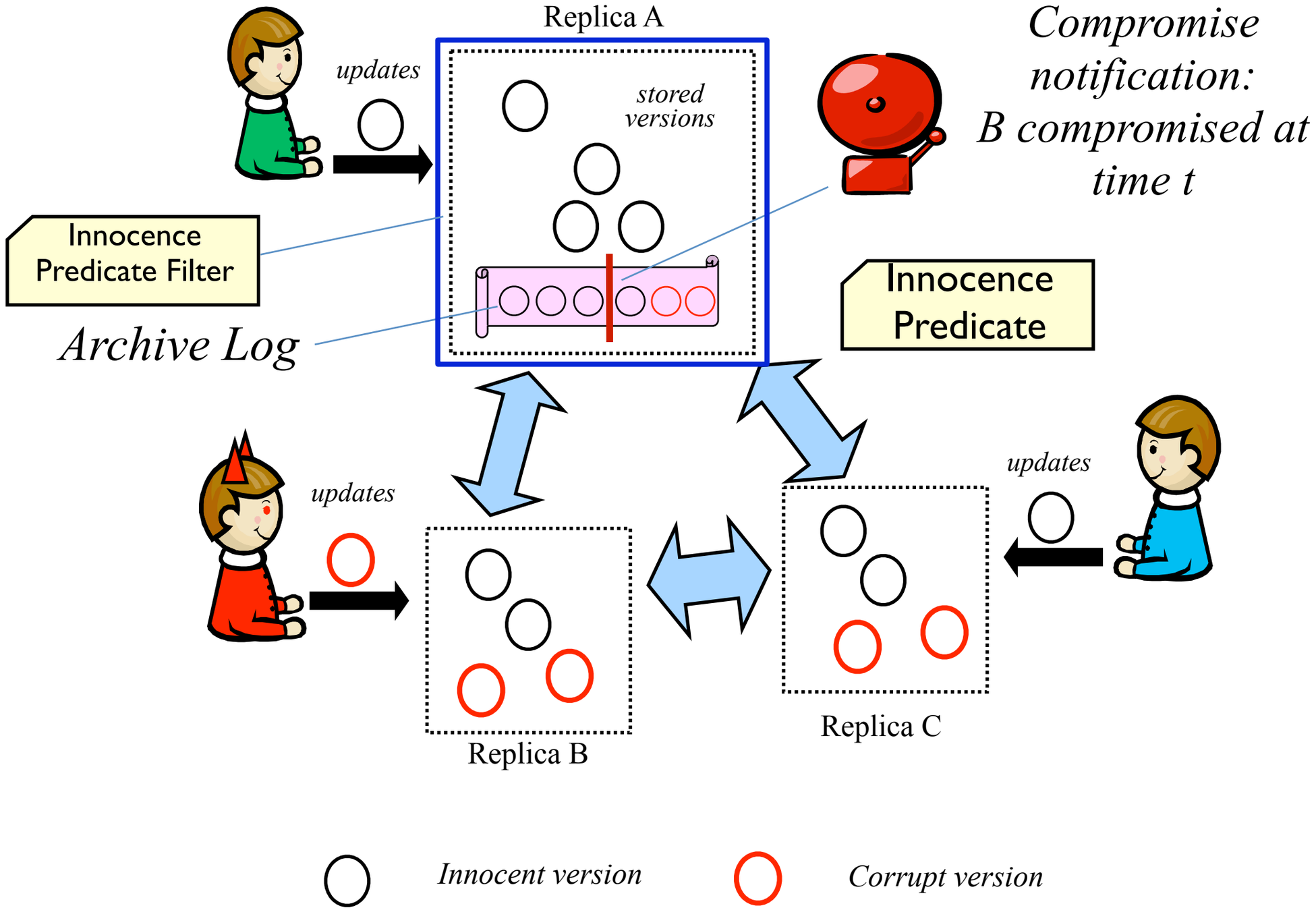
Replica-local Retention



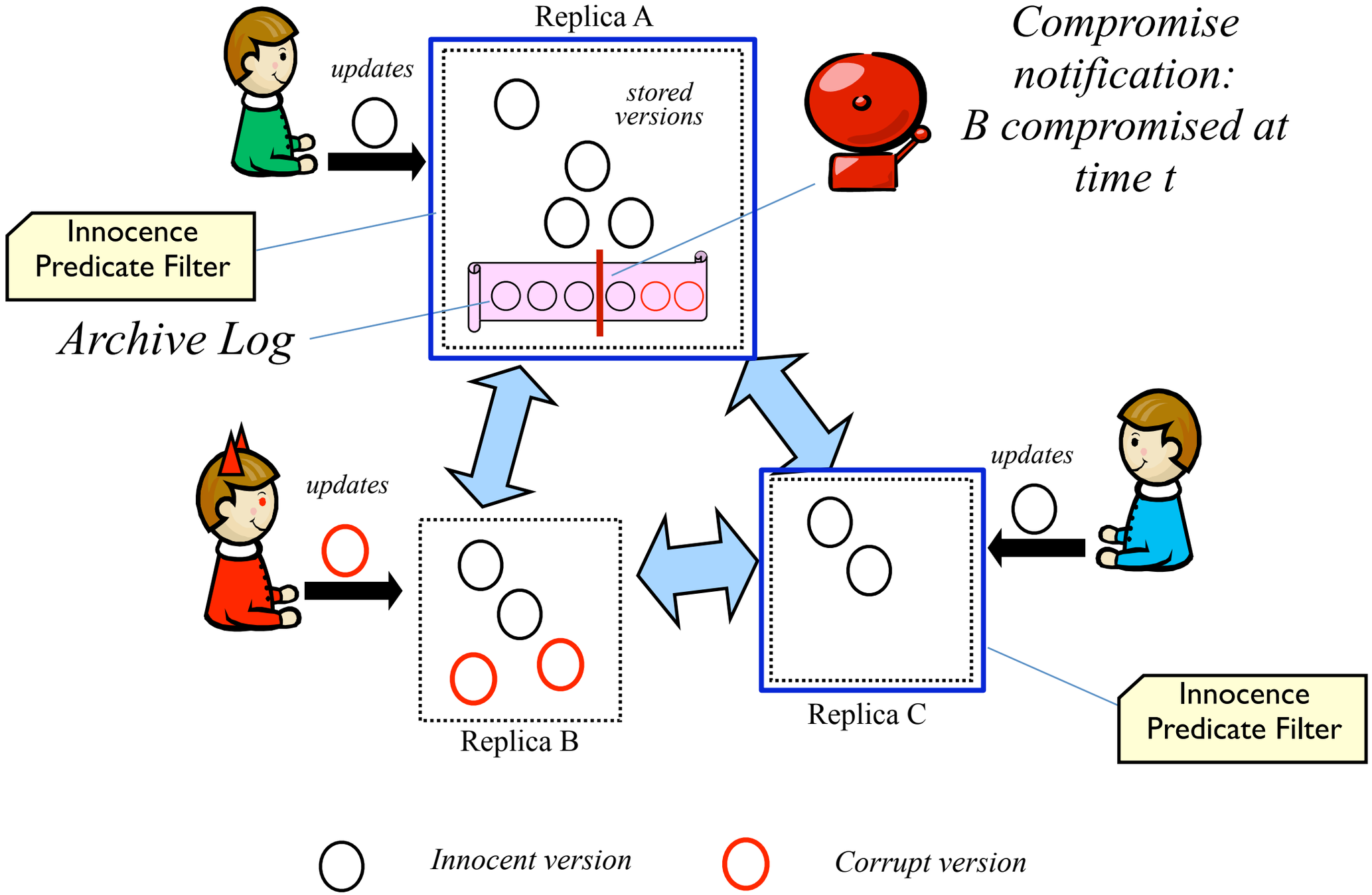
Replica-local Retention



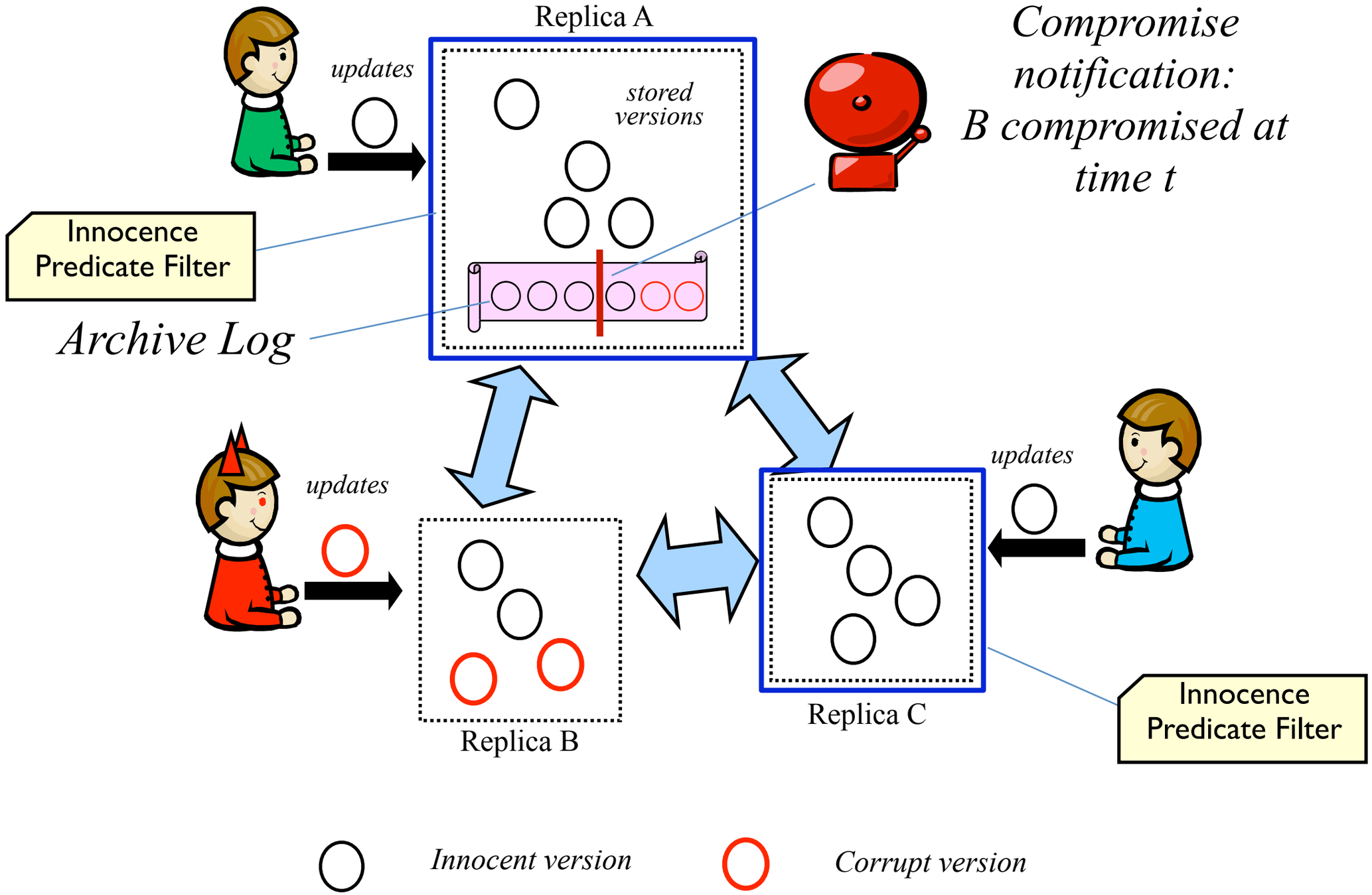
Replica-local Retention



Replica-local Retention



Replica-local Retention



Implementation

- Implemented on Cimbiosys
- Multiple compromises
  - Can recover simultaneously
- Multiple independent archives
  - Improves effectiveness, efficiency, and fault-tolerance

Evaluation

Evaluation

Setup

- 10 replicas and 1000 items
- 3000 updates overall
- 1000 updates after compromise
- Random updates and synchronizations
- 5 updates between synchronizations

Evaluation

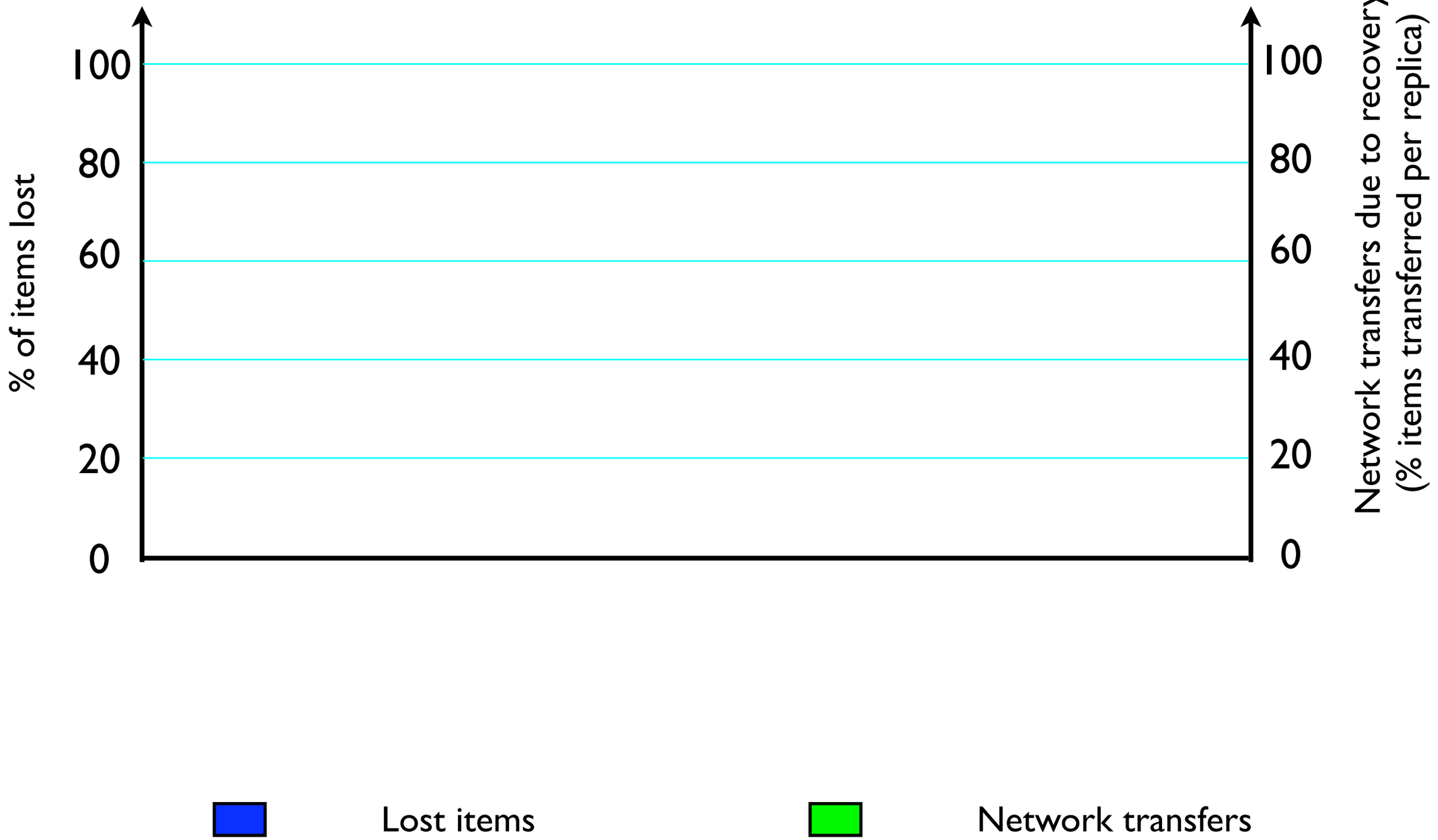
Setup

- 10 replicas and 1000 items
- 3000 updates overall
- 1000 updates after compromise
- Random updates and synchronizations
- 5 updates between synchronizations

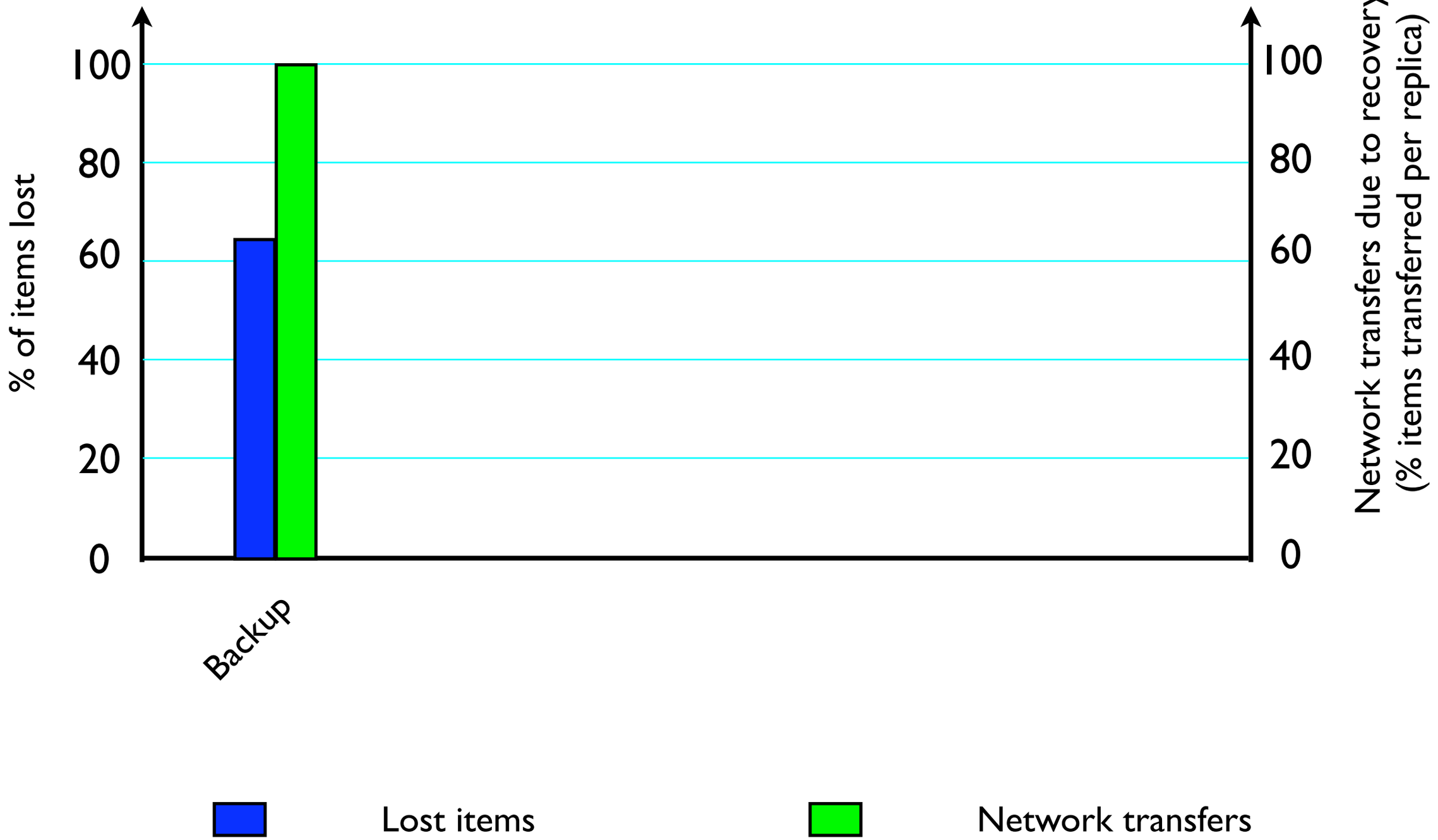
Metric

- (in)effectiveness: *lost items*
- (in)efficiency: *network overhead*

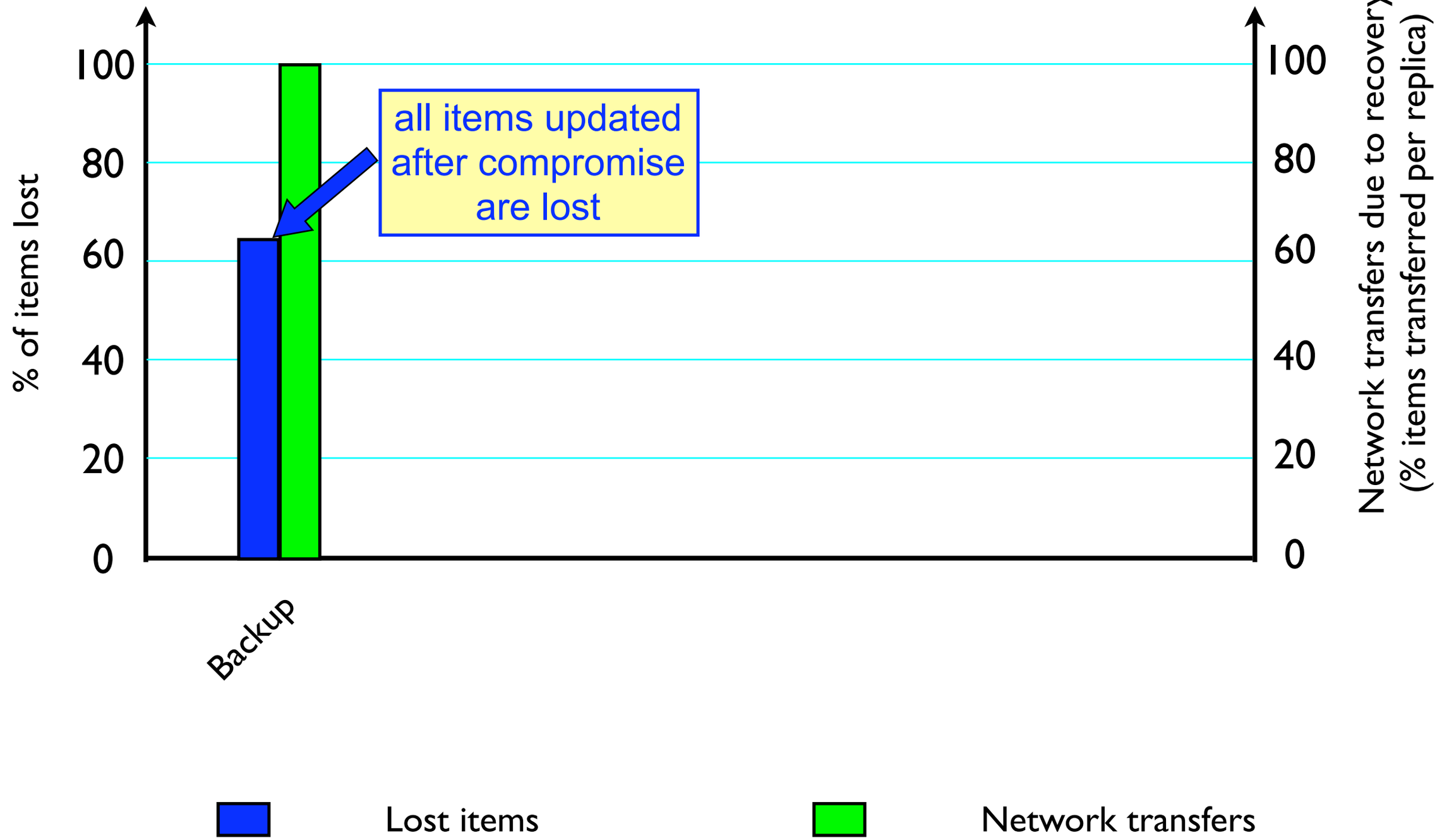
Effectiveness and Efficiency



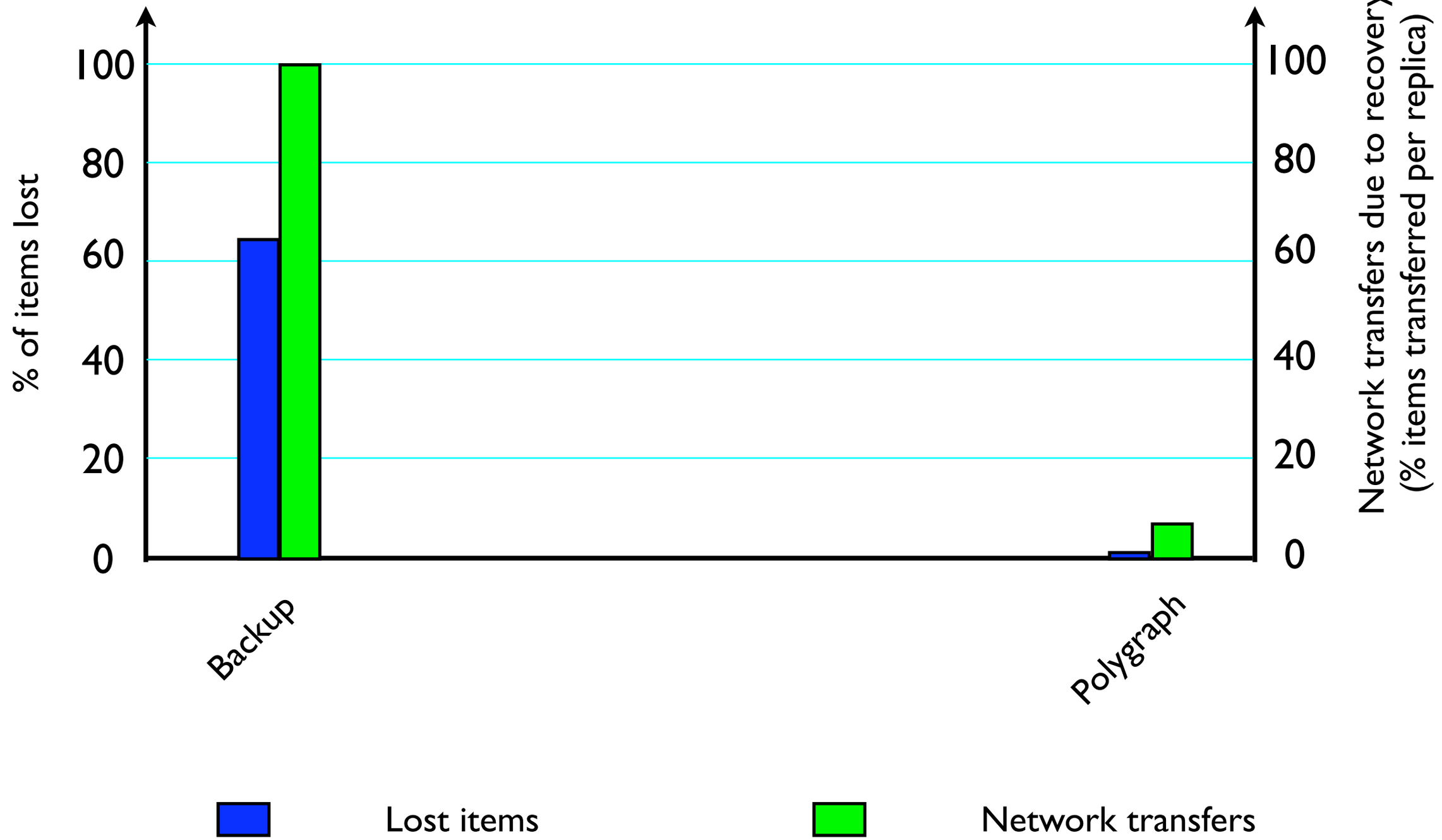
Effectiveness and Efficiency



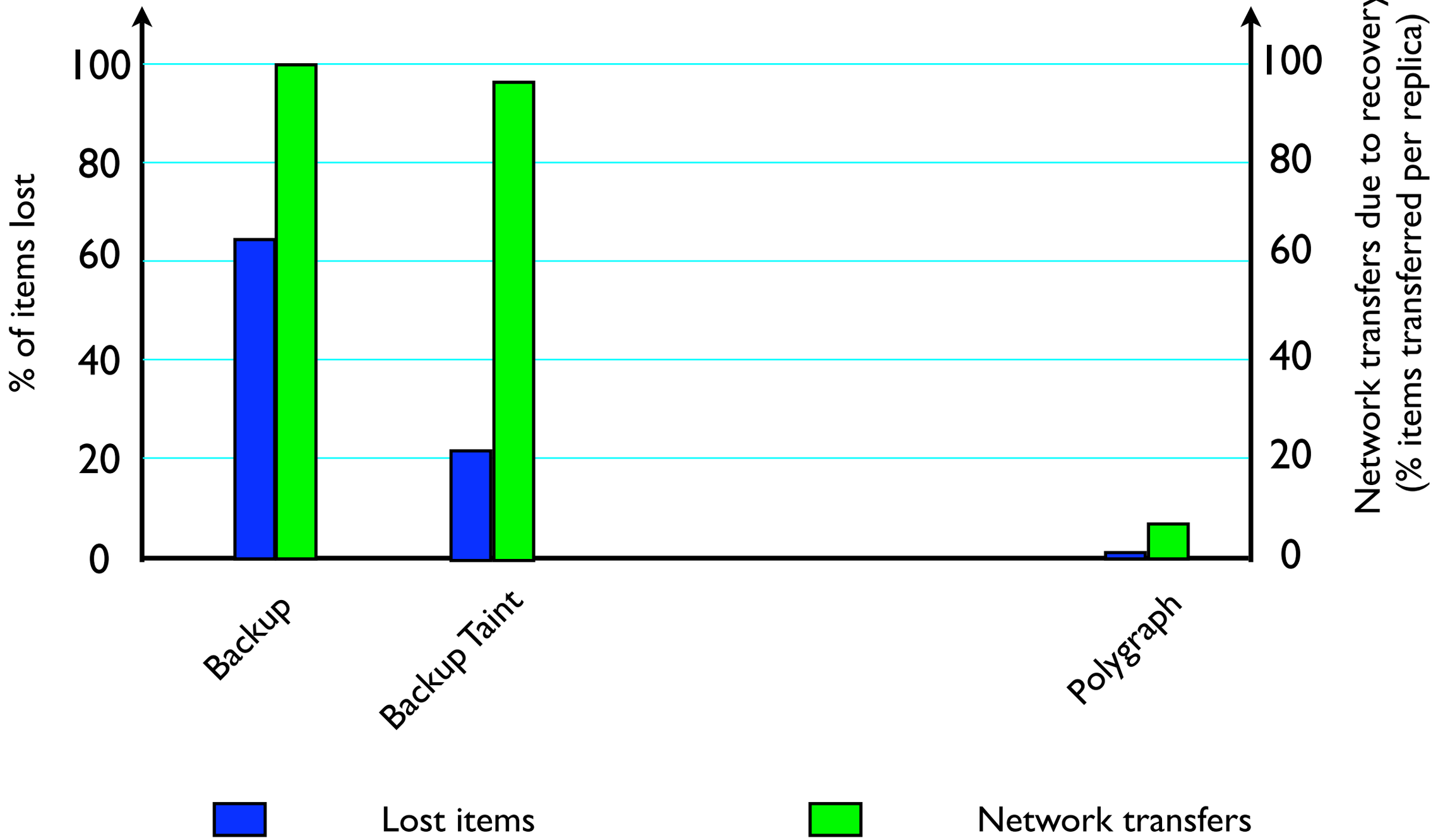
Effectiveness and Efficiency



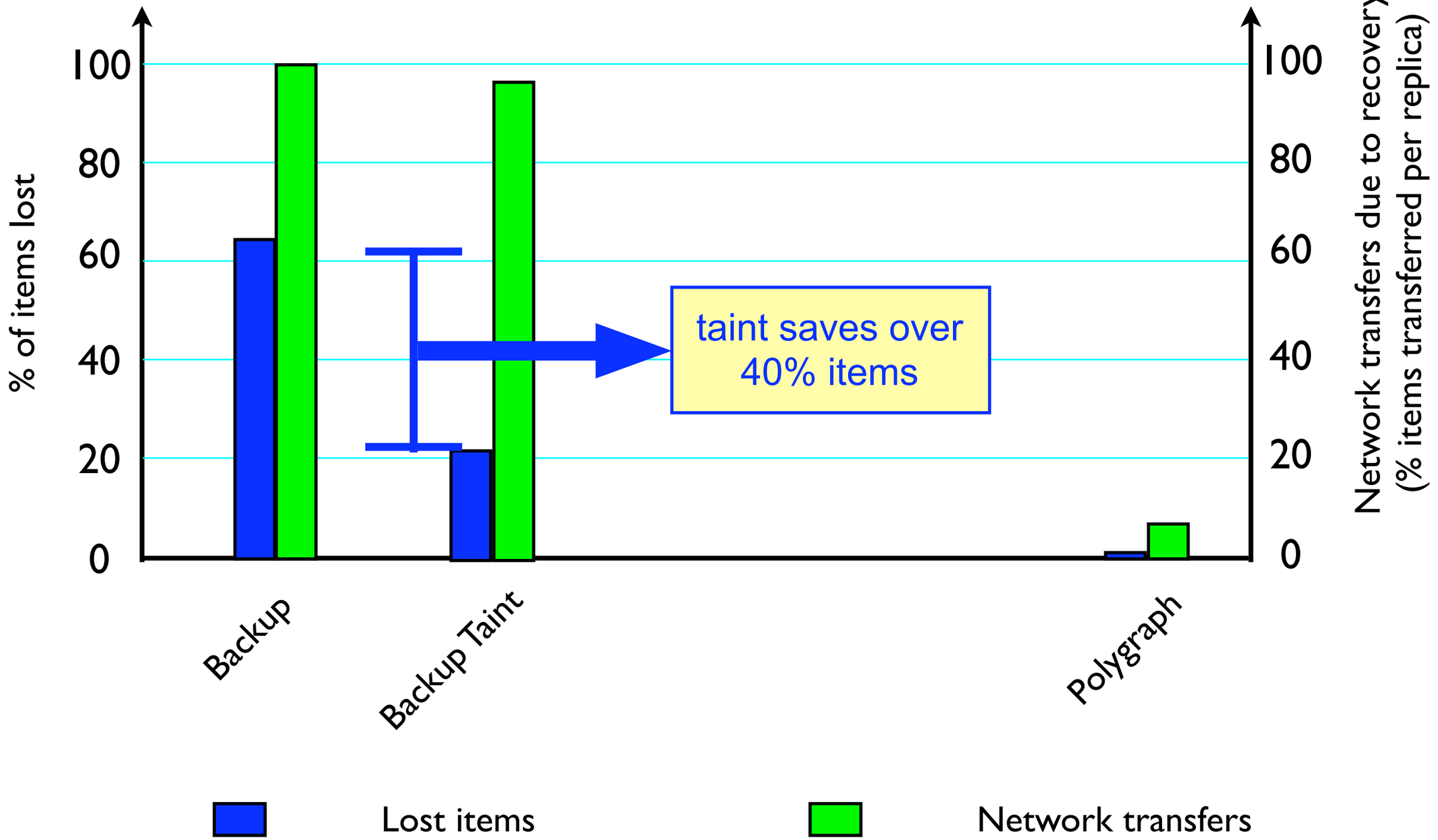
Effectiveness and Efficiency



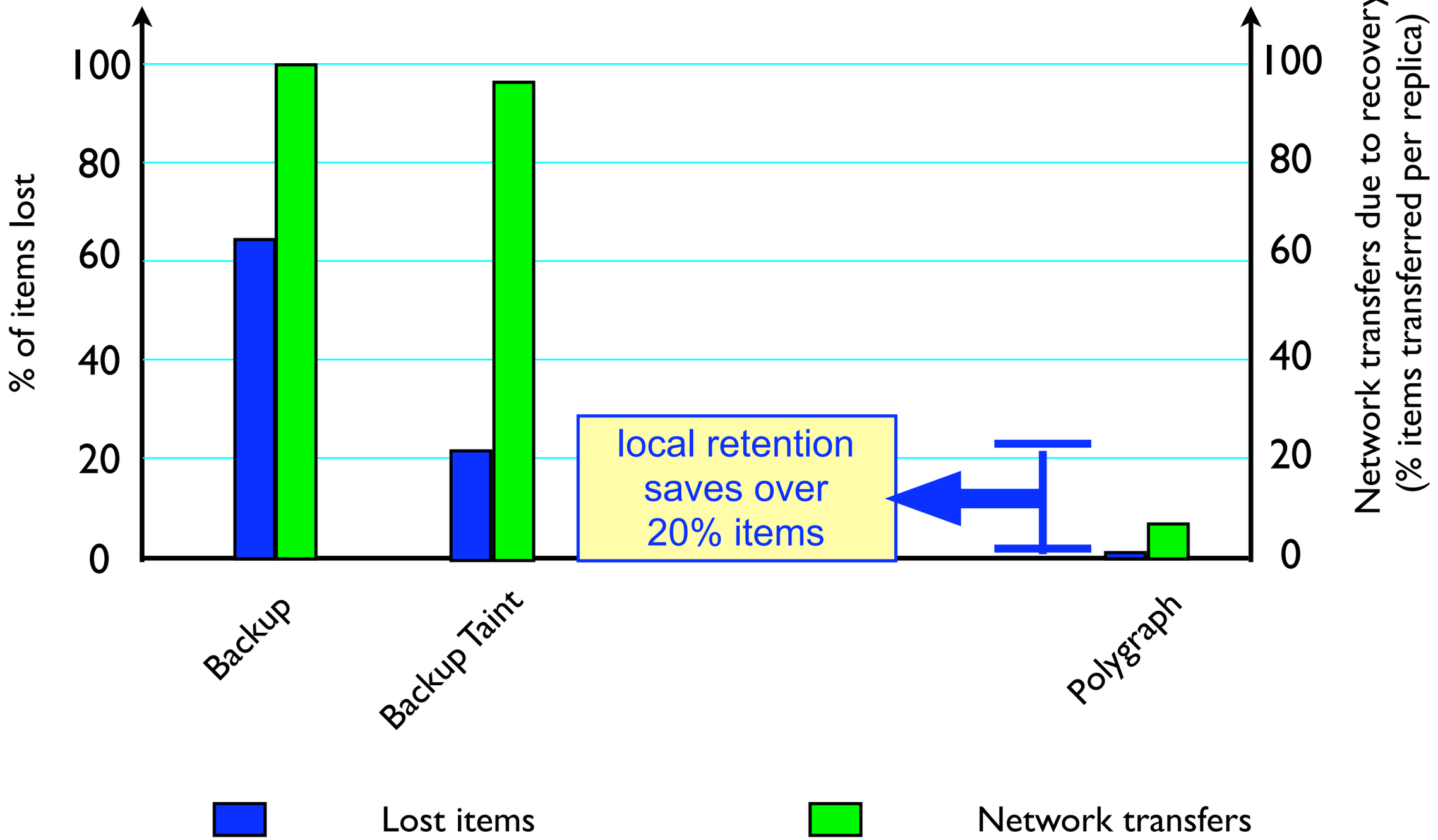
Effectiveness and Efficiency



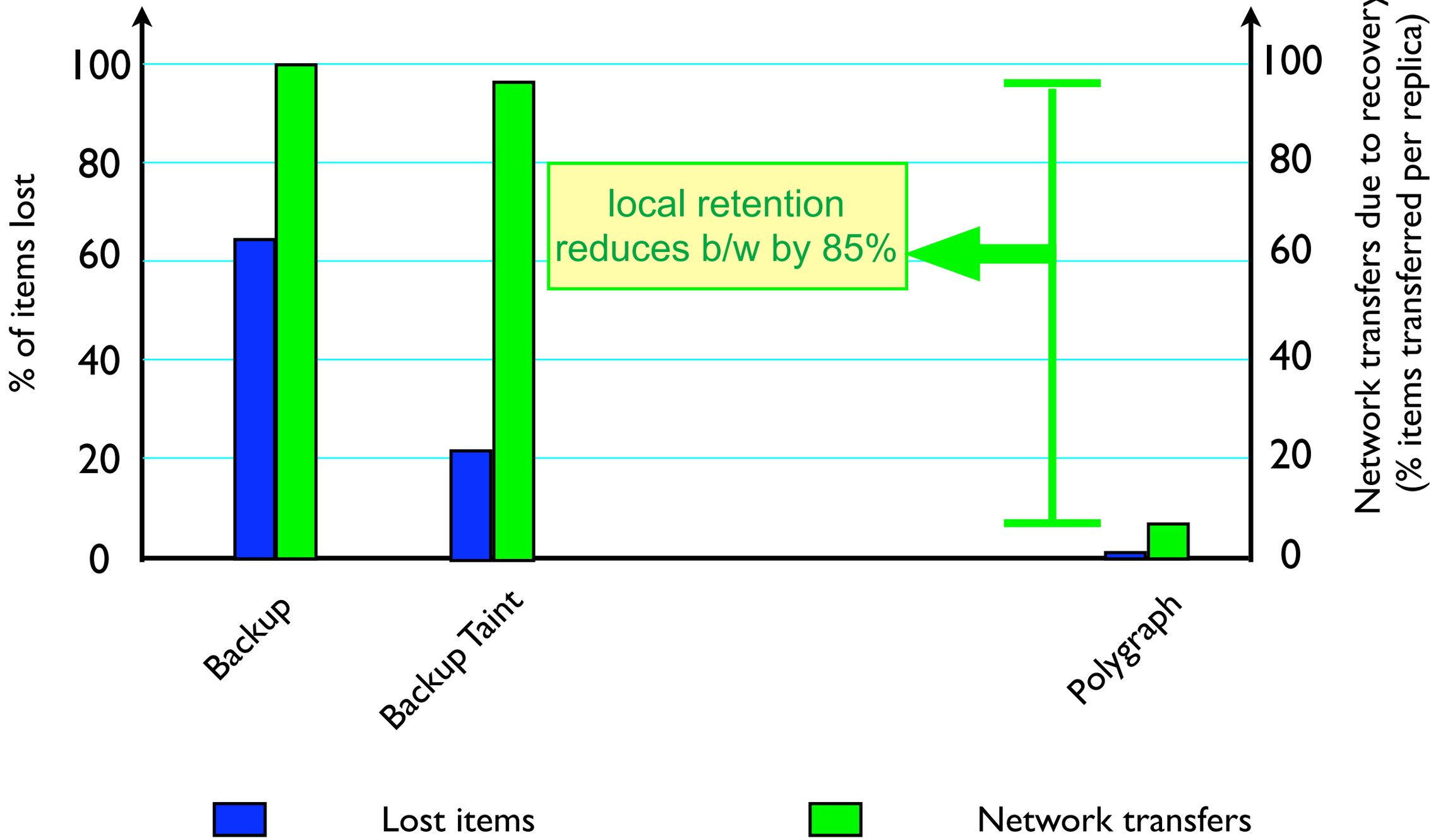
Effectiveness and Efficiency



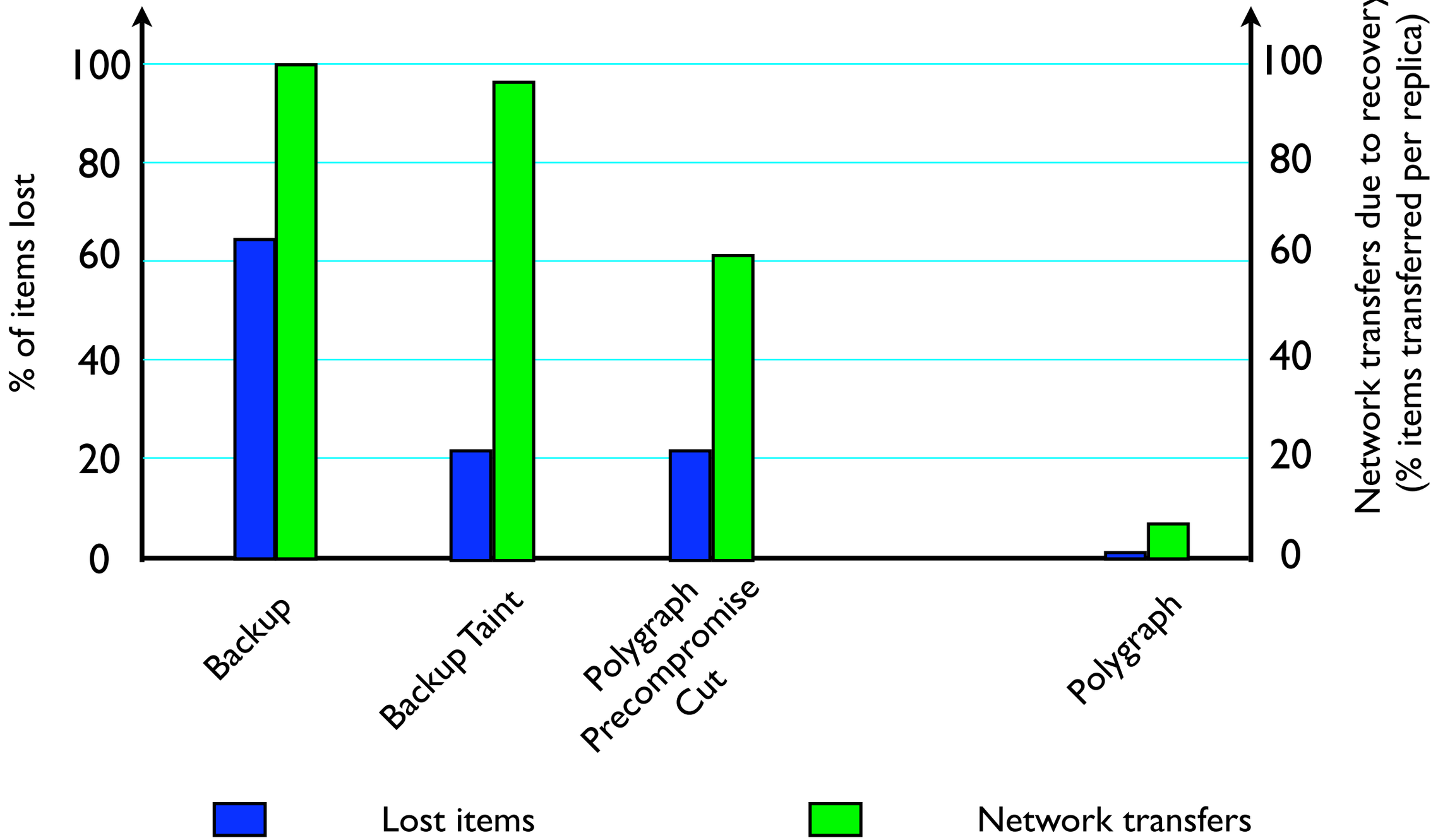
Effectiveness and Efficiency



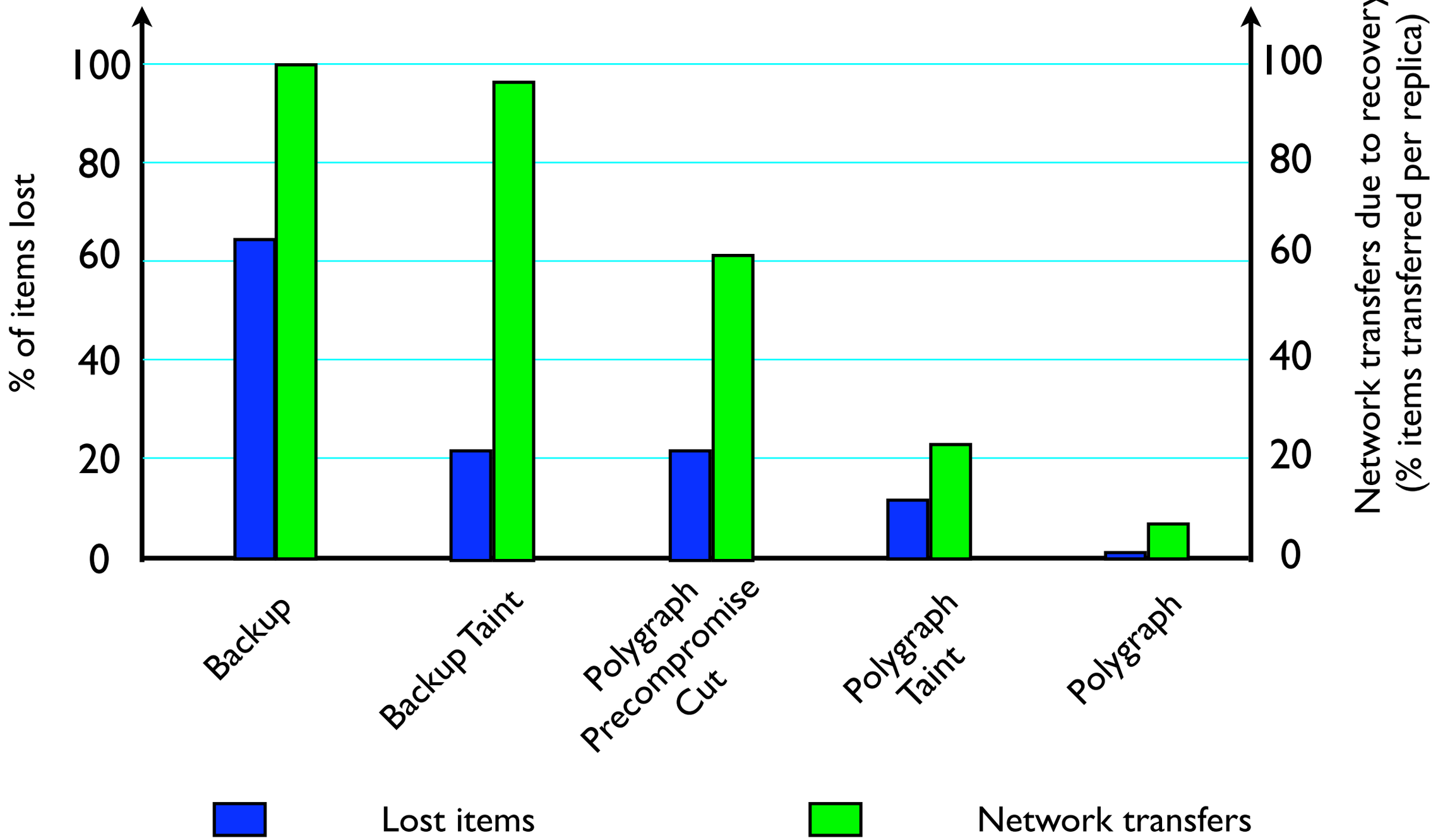
Effectiveness and Efficiency



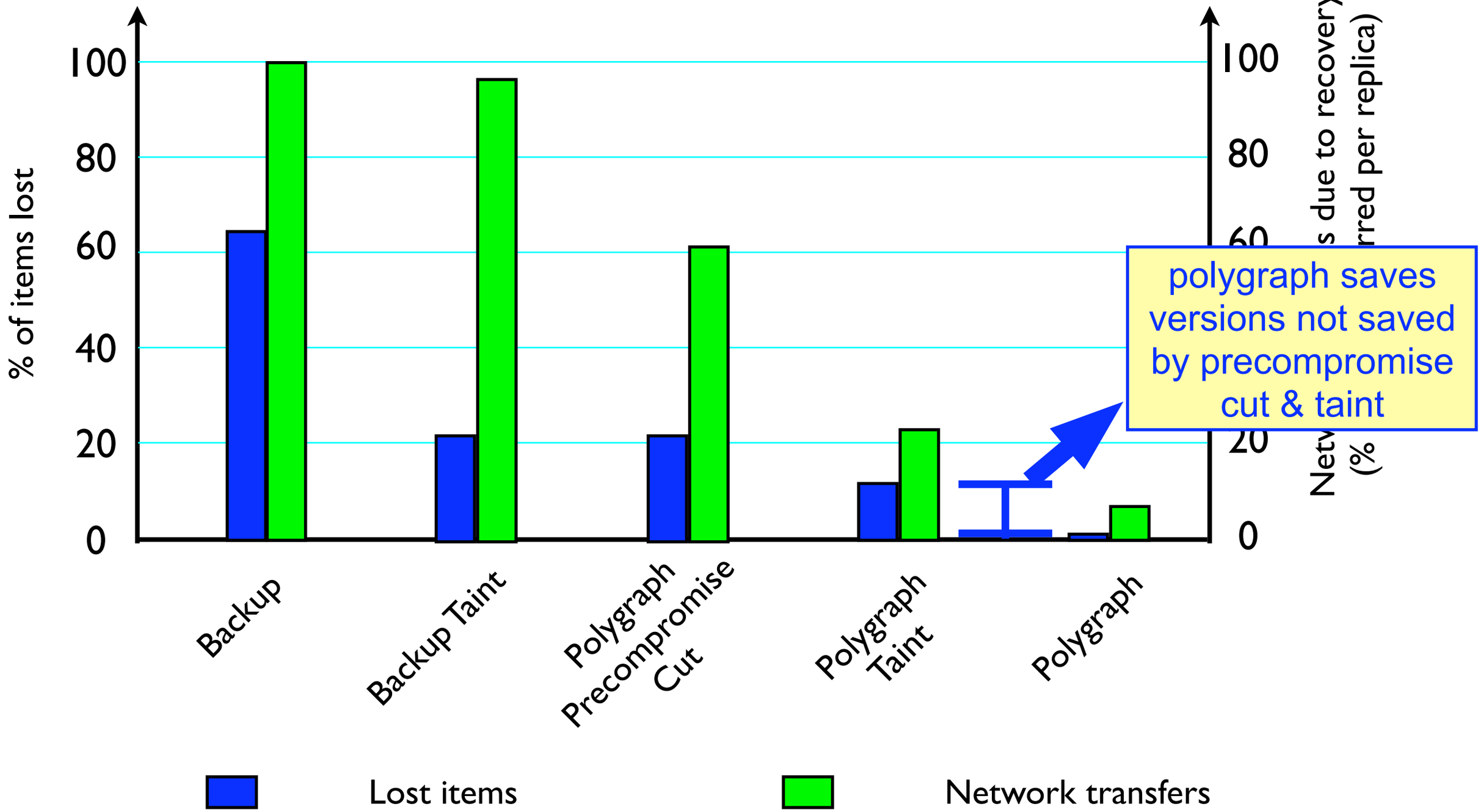
Effectiveness and Efficiency



Effectiveness and Efficiency



Effectiveness and Efficiency



Conclusion

- In a weakly consistent system, Polygraph reverses the effect of corrupt updates
  - Effective: retains most uncorrupted updates
  - Efficient: recovery uses less bandwidth
- Implemented on Cimbiosys replication system