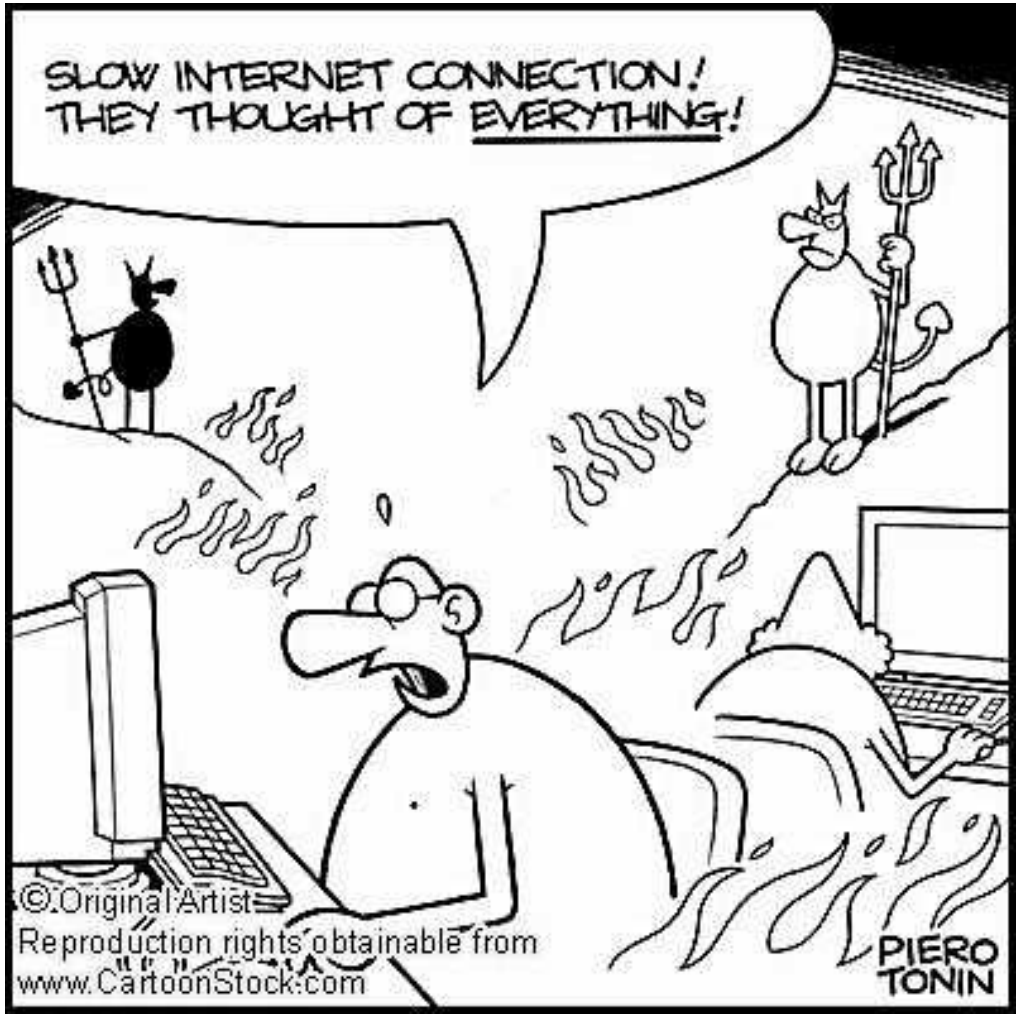
technology
from seed

Efficient Locally Trackable Deduplication in Replicated Systems

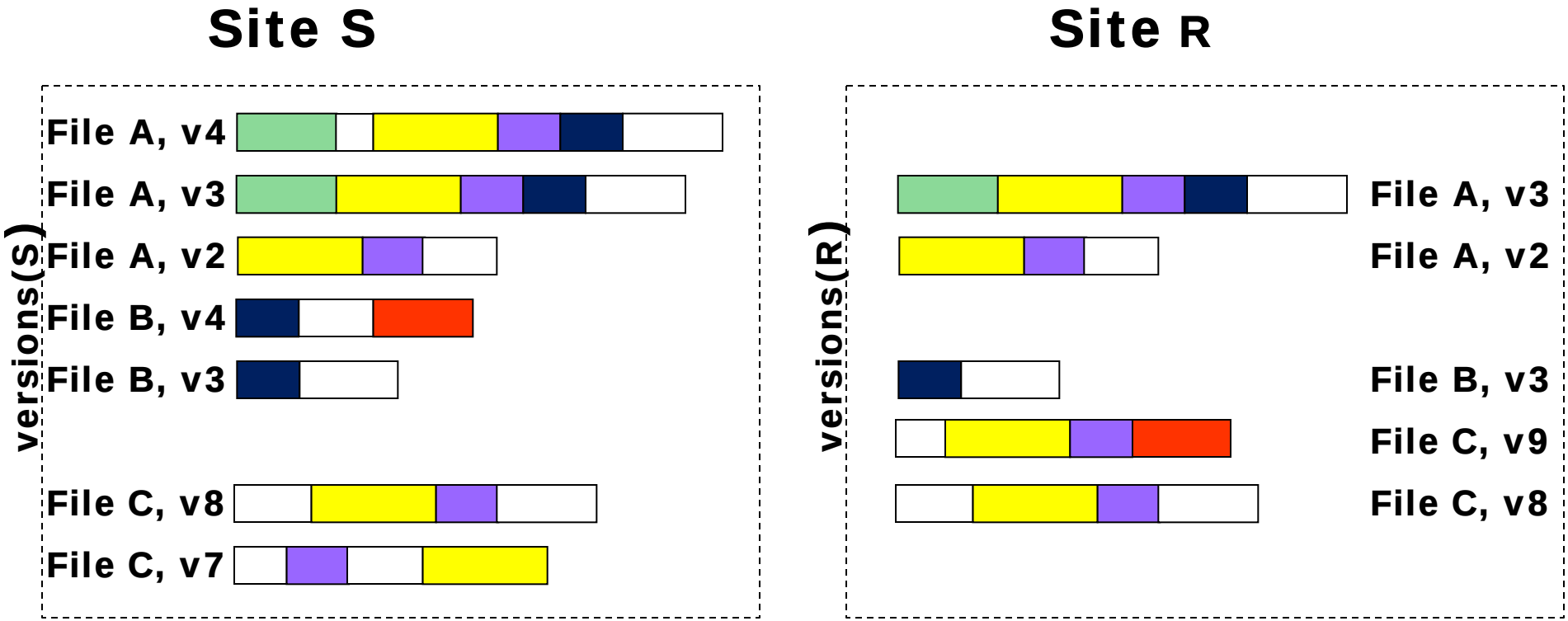
João Barreto and Paulo Ferreira
Distributed Systems Group
INESC-ID/Technical University Lisbon, Portugal

www.gsd.inesc-id.pt

Bandwidth remains scarce



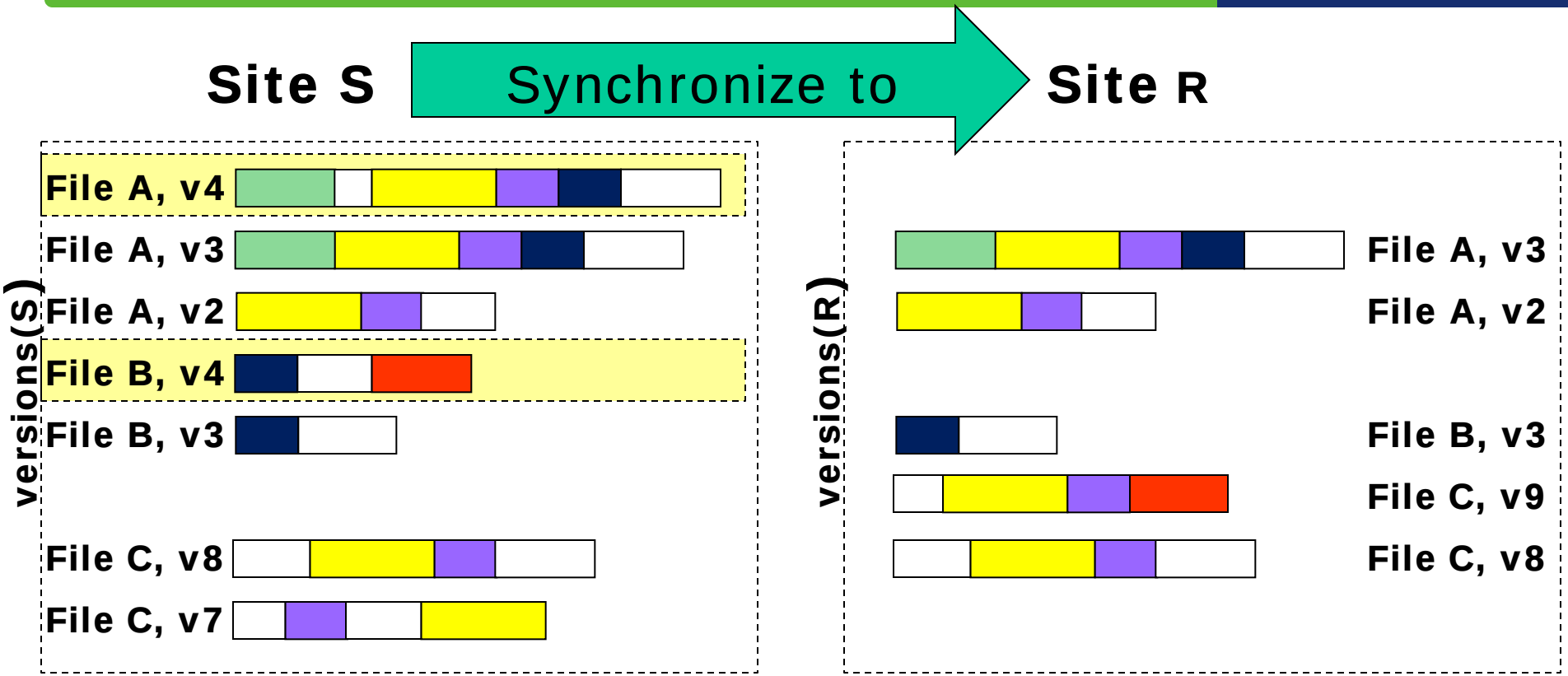
Collaboration through data replication



Distributed users share objects A, B and C

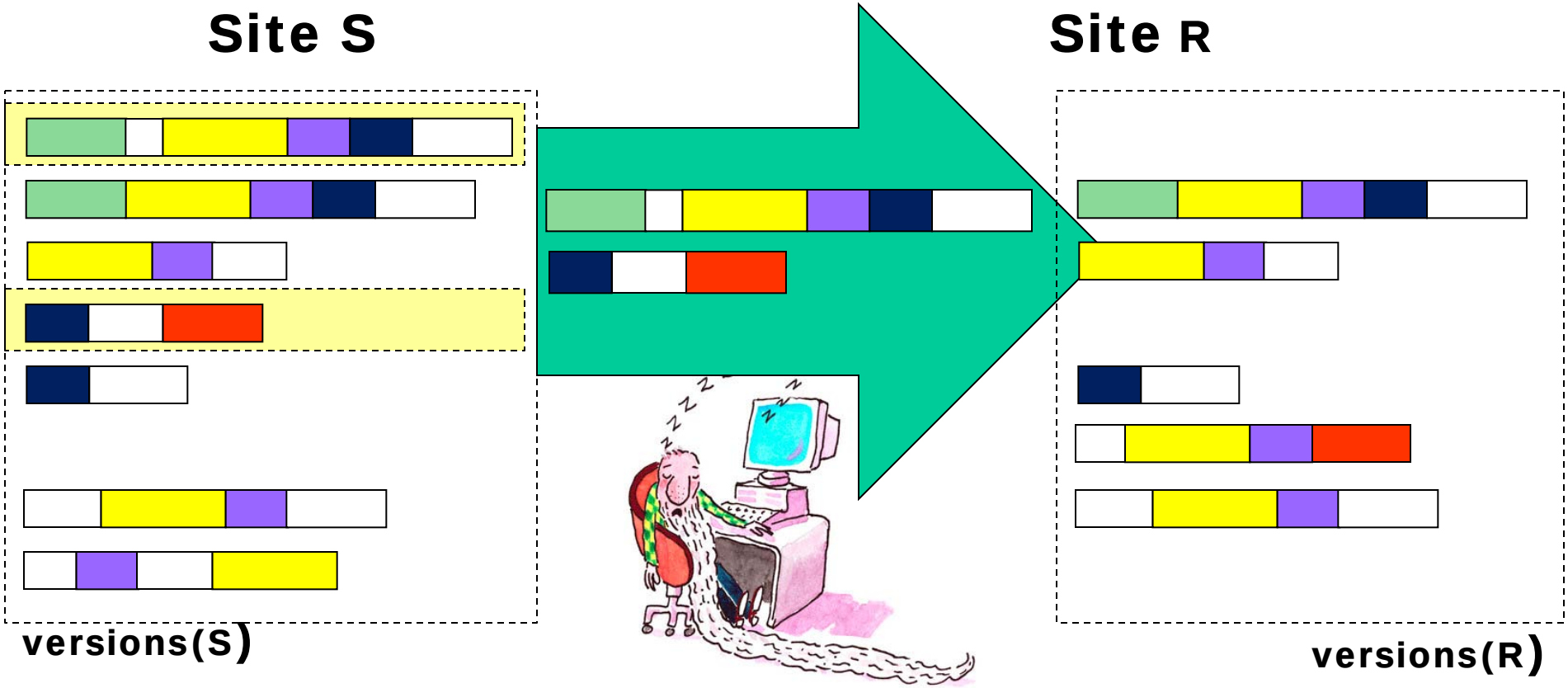
At each moment:
S stores versions(S) and R stores versions(R)

The bottleneck: synchronization



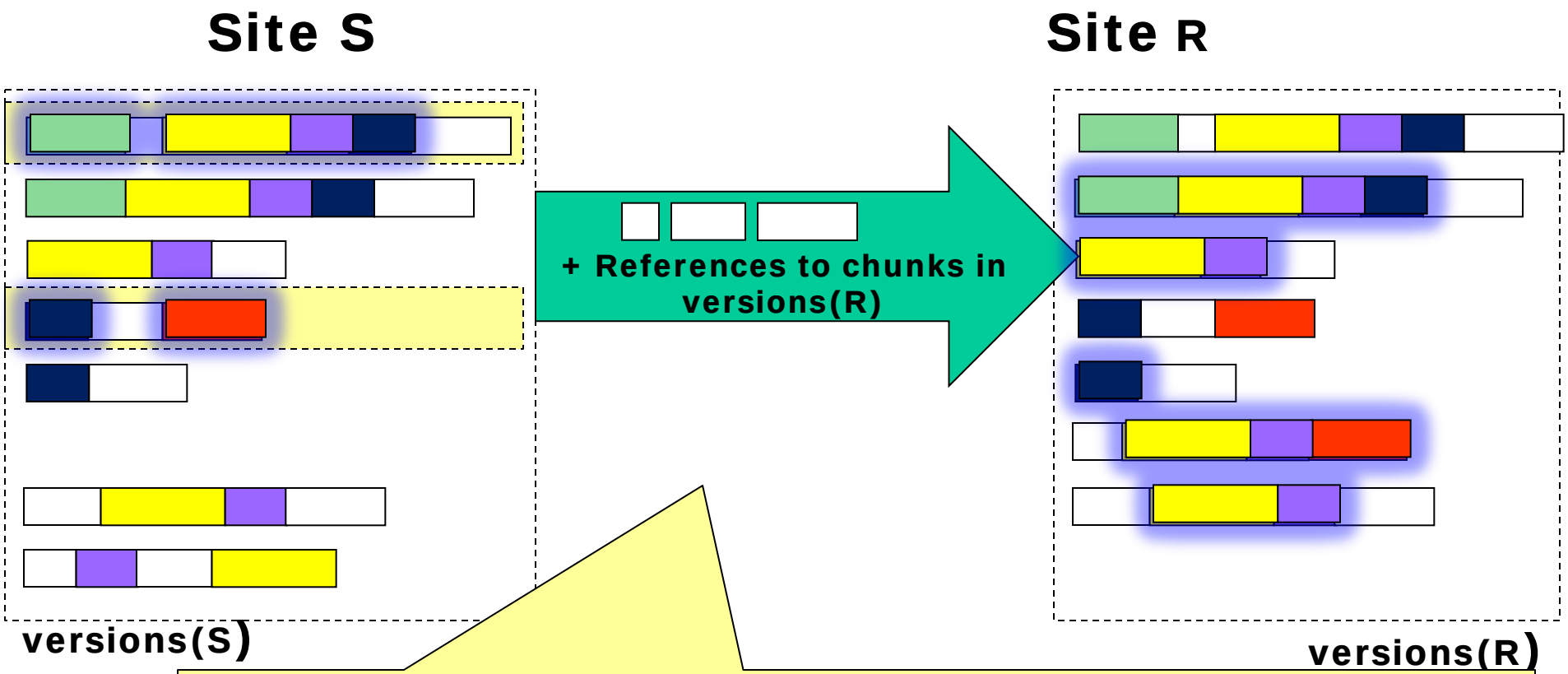
1. Determine which versions to transfer

The bottleneck: synchronization



2. Transfer versions

Deduplication: exploiting redundancy



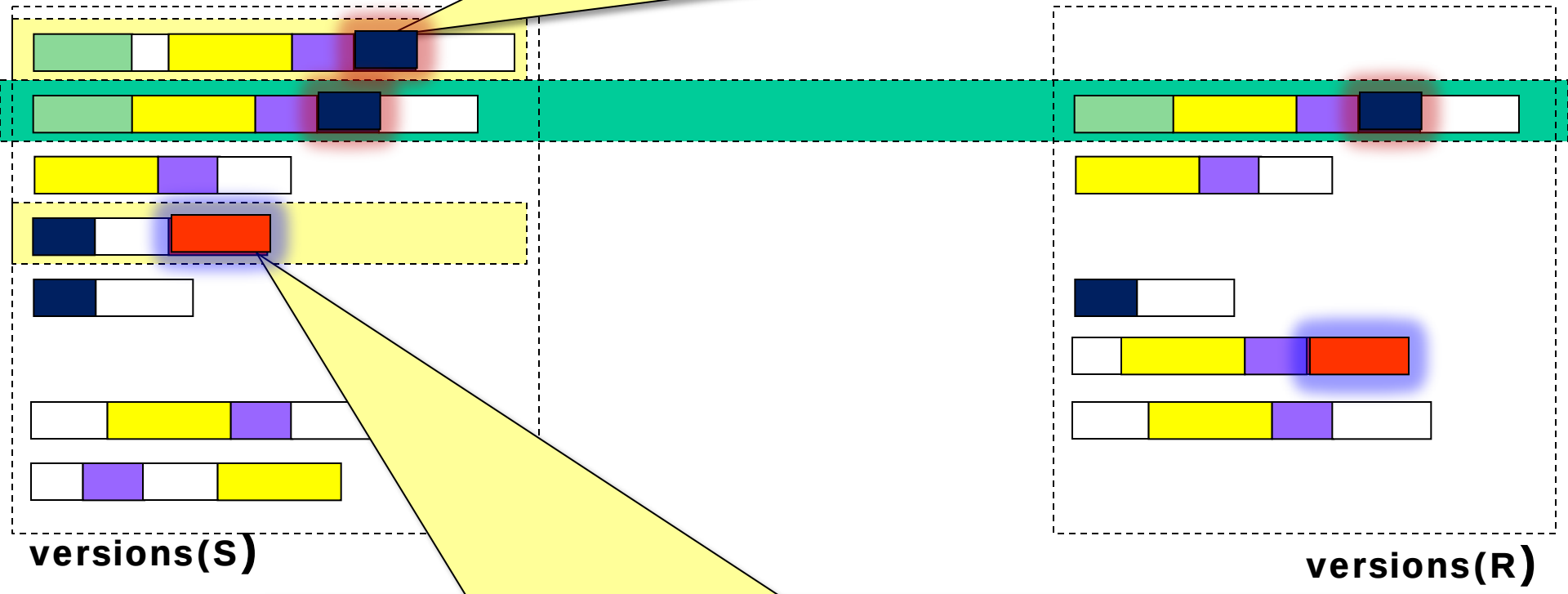
How to determine which chunks are redundant?

Locally trackable vs untrackable redunda

Locally Trackable Redundancy

chunk to transfer exists in some version that is both in versions(S) and versions(R)

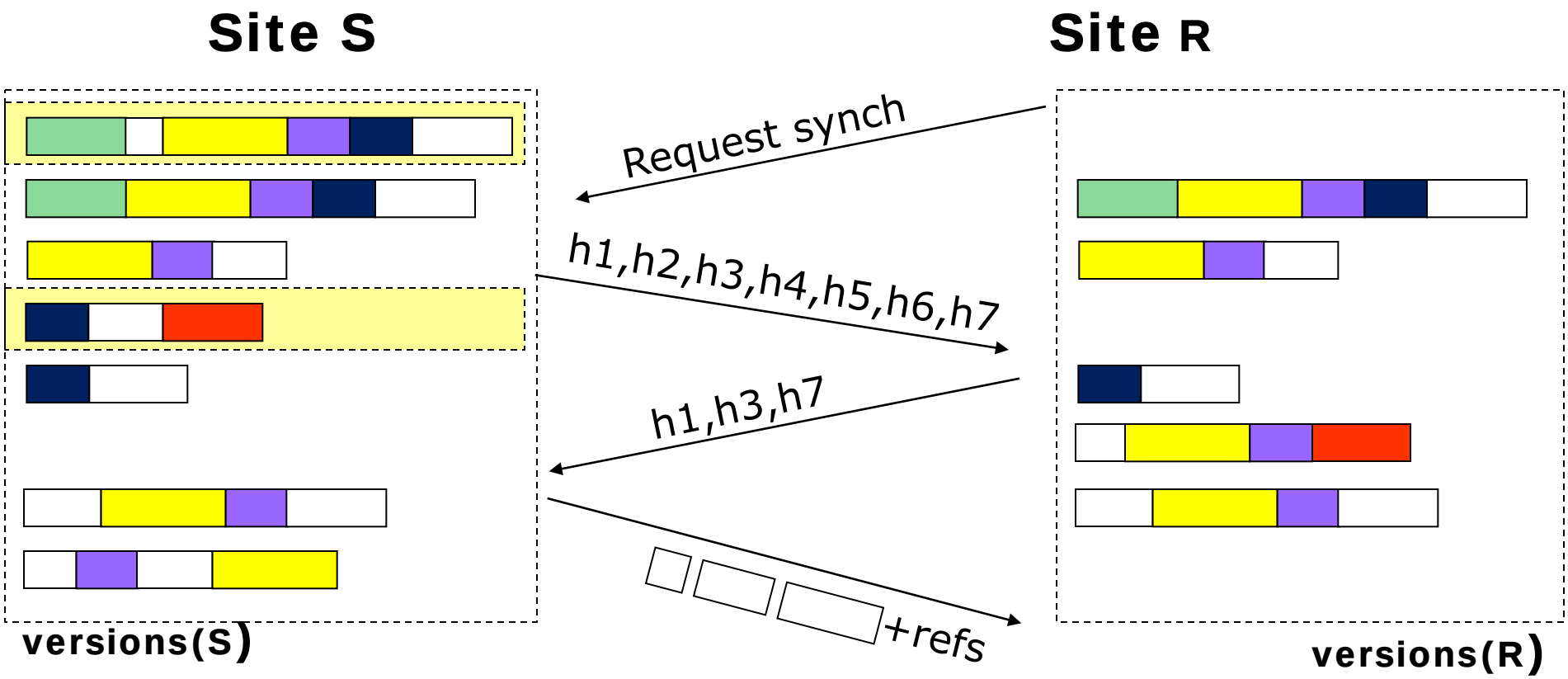
Sit



Locally Untrackable Redundancy

otherwise

Existing approaches: compare-by-hash

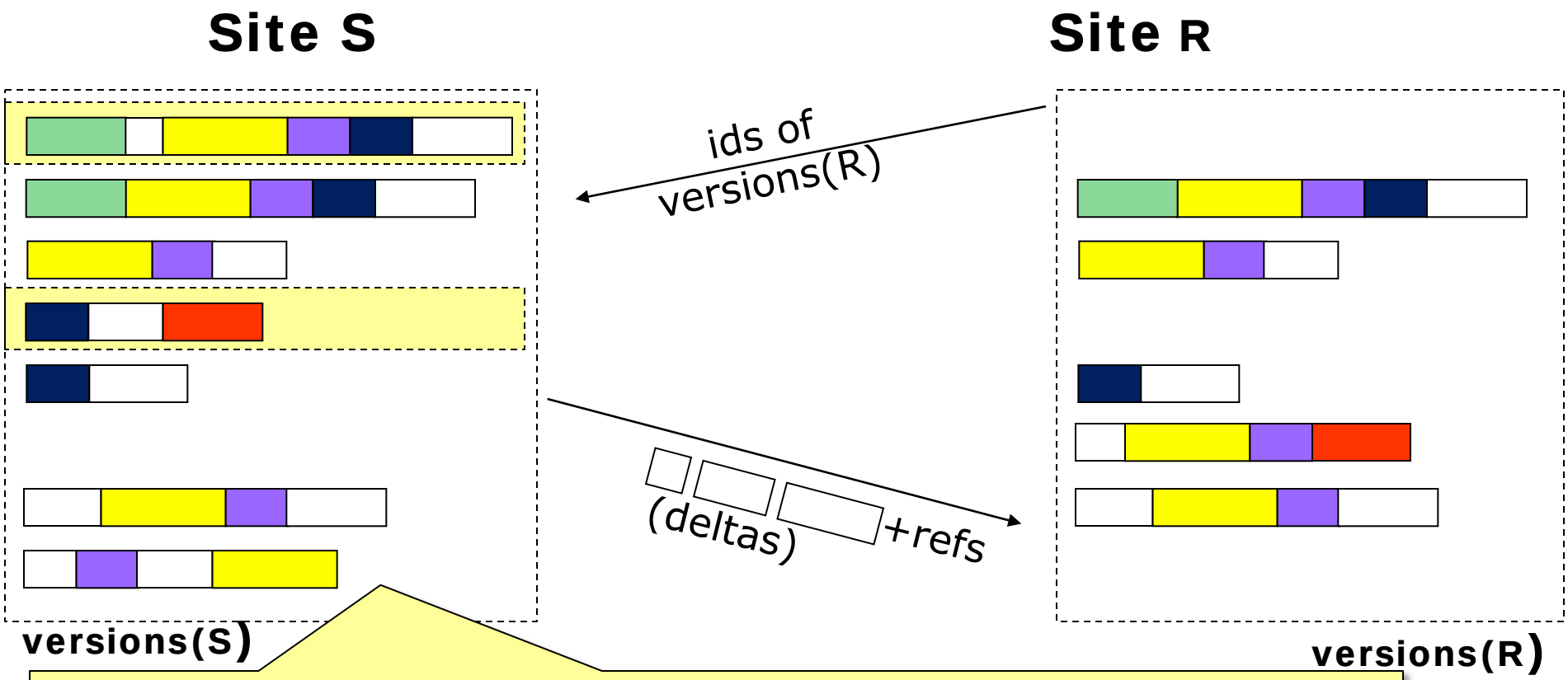


Existing approaches: advantages and shortcomings

Compare-by-hash

- ✓ Detects both locally trackable and untrackable redundancy
- ✓ Detects redundancy across any versions and/or objects
- ✗ Additional round-trip
- ✗ Limited precision:
  - smaller chunks may not compensate hash-exchange and hash-lookup overheads

Existing approaches: delta encoding



Calculate deltas from most recent versions versions(R) to each version to transfer.
Using local, high-precision algorithms.

Existing approaches: advantages and shortcomings

Compare-by-hash

- ✓ Detects both locally trackable and untrackable redundancy
- ✓ Detects redundancy across any versions and/or objects
- ✗ Additional round-trip
- ✗ Limited precision:
  - smaller chunks may not compensate hash-exchange and hash-lookup overheads

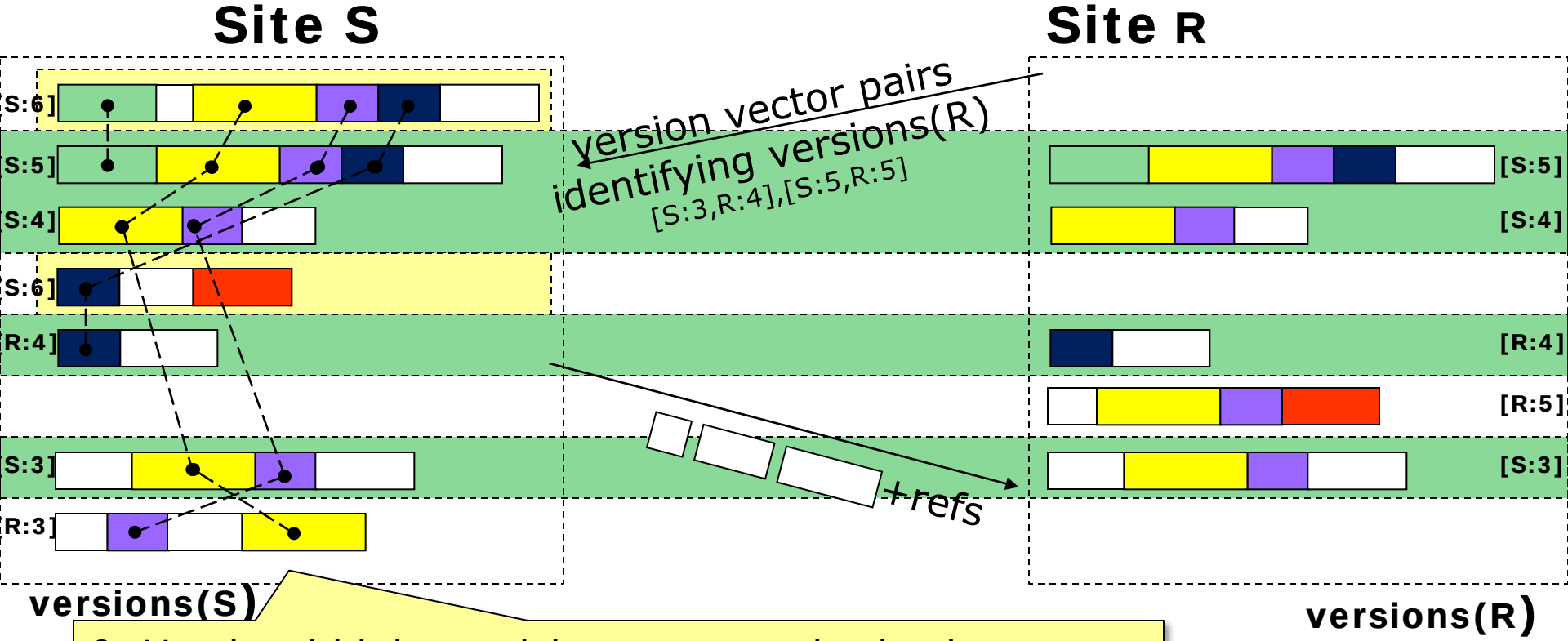
Delta Encoding

- ✗ Only detects locally trackable redundancy
- ✗ Limited to pairs of versions
- ✓ High-precision local redundancy detection
- ✓ Redundancy detection can occur ahead of transfer time

Can we devise a solution that borrows the advantages from both approaches?

Our contribution: redFS

technology
from seed



0. Use local high-precision compare-by-hash algorithm to pre-compute local redundancy relations

2. Determine C = versions(S) ∩ versions(R)

3. For each chunk to transfer, if the chunk is also in some version in C, simply send a reference to that version

INSTITUTO SUPERIOR TÉCNICO

What does redFS achieve?

technology
from seed

Compare-by-hash

- ✓ Detects both locally trackable and untrackable redundancy
- ✓ Detects redundancy across any versions and/or objects
- ✗ Additional round-trip
- ✗ Limited precision:
  - smaller chunks may not compensate hash-exchange and hash-lookup overheads

Delta Encoding

- ✗ Only detects locally trackable redundancy
- ✗ Limited to pairs of versions
- ✓ High-precision local redundancy detection
- ✓ Redundancy detection can occur ahead of transfer time
- ✓ Simple protocol

What does redFS achieve?

technology
from seed

inesc id lisboa

Compare-by-hash

- ✓ Detects both locally trackable and untrackable redundancy
- ✓ Detects redundancy across any versions and/or objects
- ✗ Additional round-trip
- ✗ Limited precision:
  - smaller chunks may not compensate hash-exchange and hash-lookup overheads

Delta Encoding

- ✗ Only detects locally trackable redundancy
- ✗ Limited to pairs of versions
- ✓ High-precision local redundancy detection
- ✓ Redundancy detection can occur ahead of transfer time
- ✓ Simple protocol

What does redFS achieve?

technology
from seed

Simpler protocol,
Able to detect
finer-grained
redundancy

Delta Encoding

redFS

More
complicated
protocol,
Limited
precision

Compare-by-hash

Detectable forms of redudancy

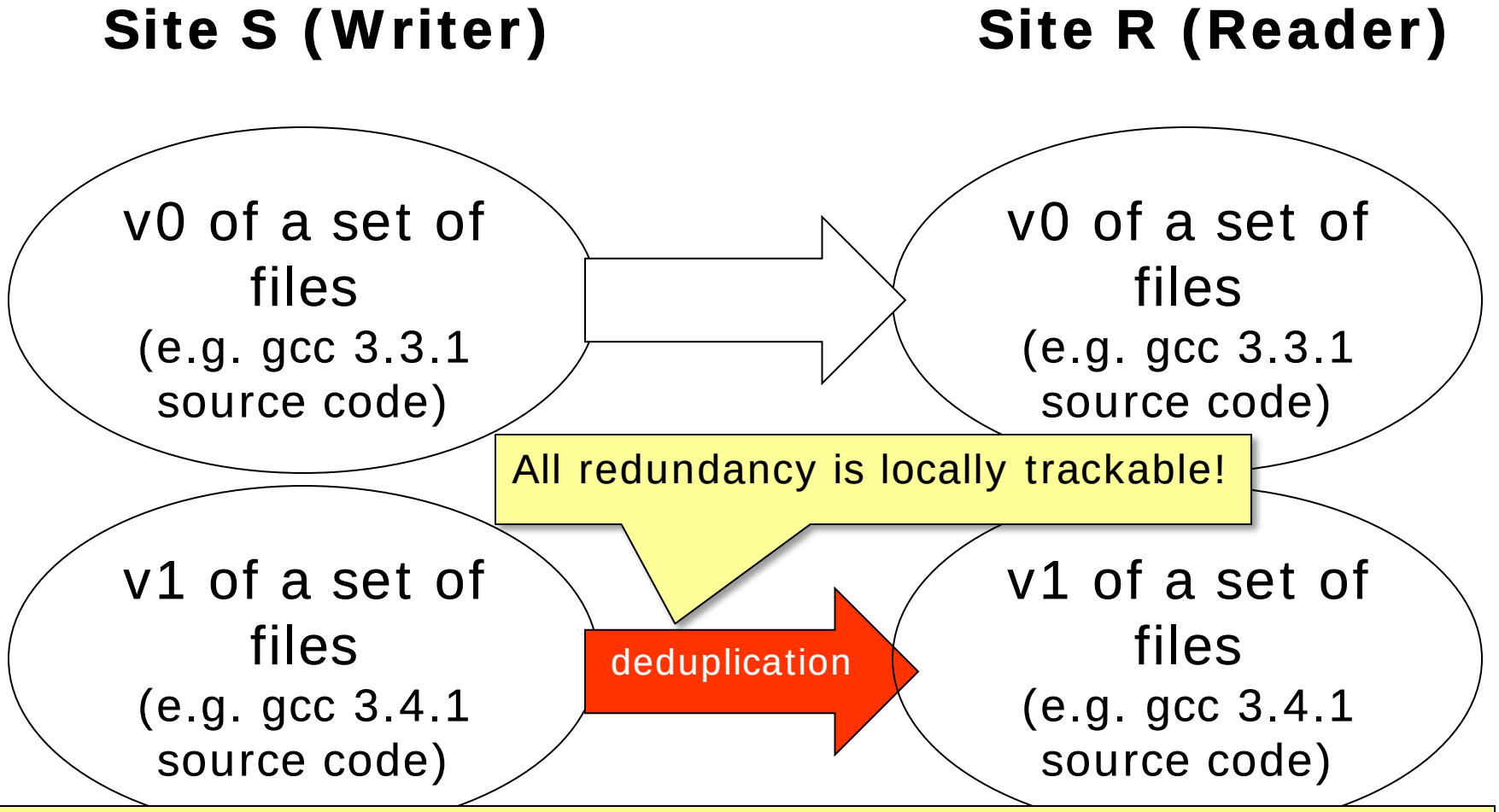
Locally trackable redundancy across pairs of consecutive versions of same object

Any locally trackable redundancy ≈ **Any redundancy**

Evaluation: methodology

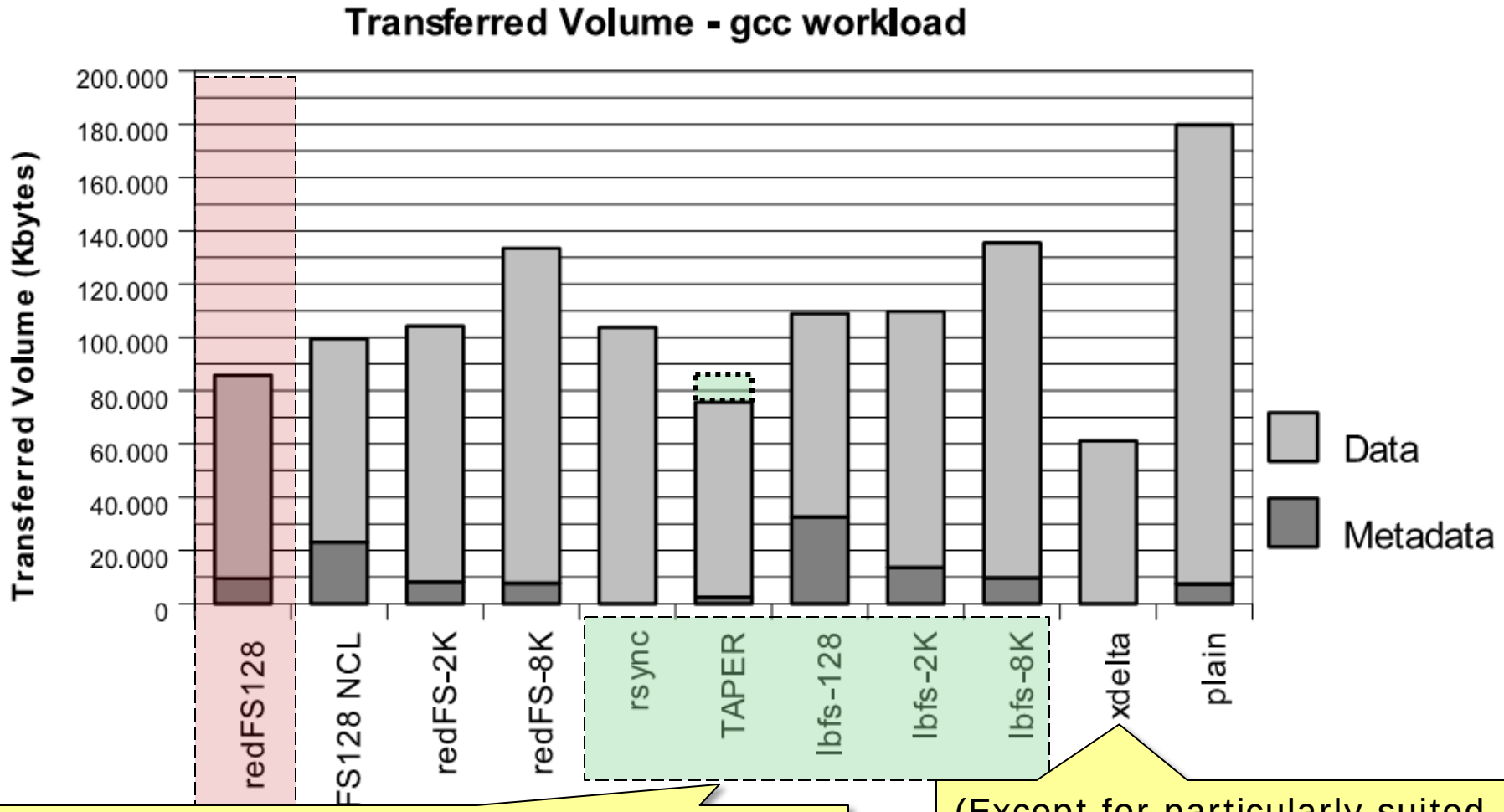
- We evaluated different solutions from every approach:
  - RedFS full implementation
  - LBFS, rsync, TAPER (compare-by-hash)
  - xdelta, svn (delta encoding)
- Two distributed sites, network with different bandwidths (3Mbps to 100Mbps)
- Real workloads
  - Single-writer Scenarios
  - Multi-writer Scenarios

Evaluation: single-writer multi-reader scenarios



Same methodology and workloads as in recent compare-by-hash papers (e.g. LBFS [SOSP'01], TAPER [FAST'05])

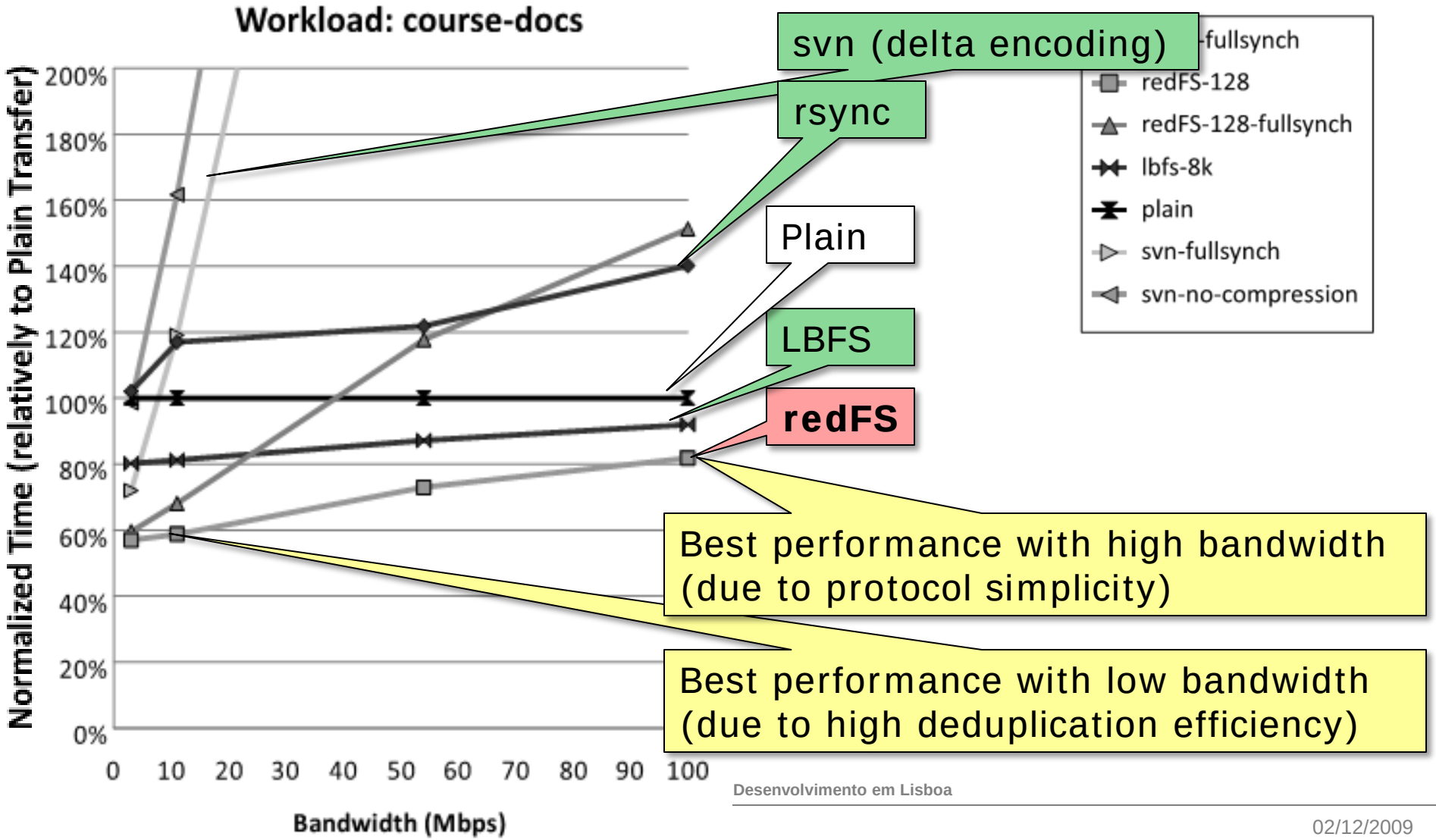
Evaluation: transferred volumes in single-writer multi-reader scenarios



RedFS transfers less (or, in few exceptions, comparable) bytes than all compare-by-hash solutions

(Except for particularly suited workloads such as this one) RedFS transfers less than delta-encoding

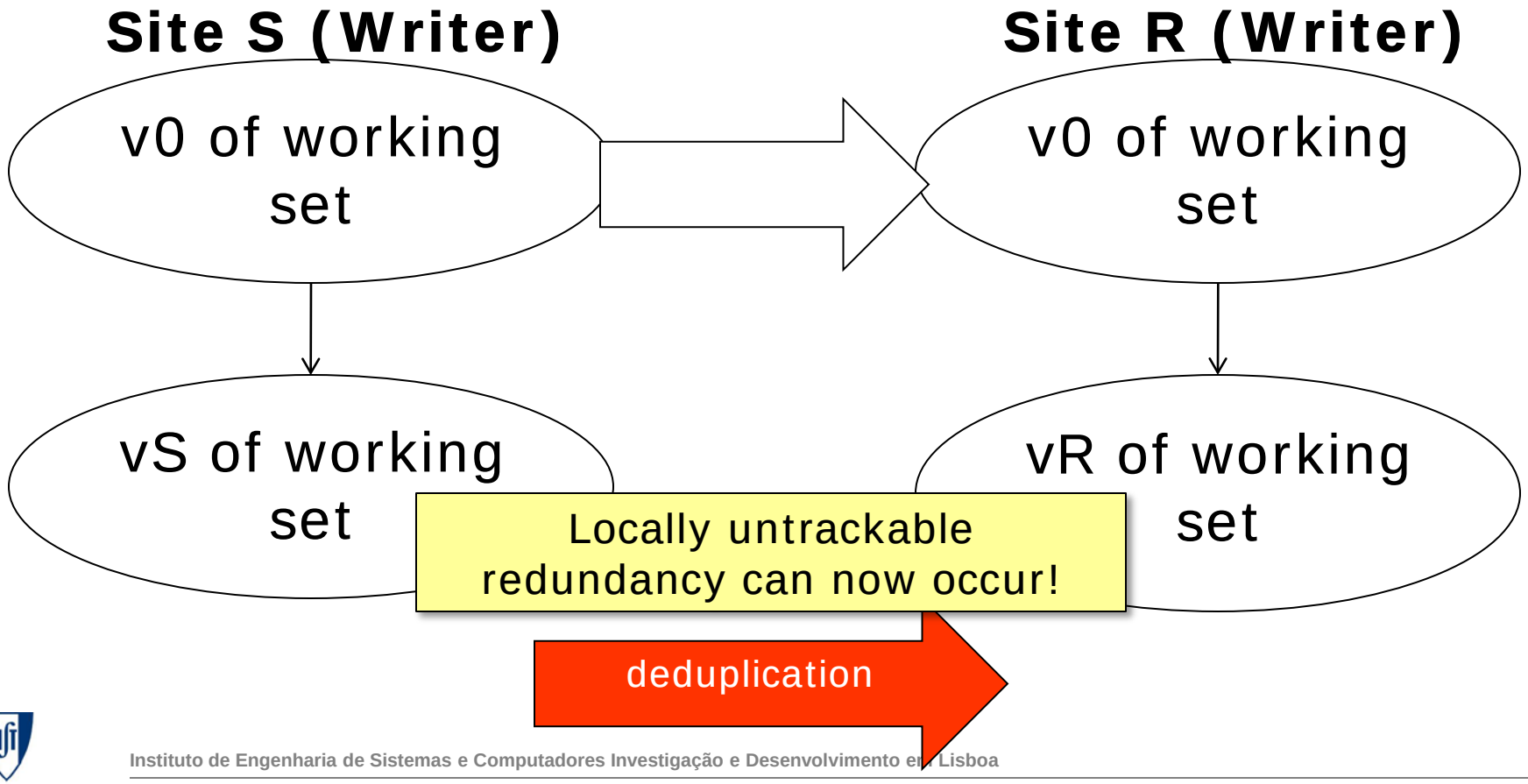
Evaluation: performace in single-writer multi-reader scenarios



Evaluation: multi-writer scenario

technology
from seed

Workloads from collaborative activity between groups of teachers and students during 1-semester courses.



Evaluation: multi-writer scenario

technology
from seed

Q: How much locally untrackable redundancy?

A: Only 1% to 4% of all redundancy generated over +3 months was locally untrackable

Advantages of redFS persist even in real scenarios where locally untrackable redundancy can occur (both in transferred volume and performance)

Conclusions

- Compare-by-hash trades simplicity and efficiency for the ability to detect *any* redundancy
- In relevant scenarios, most redundancy is locally trackable
  - Including the same scenarios that most compare-by-hash papers use to motivate their works
- By exclusively detecting locally trackable redundancy we can outperform compare-by-hash
  - One forgotten example: Delta Encoding
- Our contribution: RedFS's algorithm
  - Fully implemented as open source distributed file system
  - Evaluation shows it outperforms both compare-by-hash and delta encoding

Thank you.

Questions?

www.gsd.inesc-id.pt/~jpbarreto

technology
from seed

inesc id
lisboa

Ok, but at which storage cost?