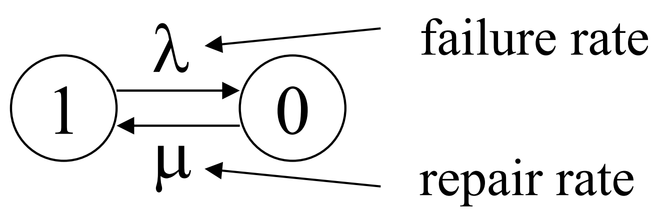
Case Study 1: Replicated File Management

Source: S. Jajodia & D. Mutchler,
"Dynamic voting algorithms for maintaining the consistency of a replicated database"
ACM Trans, Database Systems, Vol. 15, No. 2,
June 1990, pp. 230-280.

one copy: $\leftarrow$ if failed, then it is not accessible

Availability

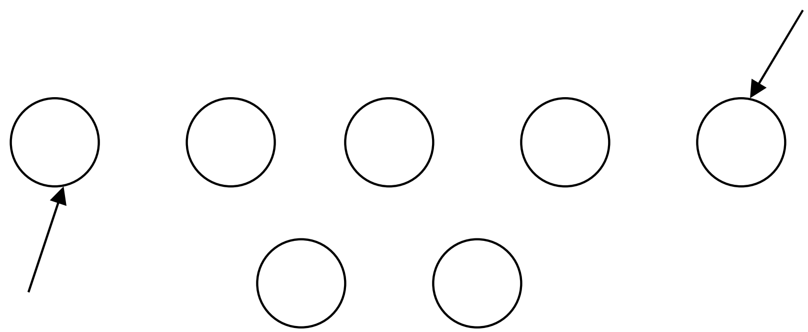


$$A(\infty) \equiv \text{Availability} = \frac{\mu}{\lambda + \mu}$$

$$\left[\begin{aligned} A(t) \equiv P_1(t) &= \frac{\mu}{\mu + \lambda} + \frac{\lambda}{\lambda + \mu} e^{-(\mu+\lambda)t} \\ 1 - A(t) \equiv P_0(t) &= \frac{\lambda}{\mu + \lambda} - \frac{\lambda}{\lambda + \mu} e^{-(\mu+\lambda)t} \end{aligned}\right]$$

Can we use replicated copies to improve availability?

consider <u>only the update operations</u>: suppose we have 7 copies



Cannot update just one copy and leave the others unchanged

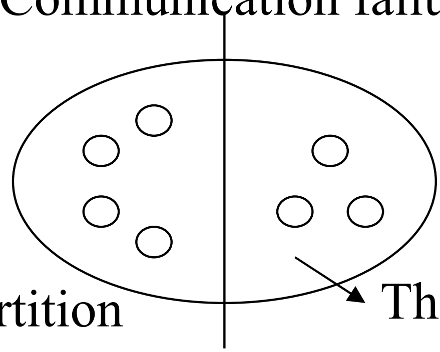
→ will create inconsistency problems

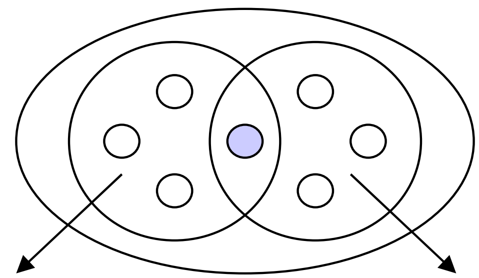
Must maintain <u>one-copy</u> illusion to the user

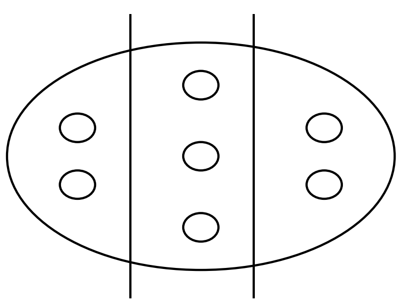
Consistency algorithms for replicated data:

Static: n copies

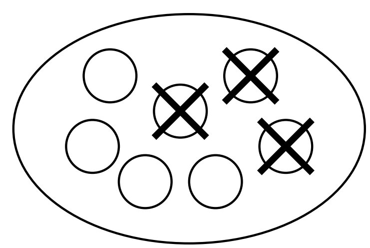
(simple voting) *can do update if a majority of n copies can be reached & updated







No partition can do any update



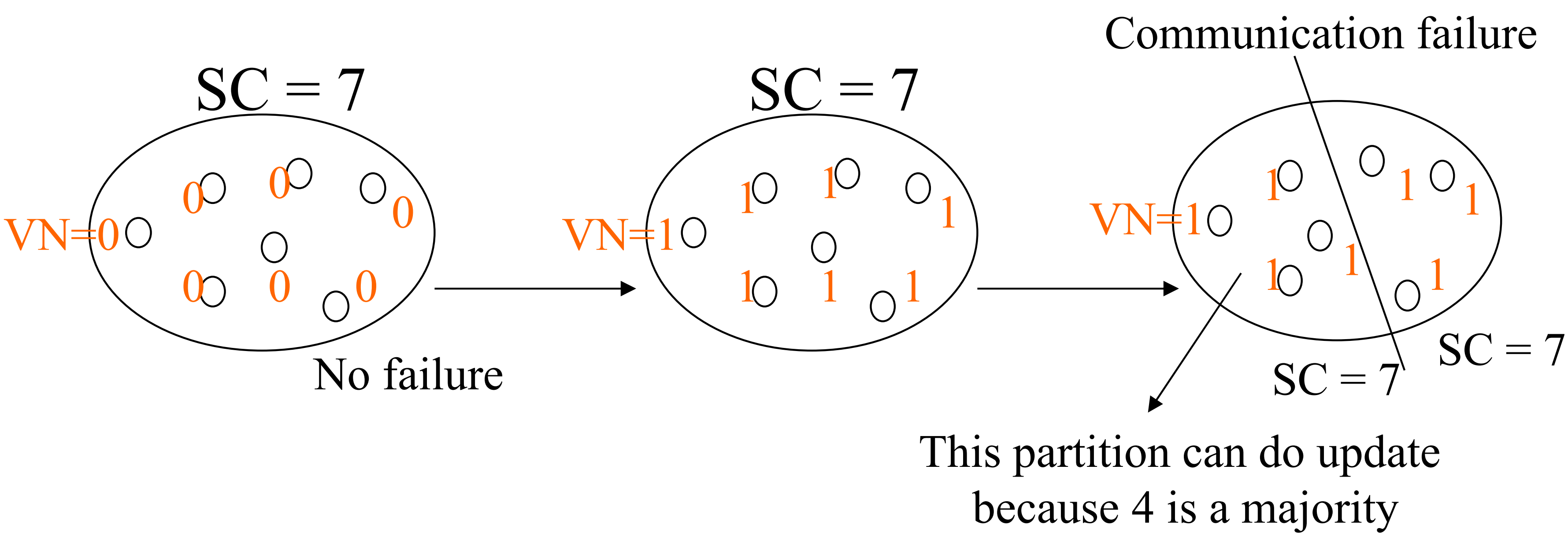
This partition can still do update

Dynamic voting:

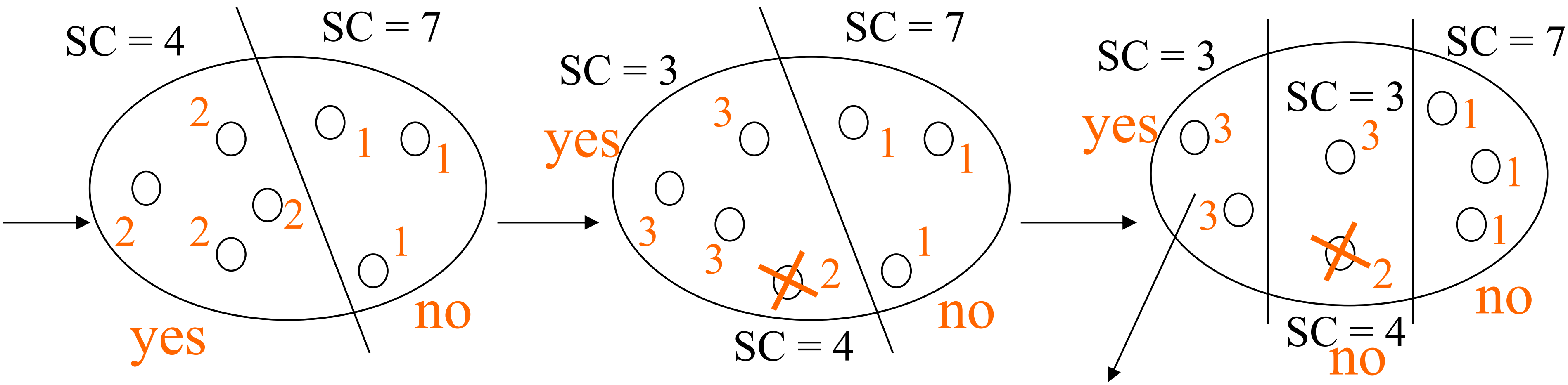
can do update if **a majority of <u>current (up-to-date) copies</u>** (since the last update) can be found and updated. These majority copies are called in the "**<u>major partition</u>**".

Each copy is associated with a set of local variables:

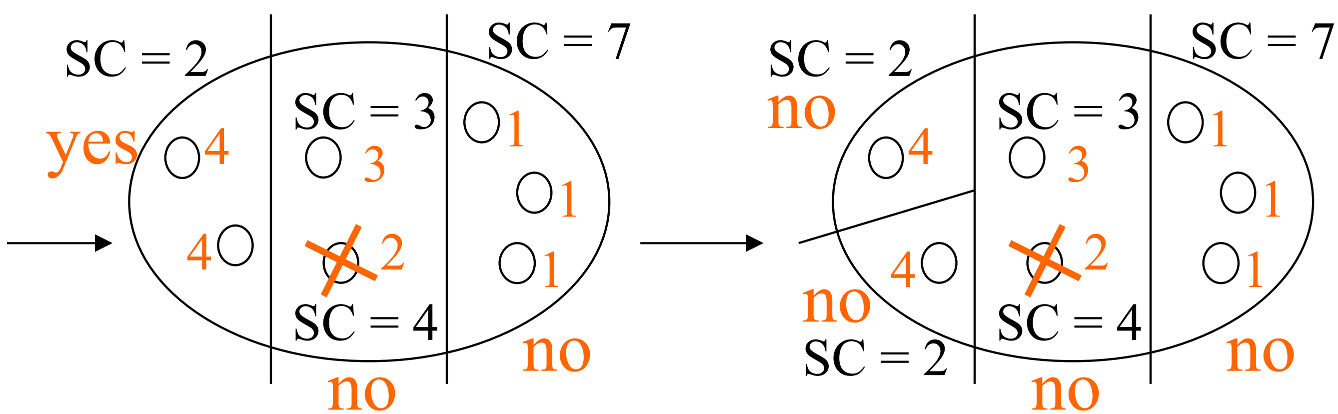
1) version number (VN): to tell if the local copy is current
2) site cardinality (SC): to tell how many copies are current, e.g., if in the last update, 5 copies were updated, then SC = 5



** All copies within the major partition are updated & the new SC is set to the # of copies in the major partition.

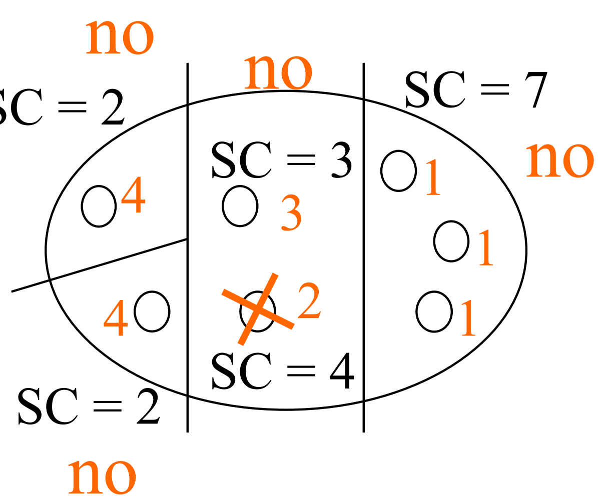


This partition can do update because 2 is a majority of SC=3



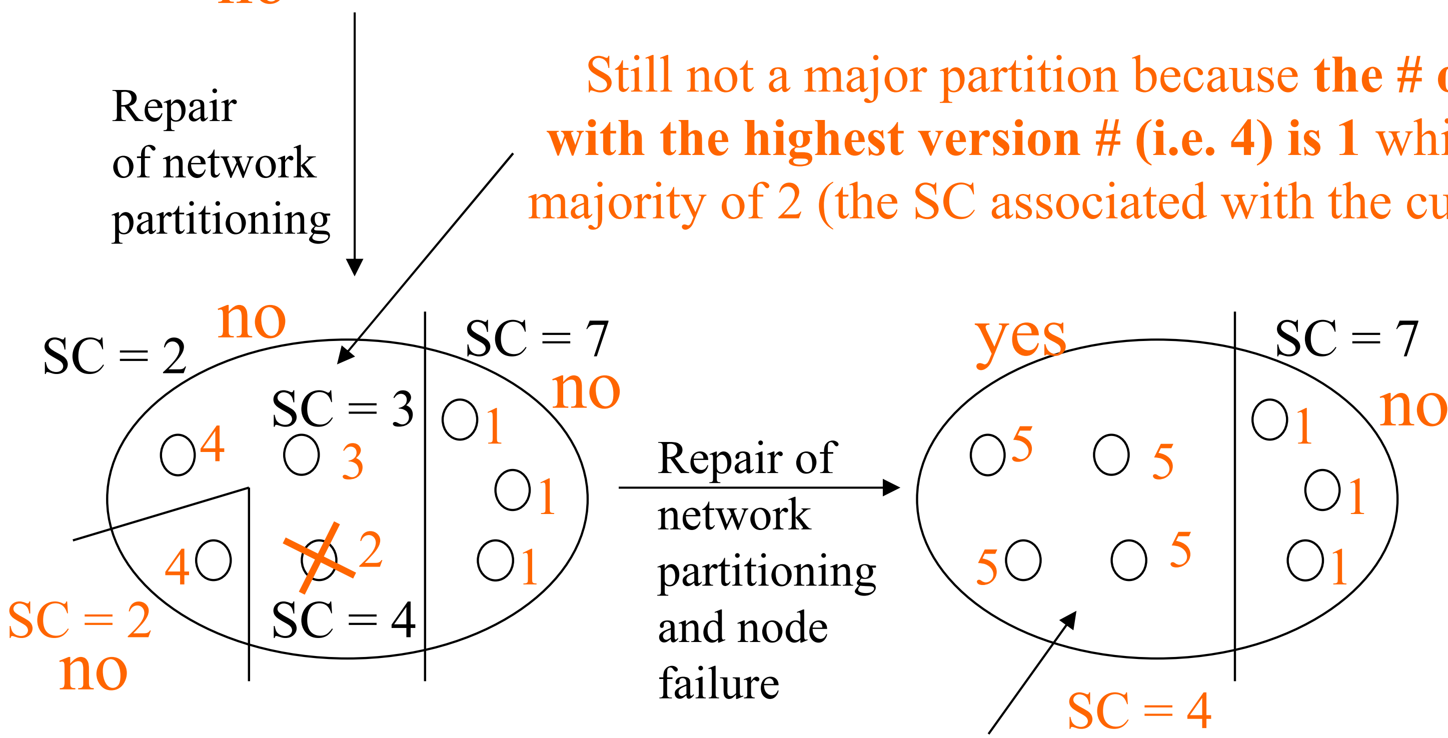
No partition can do update. System halts & must wait for repairs to occur.

Reunion Scenarios



No major partition exists

Still not a major partition because **the # of copies with the highest version # (i.e. 4) is 1** which is not a majority of 2 (the SC associated with the current copy)



No major partition exists

A major partition now

Availability modeling:

Site-failure only model: there is only one partition

system models:

1) failure rate of each site is λ
2) repair rate of each site is μ
3) updates are frequent and there is always an update immediately following a failure/repair.

Static voting: system is available as long as k out of n are available, so the "site availability" is given by:

$$A(\infty) = \sum_{k=\left\lfloor \frac{n}{2} \right\rfloor + 1}^{n} \frac{k}{n} \binom{n}{k} \left(\frac{\mu}{\lambda + \mu} \right)^{k} \left(1 - \frac{\mu}{\lambda + \mu} \right)^{n-k}$$

$$\frac{k}{n} = prob \left\{ \begin{array}{l} \text{an update request arrives at one} \\ \text{of } k \text{ sites in the major partition} \end{array} \right\}$$

Dynamic voting: no simple probability expression exists

Resort to Markov modeling

or

Petri net modeling

state representation

(X, Y, Z)

X of Y current copies are alive
∴ Y-X of Y current copies are down

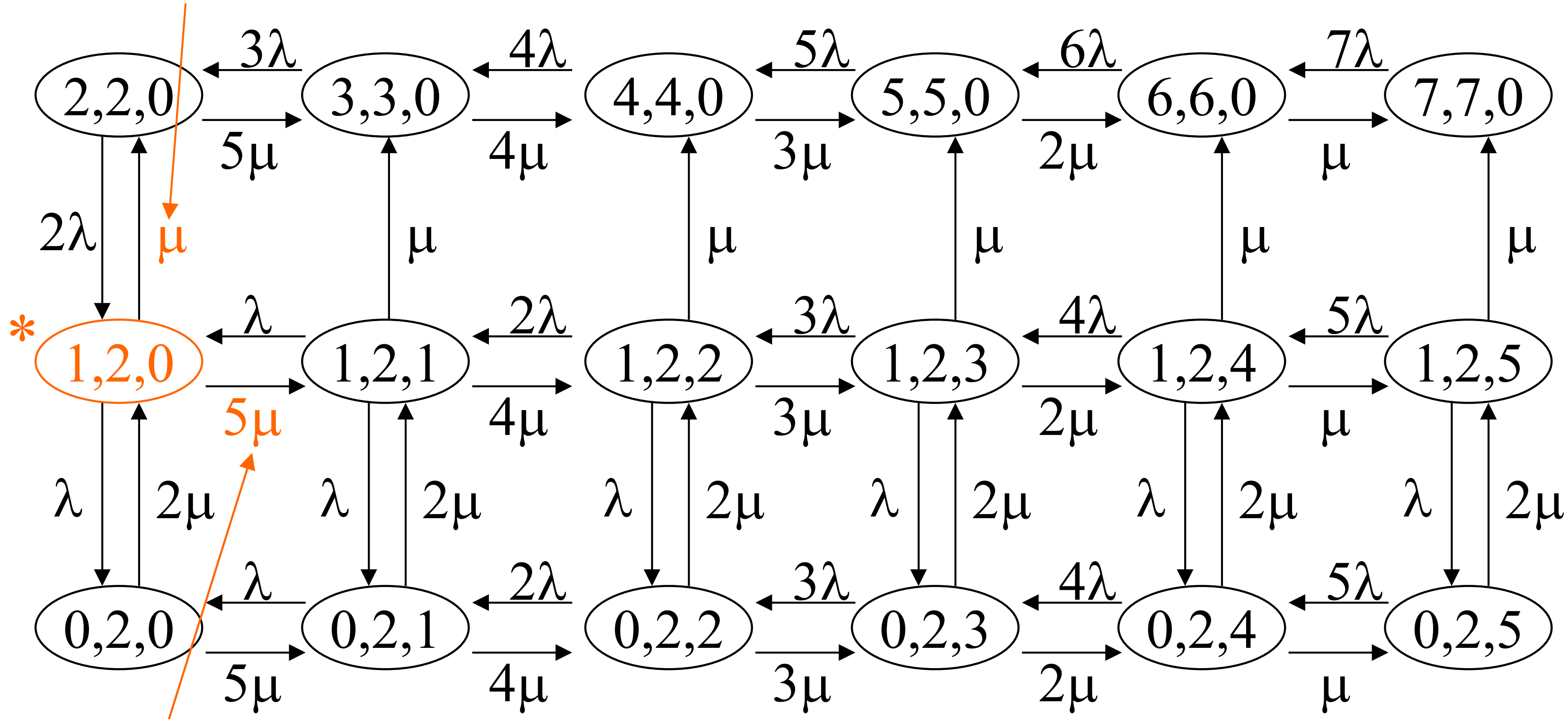
Y = current site cardinality (**SC**) or # of current copies

n: # of initial copies (e.g., n=7)

Z of the n-Y other sites are alive but out-of-date

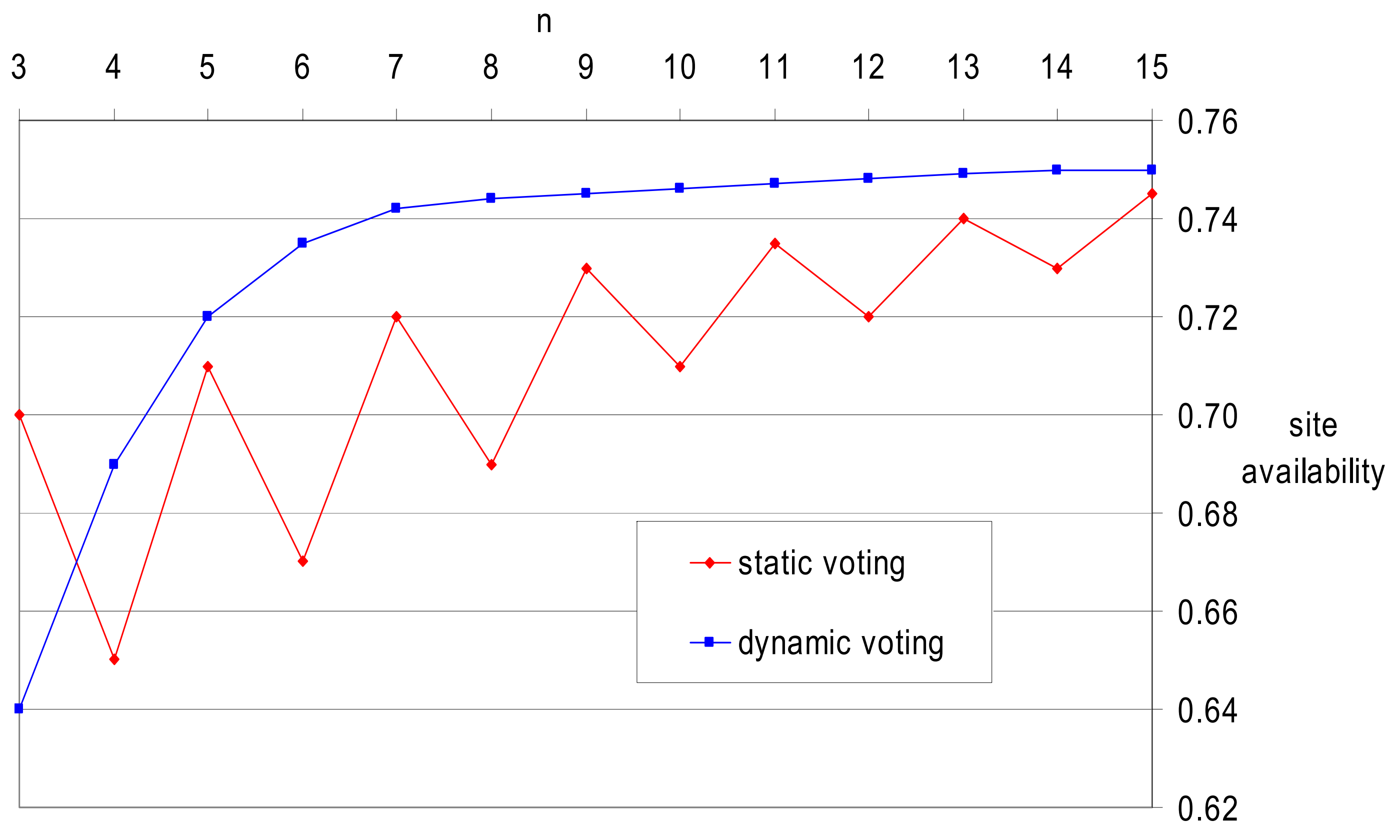
Site Availability: $$A(\infty) = \sum_{i=2}^{n \,=7} \left\{ prob\{(i,i,0)\} \times \frac{i}{n} \right\}$$

repair of a current copy in the major partition



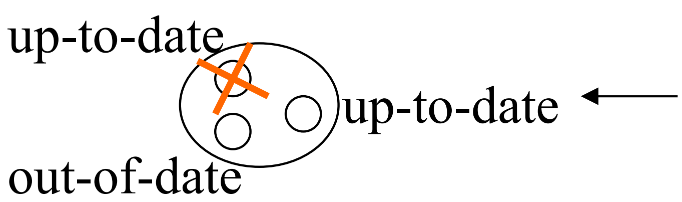
repair of an out-of-date copy in the major partition

* in state (1,2,0): no update can be performed because 1 is not a majority of 2.



<u>Site availability comparison results</u>:

* static voting is better than dynamic voting when n=3



Update is permitted in static voting but not permitted in dynamic voting

* when n>3 dynamic voting is better