Cabinet: Dynamically Weighted Consensus Made Fast

Gengrui (Edward) Zhang, Shiquan Zhang,
Michail Bachras, Hans-Arno Jacobsen

Assistant Professor

Concordia University

WELCOME TO LONDON
VLDB 2025

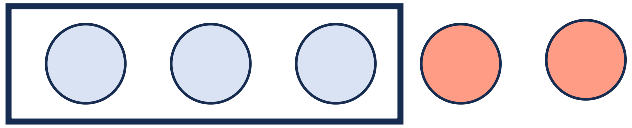
UNIVERSITÉ Concordia UNIVERSITY

UNIVERSITY OF TORONTO

Consensus with majority quorums

- Consensus algorithms reach agreement on committing values among all servers even if some servers fail
- How to reach consensus?
  - Majority quorums!
  - Every decision must be endorsed by a majority
    - Paxos, Raft
  - Tolerate $f = \frac{\lfloor n-1 \rfloor}{2}$ failures

5 nodes, 3 is a majority



When $n = 5$,
- Quorum size $= 3$
- Tolerate 2 failures

Majority quorums may become inefficient in modern computing applications

Features of modern computing

1. System scales continue to grow; e.g., distributed databases and blockchain applications (Hyperledger Fabric)
2. Systems are becoming increasingly heterogeneous
   - Strong nodes
   - Weak nodes

Strong nodes often compute, store, and respond faster then weak nodes

Strong nodes

Weak nodes

Under large-scale, especially heterogeneous clusters

Majority quorums may become inefficient because of the quorum size required by each round

When $n = 100$, quorum size $= 51$, tolerating 49 failures

Strong nodes are compelled to wait for weak nodes, resulting in low throughput and high latency!

The Cabinet

Established in Article II, Section 2 of the Constitution, the Cabinet's role is to advise the President on any subject he may require relating to the duties of each member's respective office.

Cabinet:

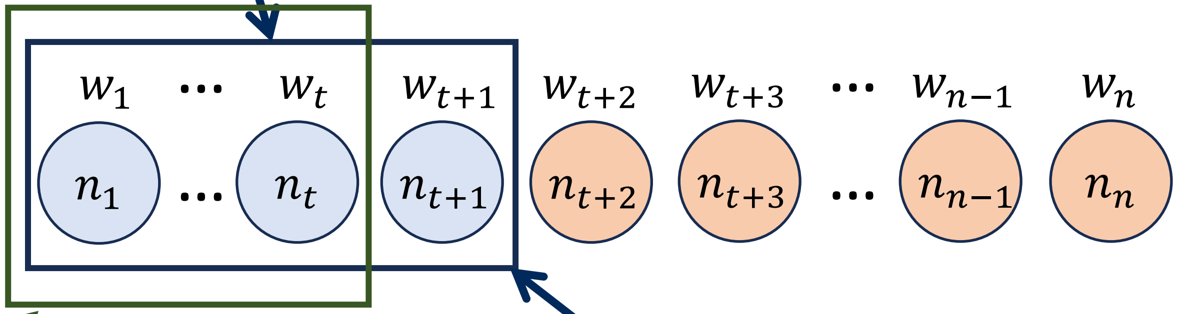
Dynamically Weighted Consensus Made Fast

One-size-fits-all weight scheme

- A configurable failure threshold, t $\left(1 \leq t \leq \left\lfloor \frac{n-1}{2} \right\rfloor\right)$
  - Tolerate at least t failures with a quorum size of $t + 1$

Cabinet members: top $t + 1$ highest weighted nodes

$$CT = \sum_{i=0}^{n} \frac{w_i}{2}$$



Invariant 1: $\sum_{i=1 \to t+1} w_i > CT$

Invariant 2: $\sum_{i=1 \to t} w_i < CT$

Safety
No two correct nodes decide differently

Liveness
Nodes eventually decide

Fast Agreement
System wide agreement can be made as soon as $t + 1$ nodes reach an agreement (e.g., $t = 5$, $n = 100$)

Cabinet's implementation of weight scheme

- Cabinet uses geometric sequences to construct weight schemes

Weights: $w_1 > w_2 > \dots w_{n-2} > w_{n-1} > w_n$

Geometric sequence: $a_1 r^{n-1} > a_1 r^{n-2} \dots a_1 r^2 > a_1 r > a_1$

$$CT = \sum_{i=0}^{n} \frac{w_i}{2}$$

Simple and effective!

Have your own weight scheme with n and t
(n: # of nodes; t: # of failures)

Invariant 2: $\sum_{i=1 \to t} w_i < CT$

Invariant 1: $\sum_{i=1 \to t+1} w_i > CT$

$$\sum_{i=n-t}^{n-1} a_1 r^i < CT = \frac{1}{2} \sum_{i=0}^{n-1} a_1 r^i < \sum_{i=n-t-1}^{n-1} a_1 r^i$$

$$\boxed{r^{n-t-1} < \frac{1}{2}(r^n + 1) < r^{n-t}}$$

Example of Cabinet's weight schemes

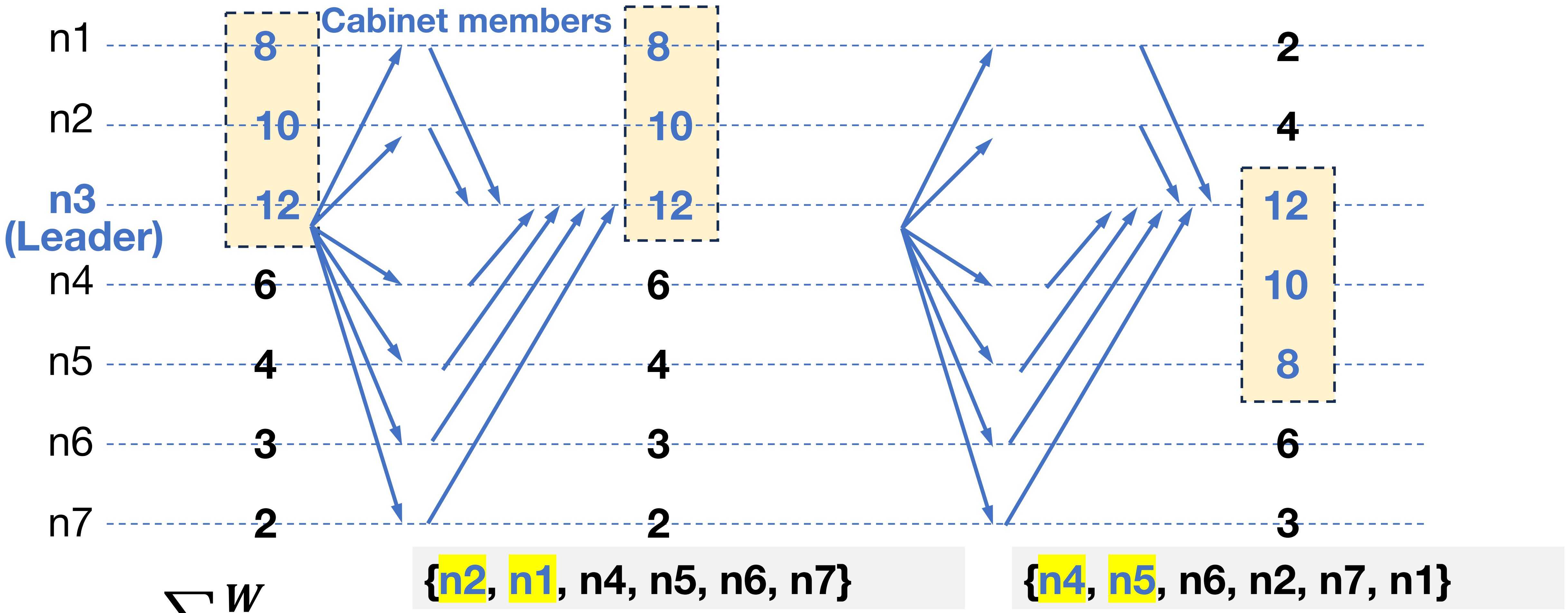
# of failures tolerated

Cabinet members

t	r	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
1	1.40	**20.7**	**14.8**	10.5	7.5	5.4	3.8	2.7	2.0	1.4	1
2	1.38	**18.2**	**13.2**	**9.5**	6.9	5.0	3.6	2.6	1.9	1.4	1
3	1.19	**4.8**	**4.0**	**3.4**	**2.8**	2.4	2.0	1.7	1.4	1.2	1
4	1.08	**2.0**	**1.9**	**1.7**	**1.6**	**1.5**	1.4	1.3	1.2	1.1	1

Cabinet weight schemes with different customized failure thresholds ($t = 1, 2, 3, 4$) in a $n = 10$ system

Dynamic weight assignment (t=2)



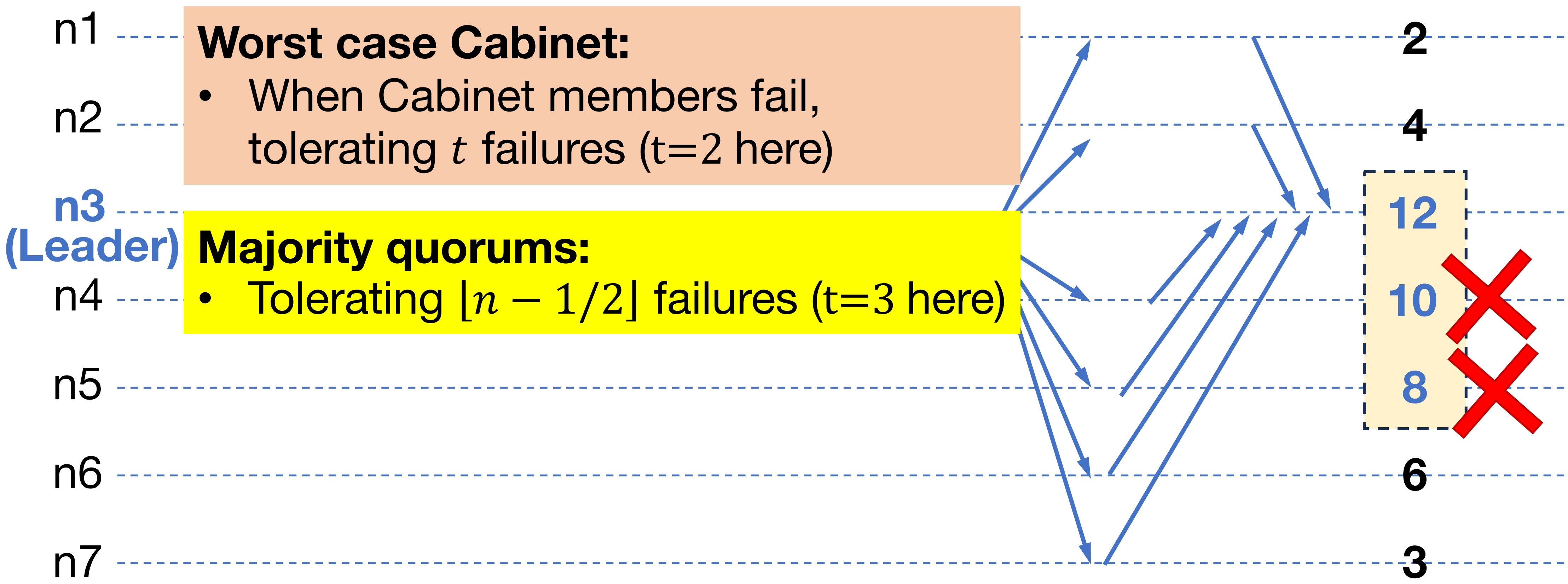
$$CT = \sum \frac{W}{2} = 22.5$$

The first $t + 1$ replying nodes in round r become the cabinet members in round $r + 1$

Feature 1: Tolerating more than t failures

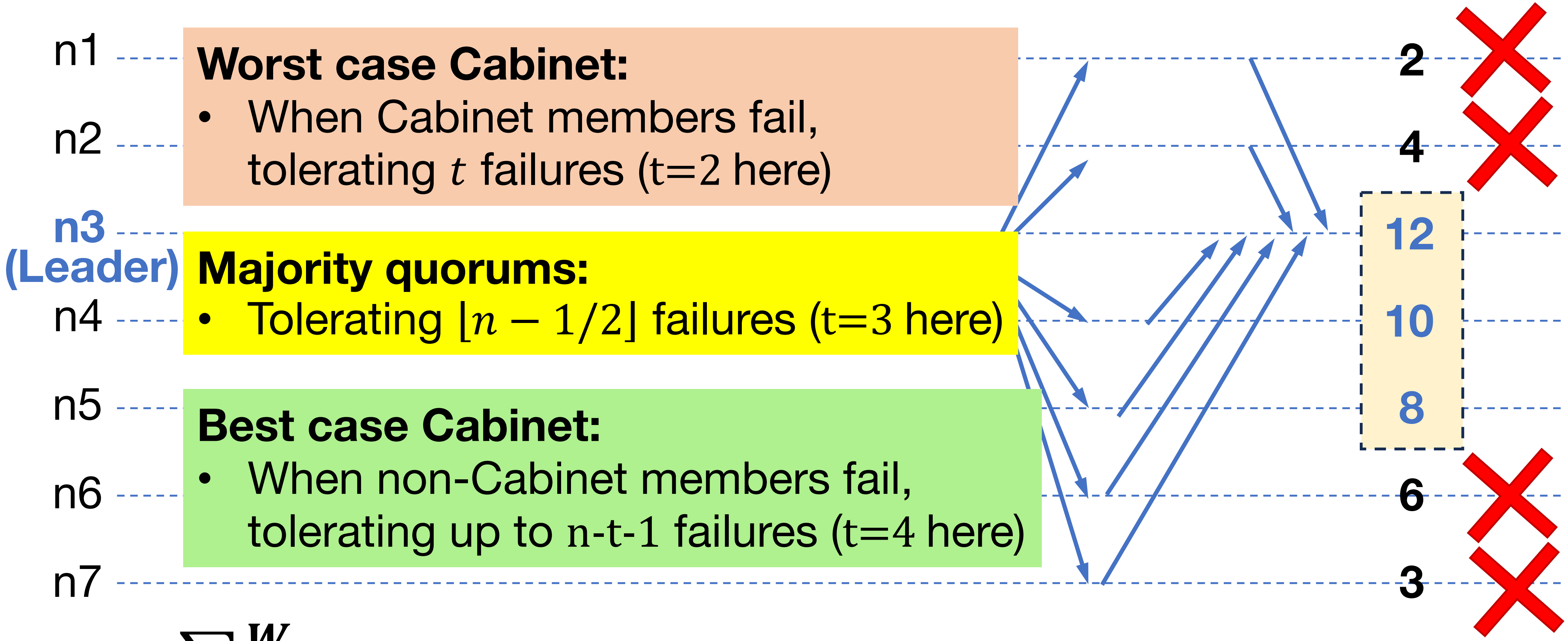
Worst case Cabinet:
- When Cabinet members fail, tolerating t failures (t=2 here)

Majority quorums:
- Tolerating $\lfloor n - 1/2 \rfloor$ failures (t=3 here)



$$CT = \sum \frac{W}{2} = 22.5$$

Feature 1: Tolerating more than t failures



Worst case Cabinet:
- When Cabinet members fail, tolerating t failures (t=2 here)

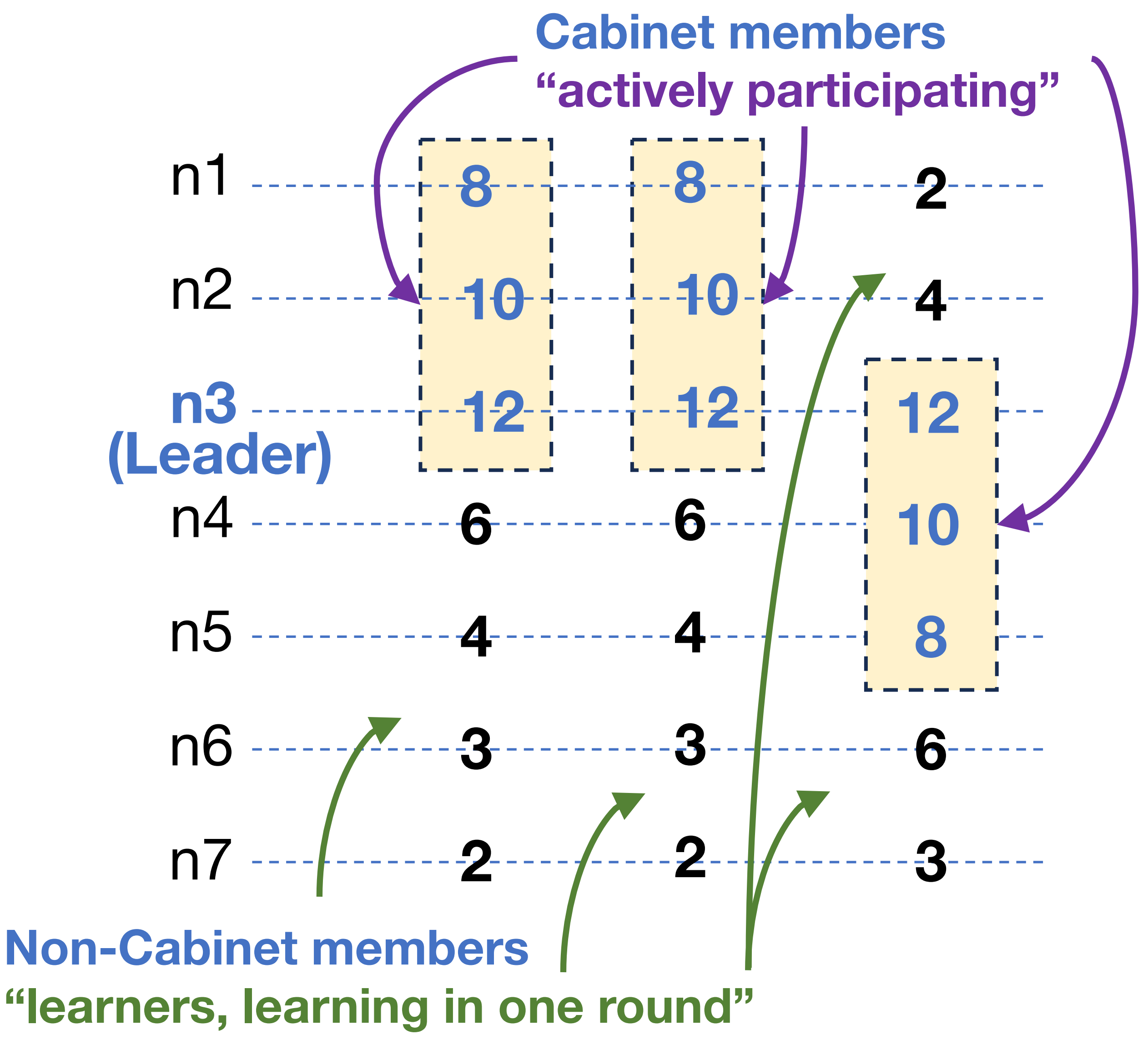
Majority quorums:
- Tolerating $\lfloor n - 1/2 \rfloor$ failures (t=3 here)

Best case Cabinet:
- When non-Cabinet members fail, tolerating up to n-t-1 failures (t=4 here)

$$\boldsymbol{CT = \sum \frac{W}{2} = 22.5}$$

Feature 2: Avoid manual role selections

- Under large replication deployment, Cabinet does not need to manually partition nodes to "acceptors/followers" or "learners"
- Cabinet members are actively participating nodes – aka **acceptors/followers**
- Non-Cabinet members are **learners**, and still learn the results in one round



Evaluation: cluster setup

- Homogeneous and heterogeneous clusters of sizes of $n = 10, 20, 50, 100$
  - Heterogeneity of CPUs, RAM, and Disk
- Evaluated using YCSB workloads, where each follower runs a MongoDB

Cluster	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5
Homogeneous	$n/5\times$**C3**	$n/5\times$**C3**	$n/5\times$**C3**	$n/5\times$**C3**	$n/5\times$**C3**
Heterogeneous	$n/5\times$**C1**	$n/5\times$**C2**	$n/5\times$**C3**	$n/5\times$**C4**	$n/5\times$**C5**

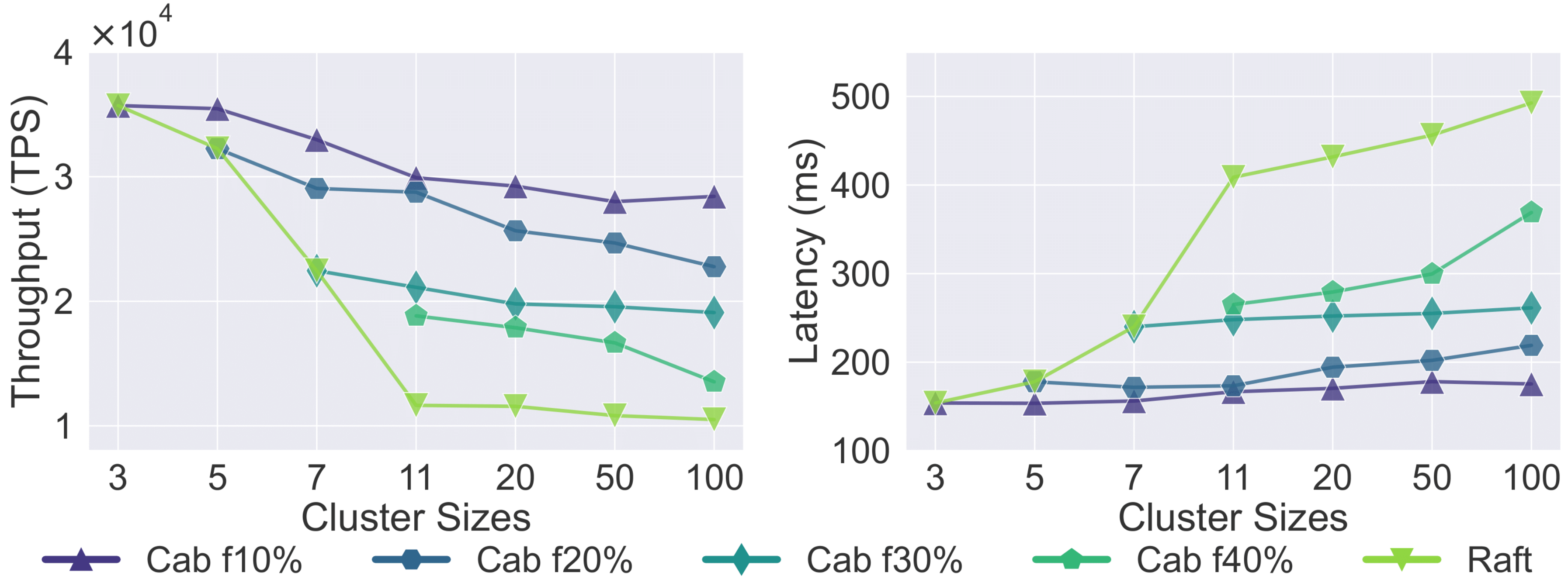
C1 1c-7.5gb-56
C2 2c-15gb-92
C3 4c-15gb-164
C4 8c-30gb-308
C5 16c-60gb-596

CPU: 2.40 GHz Intel Xeon processors (Skylake)
Operating system: Ubuntu 18.04.1 LTS
TCP/IP bandwidth: $\approx$400 Megabytes/s
Raw network latency: $<$1 ms

1 vCPU, 7.5GB RAM, and 56GB Disk

Homogeneous	$n = 50$	Heterogeneous
10 nodes × C3	Z1	10 nodes × C1 (weakest)
10 nodes × C3	Z2	10 nodes × C2
10 nodes × C3	Z3	10 nodes × C3
10 nodes × C3	Z4	10 nodes × C4
10 nodes × C3	Z5	10 nodes × C5 (strongest)

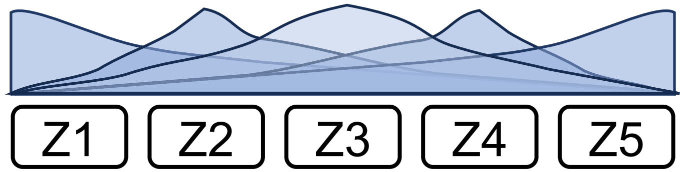
Performance under scaling clusters



Heterogeneous clusters under
YCSB workload A

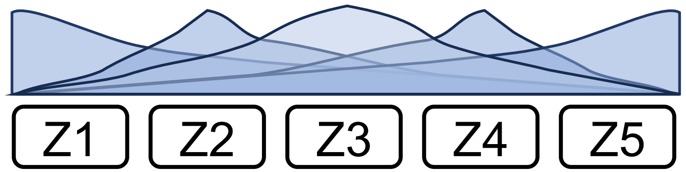
Cabinet’s weighted consensus mechanism **consistently outperforms** Raft’s traditional consensus at all scales

Performance under dynamic network delays



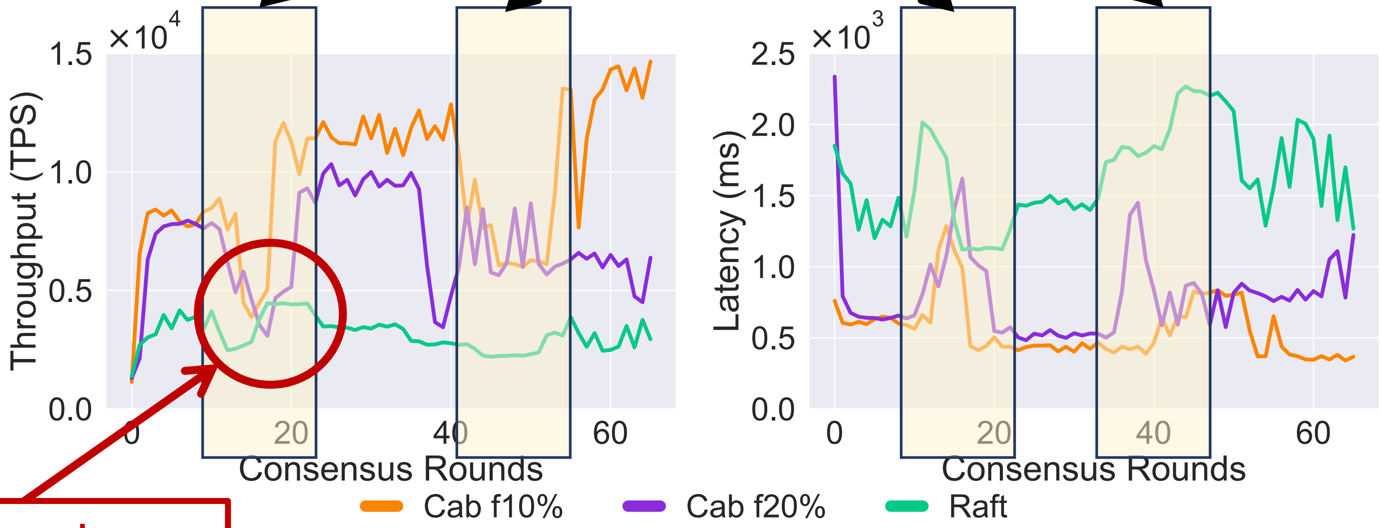
Network delays of $1000 \pm 200ms$ to $100 \pm 20ms$ are dynamically imposed across 5 zones

Performance under dynamic network delays



Network delays of $1000 \pm 200ms$ to $100 \pm 20ms$ are dynamically imposed across 5 zones

Weight reassignment promptly reassigns high weights to currently faster nodes.
Cabinet stays at optimal performance!



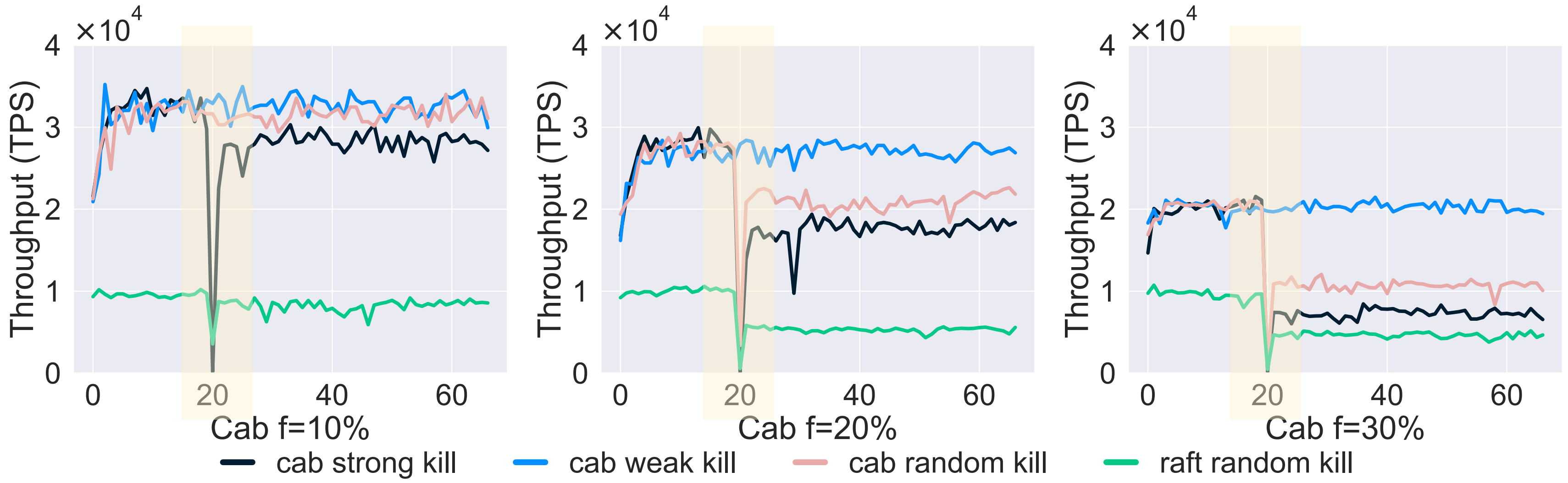
Strong nodes experience high network delays

$n = 50$ heterogeneous cluster under YCSB workload A

Performance under crash failures

At round 20, we crashed x nodes

- **Strong kills** crash x top highest-weight nodes
- **Weak kills** crash x bottom lowest-weight nodes
- **Random kills** randomly crash x nodes



Cabinet outperforms Raft under all failure strategies

Conclusions

- Cabinet is the first **dynamically weighted** consensus algorithm
  - Achieving fast agreements with a quorum size of t+1
  - Tolerating at least ***t* failures at least** and ***n*-*t*-1 failures at most**
- Cabinet offers **a new tradeoff frontier** between **performance** and **fault tolerance**
  - Gains higher performance by relaxing ***absolute*** *fault-tolerance guarantees* (in practice, it can often tolerate more failures)
  - Adds only **two integers** into Raft's RPCs, making integration into existing Raft systems straightforward

Thank you!

Code:

Website: