Multi-Version Coding in Distributed Storage

Zhiying Wang, Viveck Cadambe

April 28, 2014

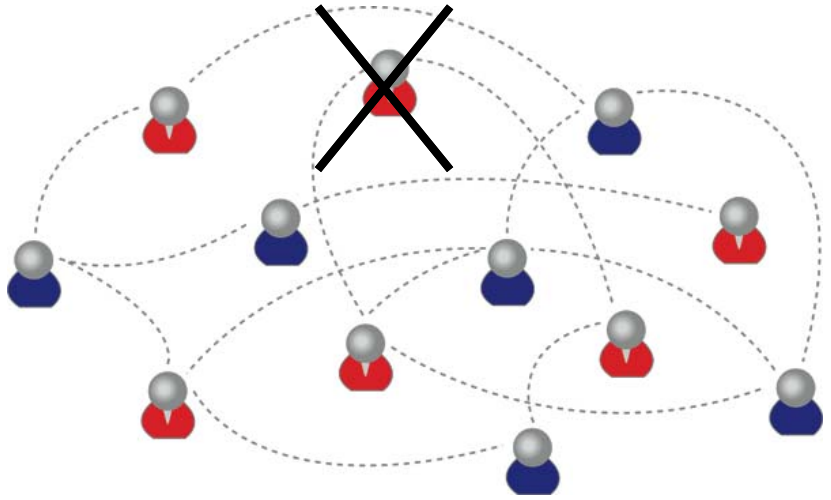
MIT

Center for Science of Information
NSF Science and Technology Center

Coding in Distributed Storage

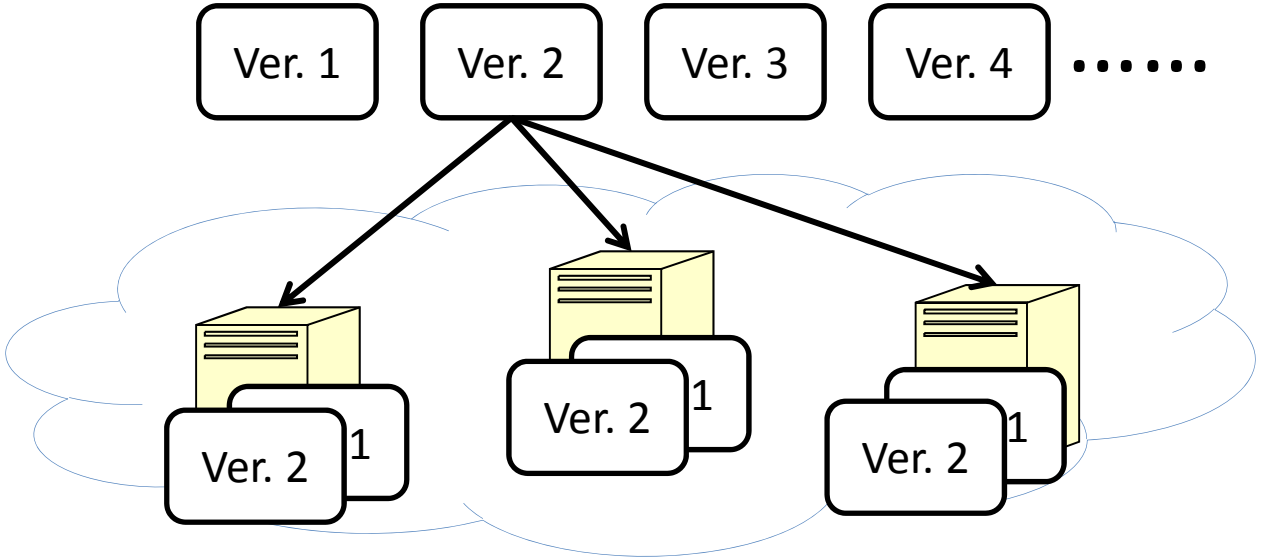
Coding in Distributed Storage

- Error-correction ability
- Redundancy
- Read/write, update, reconstruct
- Repair [Dimakis et al., 2007]
- …
- Assumptions:
  - One data set
  - Info available everywhere

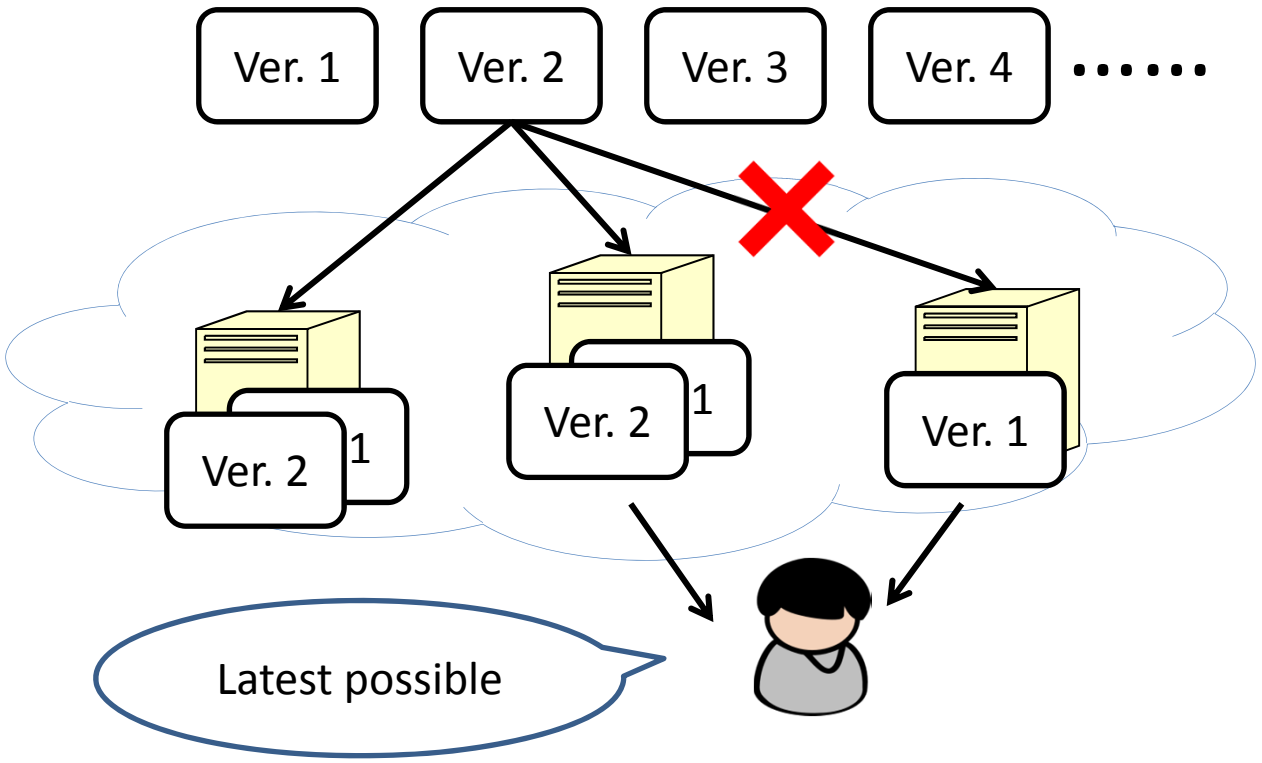


Data of Multiple Versions

- Later versions are more interesting

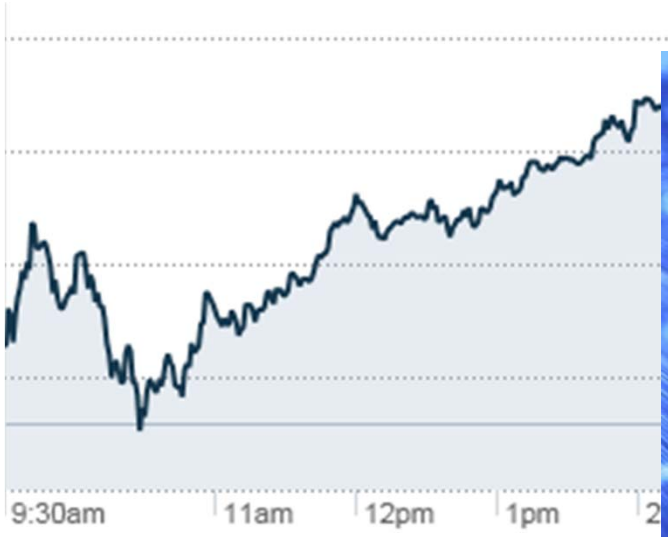


Snapshot in the Real World



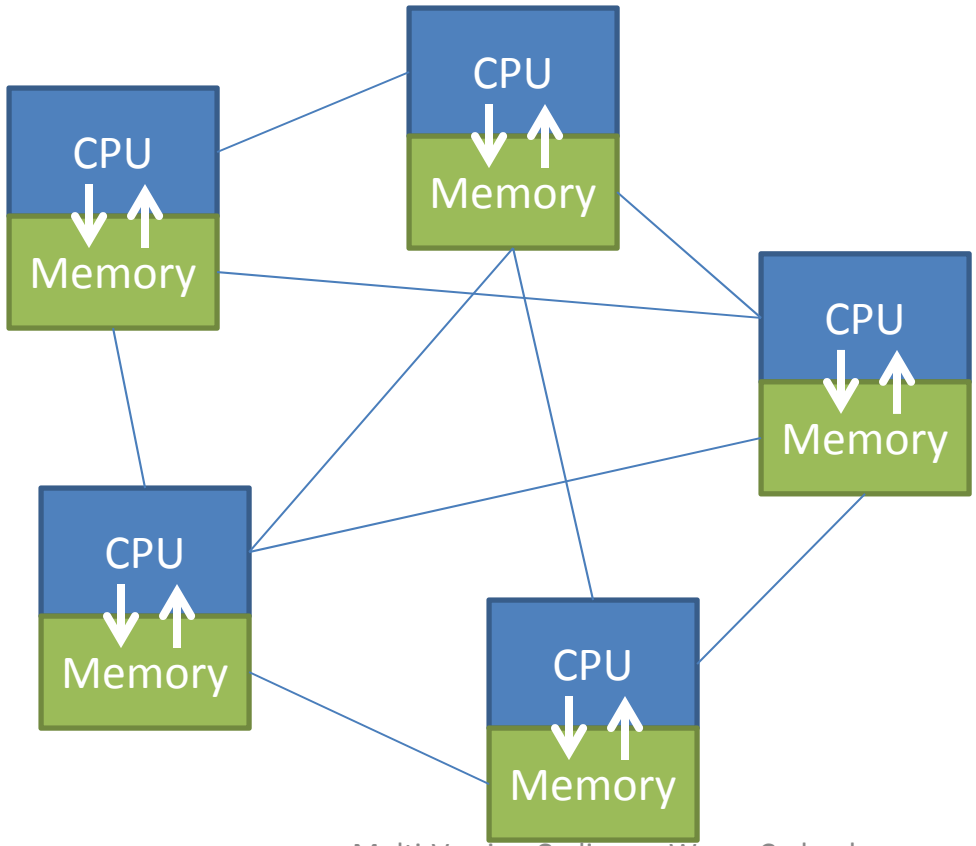
Why Interesting

- Information storage

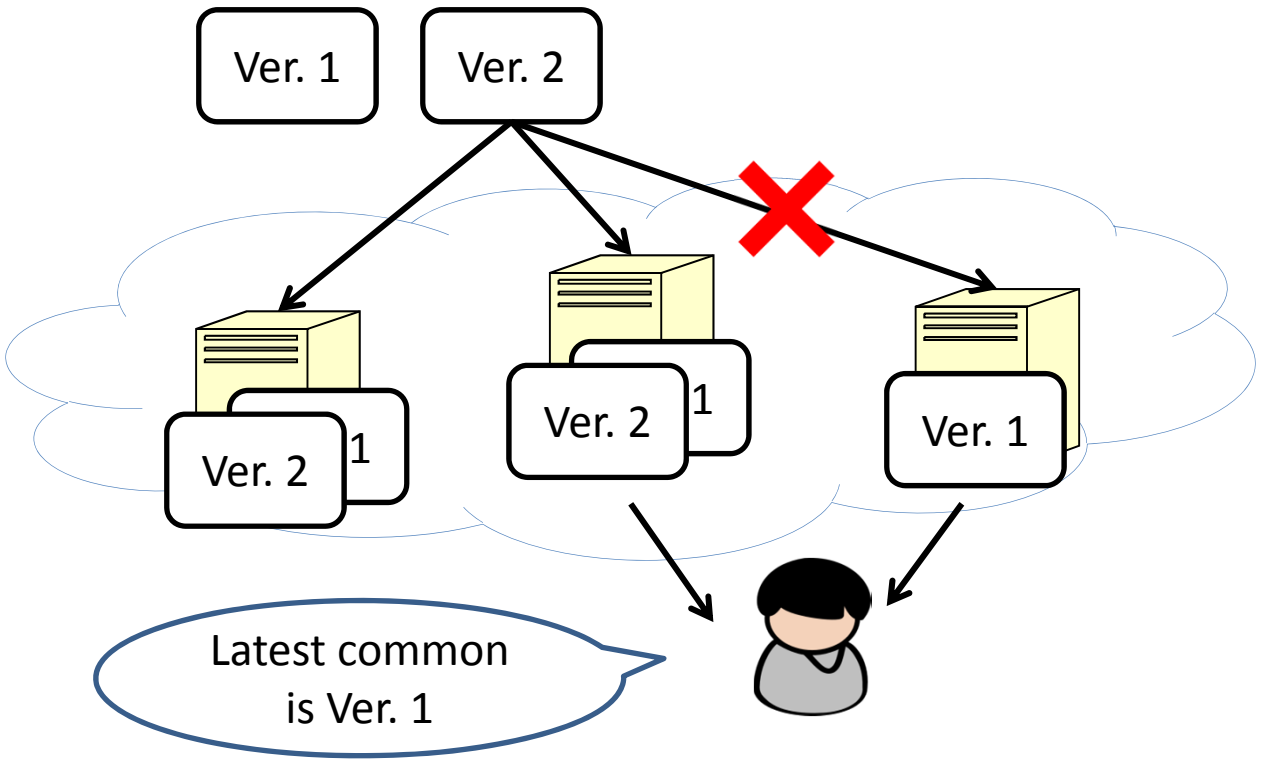


Why Interesting

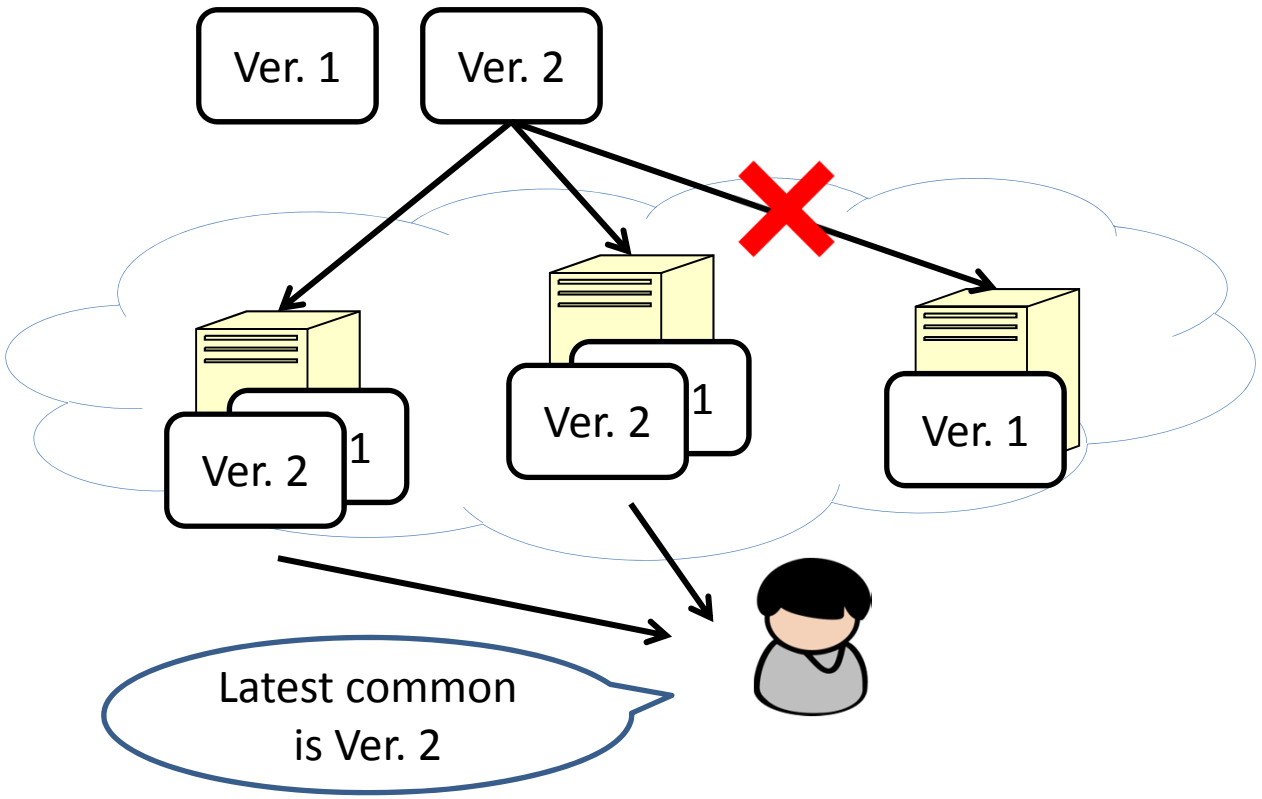
- Multiprocessor programming



Our Problem: Multi-Version Coding



Our Problem: Multi-Version Coding

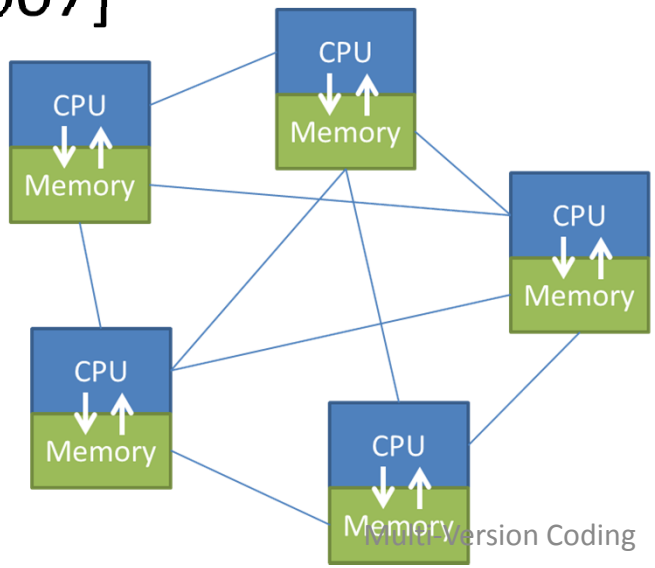


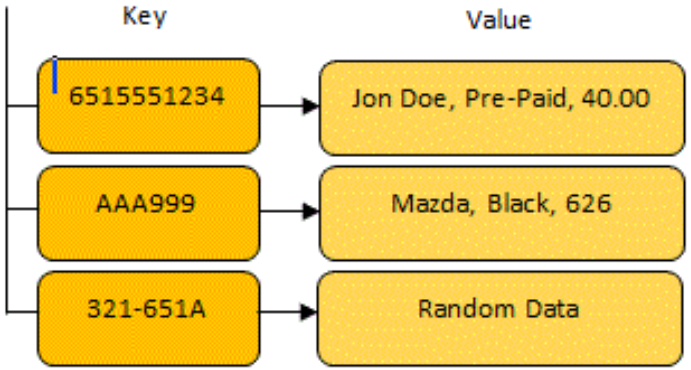
Features for Distributed Systems

- Consistency:
  - Decode the latest possible version
- Asynchrony
  - Every server receives different subsets of versions
  - Every server has no information about others

Connection to Distributed Computing

- Distributed computing theory
  - Shared memory system [Attiya et al., 1995]
- Practical distributed storage system
  - Amazon Dynamo key value store [DeCandia et al., 2007]



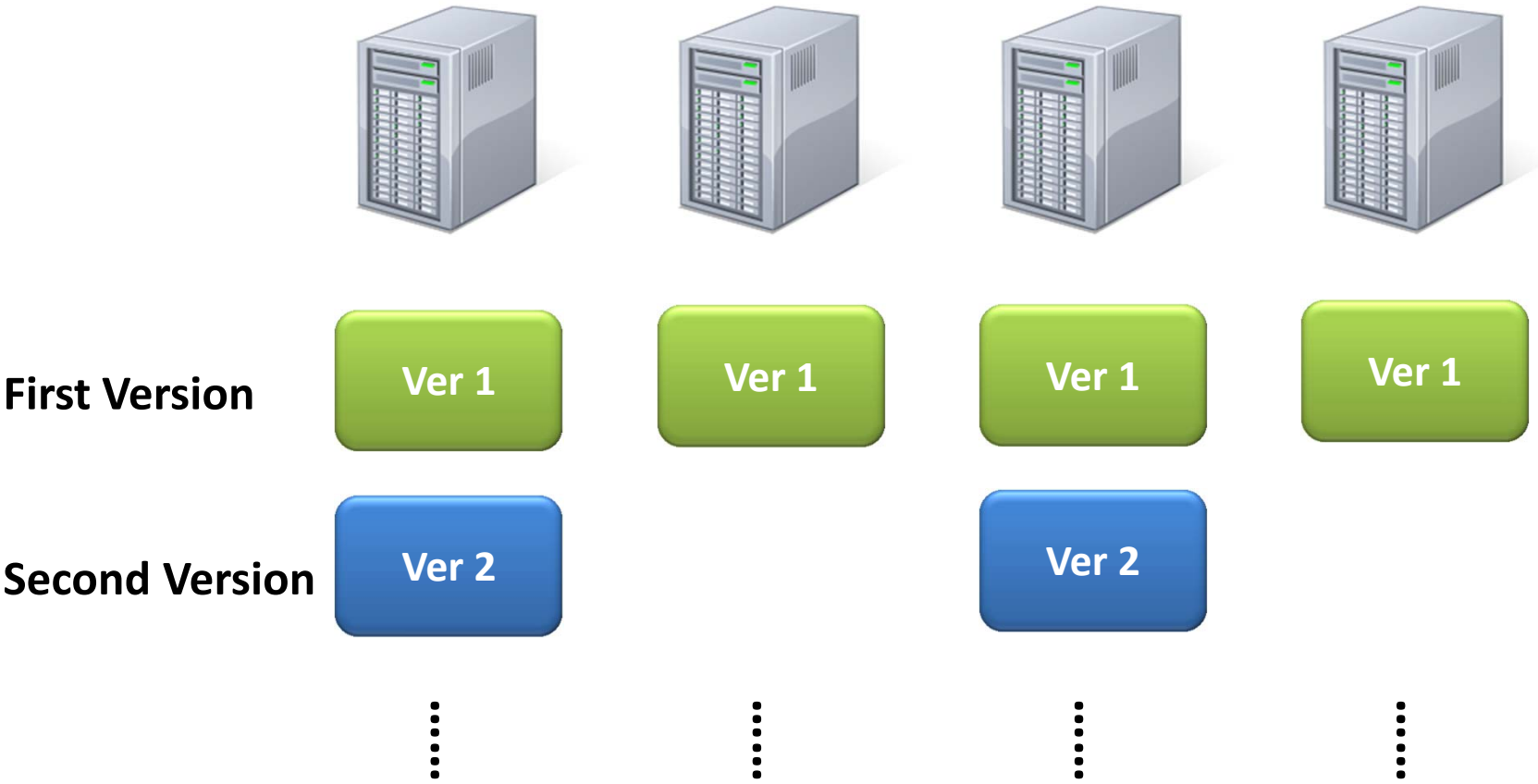


Problem Settings

- *n* servers
- *v* independent versions of the message
- *c* servers connected to a client
- Goal: recover the latest common version
- Q: worst-case storage size of each server?
  - Code construction
  - Bound

Replication

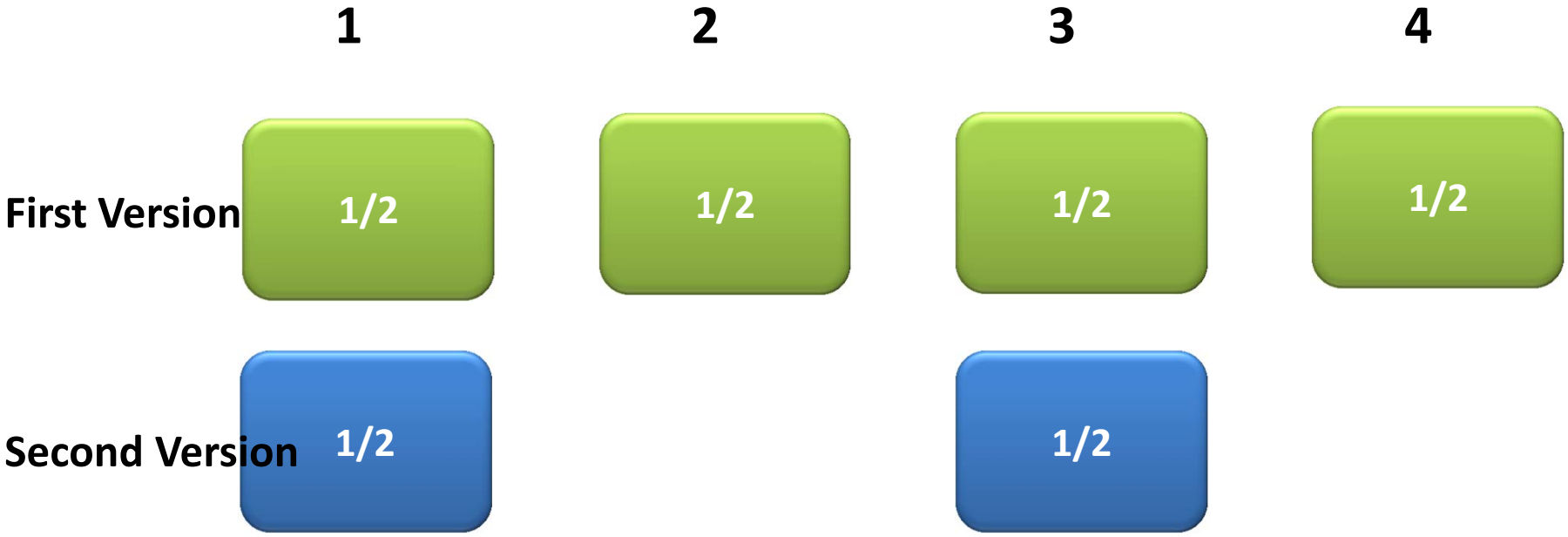
- Storage size = message size



[Attiya et al., 1995], [DeCandia et al., 2007]

(4,2) MDS Codes

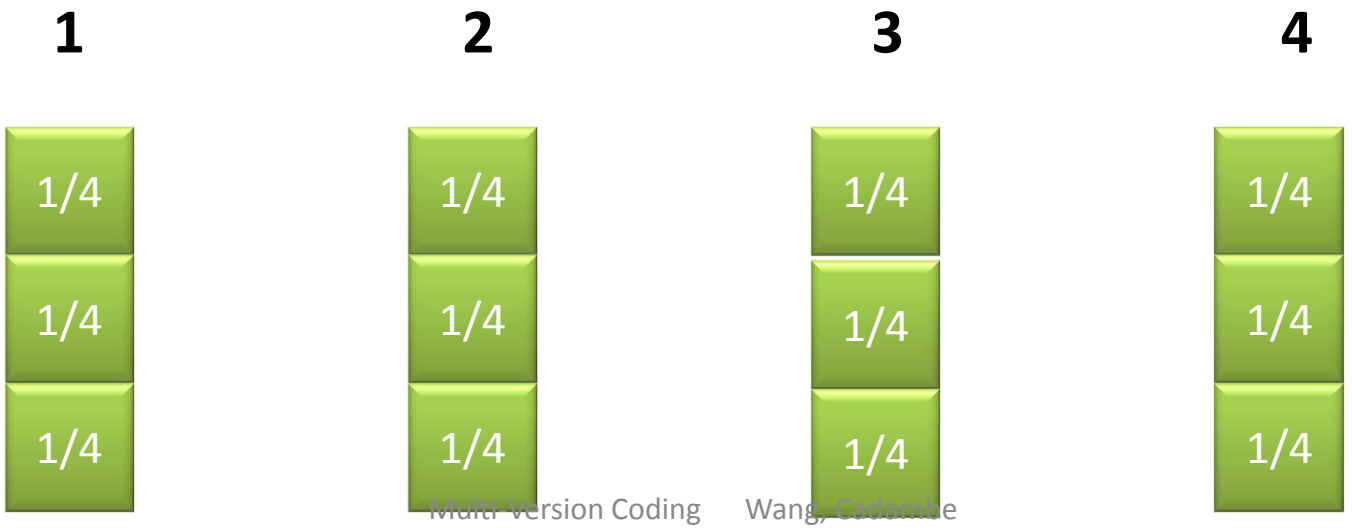
- Decode “latest common” from any 2 servers
- (Worst-cast) storage size = message size



[Hendricks et al., 2007], [Dutta et al., 2008], [Cadambe et al, 2013], [Dobre et al., 2013]

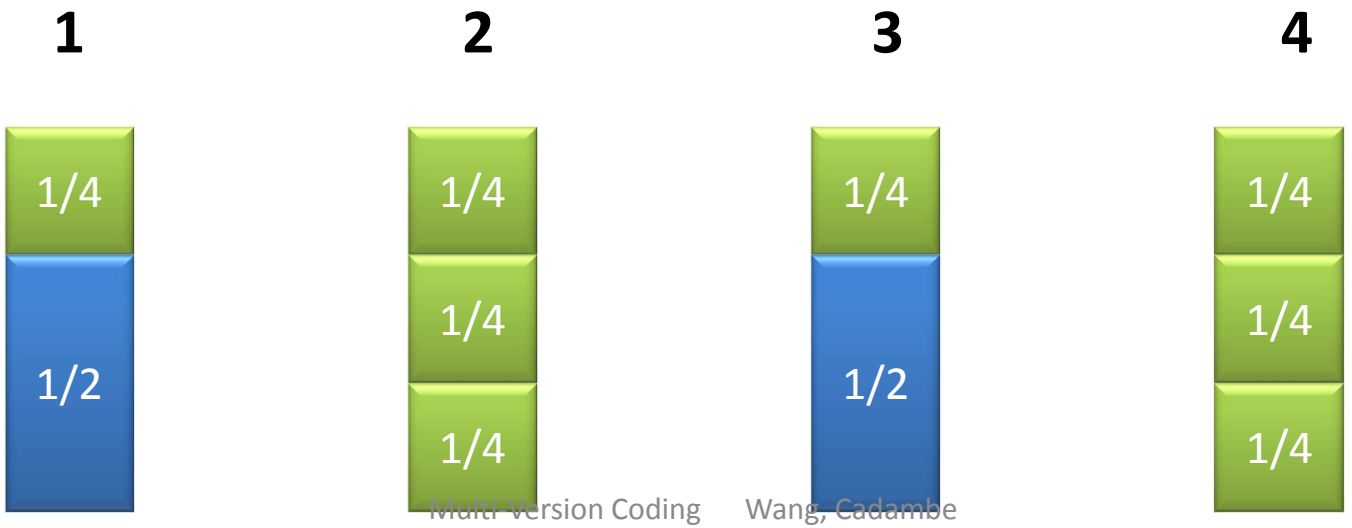
Our Code

- (9,4) code for Version 1



Our Code

- (9,4) code for Version 1
- (4,2) code for Version 2
- Decode “latest common” from any 2 servers
- Storage size = (message size)*3/4



Construction

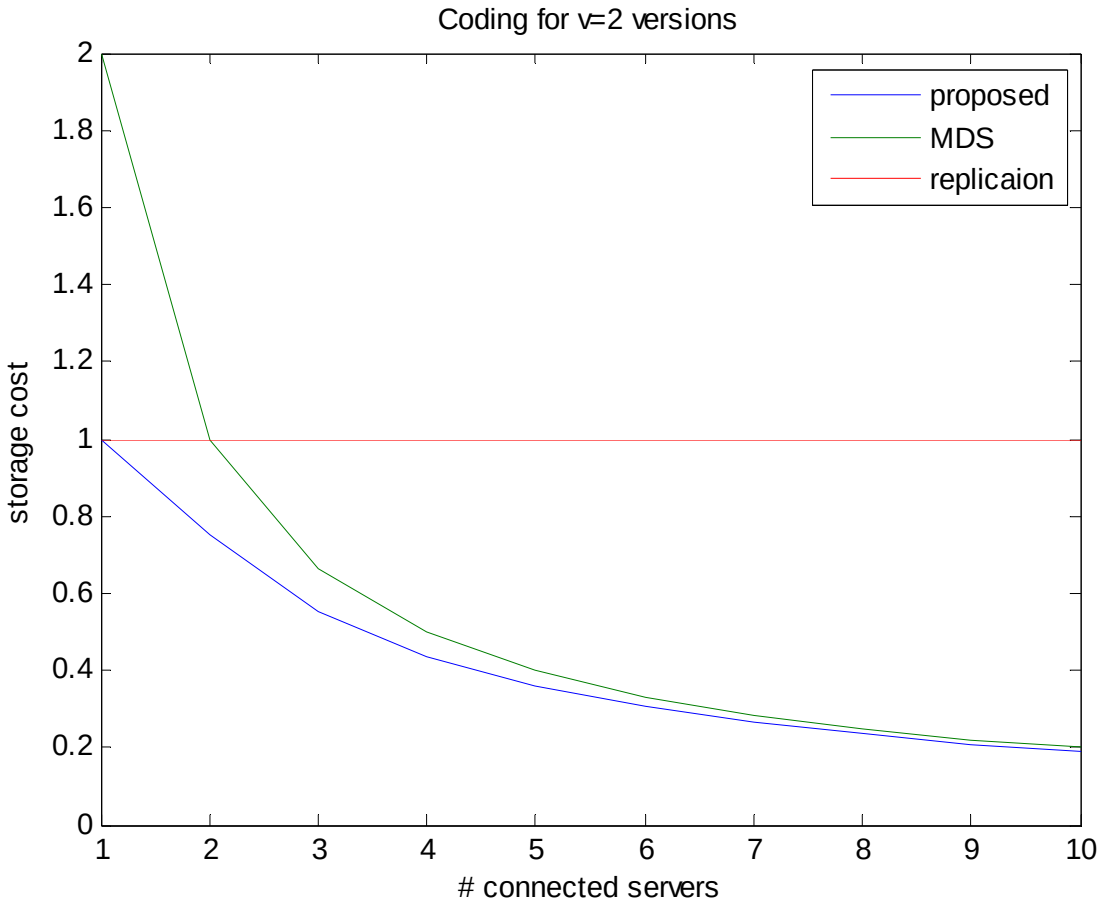
- Encode every version separately
- Assign different storage size for different versions
- Use MDS codes of different rates

- Find optimal assignment by linear programming
- Upon arrival of a new version, over-write

Construction

- Theorem: there exist a code for v versions and c connected servers such that
  - $\frac{\text{Storage size}}{\text{Message size}} = \frac{vc-v+1}{c^2}$
- Comparison
  - MDS: $\frac{v}{c}$
  - Replication: 1

Comparison


Coding for v=2 versions
proposed
MDS
replicaion
storage cost
connected servers
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
1
2
3
4
5
6
7
8
9
10

Bound

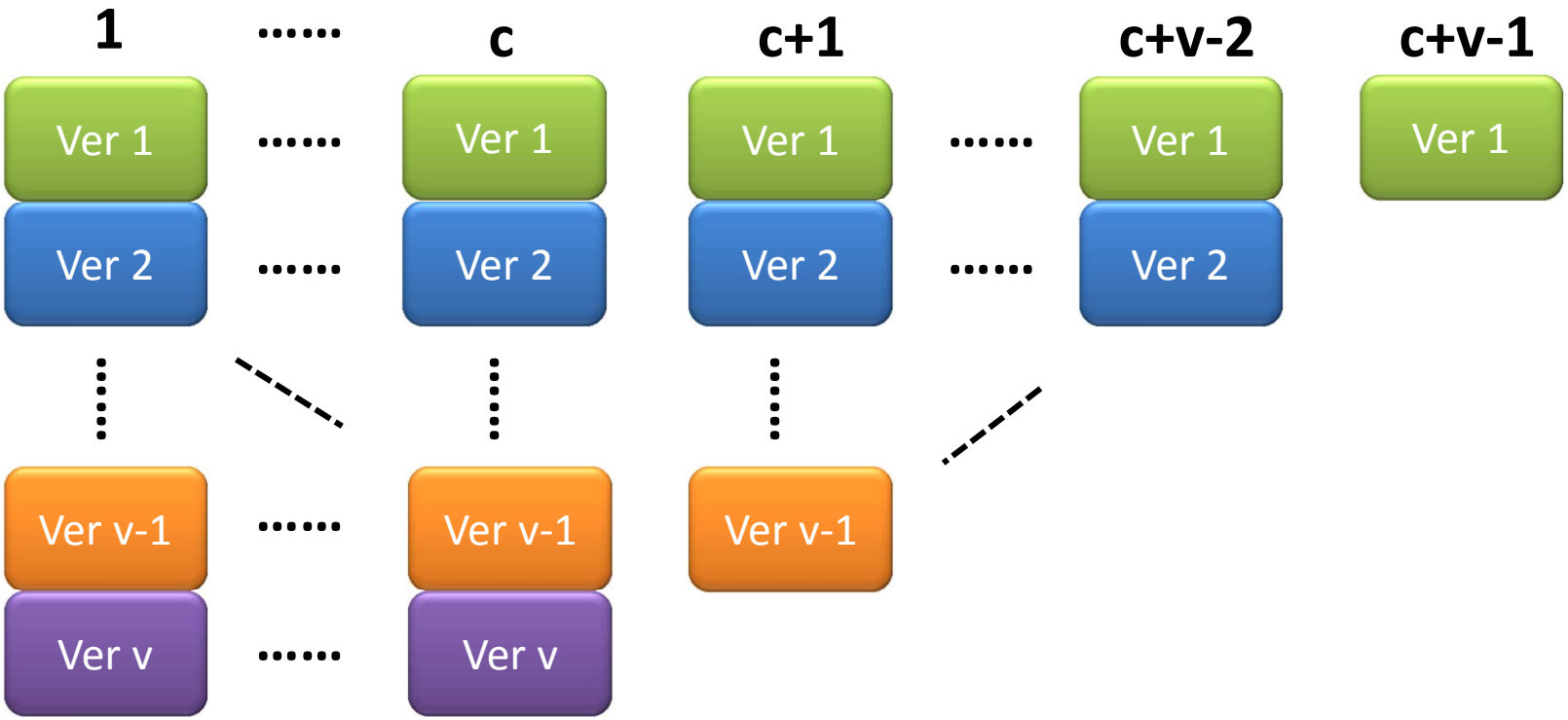
- Theorem: every multi-version code for v versions and c connected servers satisfies

$$\frac{\text{Storage size}}{\text{Message size}} \geq 1 - \left(1 - \frac{1}{c}\right)^{v}$$

- Tight for v=2 versions

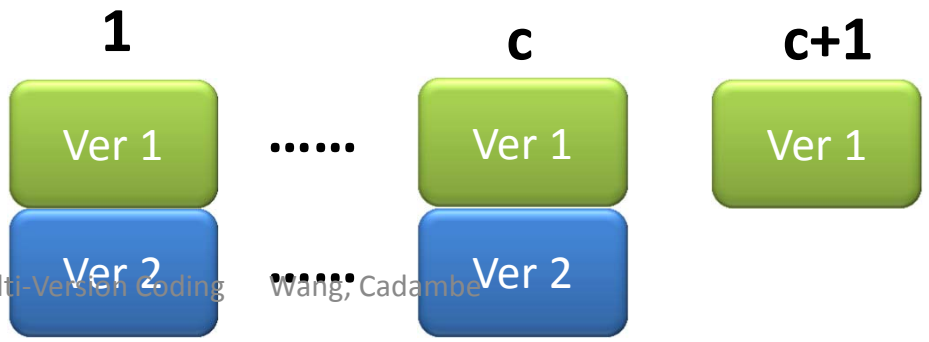
Proof Idea

- Consider a special scenario
- Lower bound the mutual information



Proof for v=2

- X_i information of Server i
- W_j information of Version j
- α server storage size
- B message size

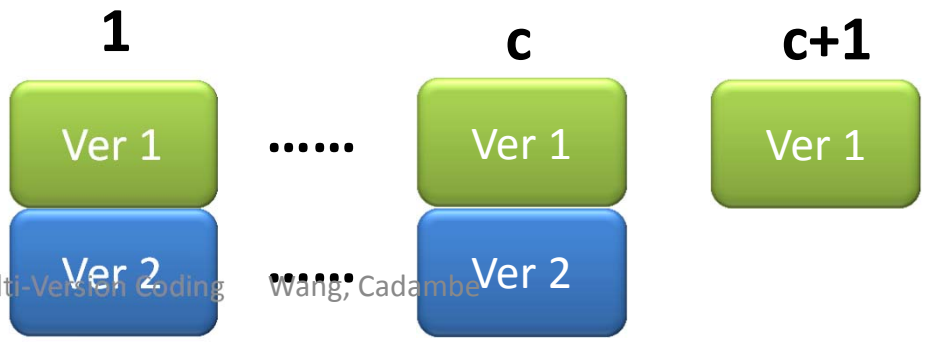


Proof for v=2

- Lemma 1. $I(X_{[c-1]}; W_1) \geq B - \alpha.$
- Lemma 2. W.l.o.g., $I(X_{[c-1]}; W_2|W_1) \geq \frac{c-1}{c} B.$
- Proof of theorem:

$$
\begin{aligned}
&(c-1)\alpha \\
\geq &I(X_{[c-1]}; W_1, W_2) \\
= &I(X_{[c-1]}; W_1) + I(X_{[c-1]}; W_2|W_1) \\
\geq &B - \alpha + \frac{c-1}{c} B.
\end{aligned}
$$

X_i information of Server i
W_j information of Version j
α server storage size
B message size



Future Directions

- Tight bound
- Dependent versions
- Average-case storage size
- Connect coding in distributed storage & distributed algorithms

Center for
Science of Information
NSF Science and Technology Center

THANK YOU!