


**CS 347:**  
 Distributed Databases and  
 Transaction Processing  
**Notes 09: Network Partitions**  
  
 Hector Garcia-Molina

CS347
Notes09
1

Network partitions

- Groups of nodes may be isolated or nodes may be slow in responding



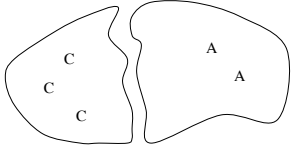
CS347
Notes09
2

- Where are partitions of interest?
  - True network partitions (disaster)
  - Single node failure cannot be distinguished from partition (e.g., ethernet board fails)
  - Phone-in, wireless networks
  - Autonomous nodes

CS347
Notes09
3

No data replication

- If data unavailable, we are stuck...
- A problem: partition during commit protocol



CS347
Notes09
4

Quorums

```

      . a
    b . . c
      . d
  
```

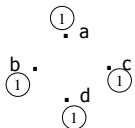
$C_1 = \{\{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}\}$   
 $A_1 = \{\{a,b\}, \{a,c\}, \{a,d\}, \{b,c\}, \{b,d\}, \{c,d\}\}$

Important property:  $X \in C_1 \Rightarrow \forall Y \in A_1 \ X \cap Y \neq \emptyset$   
 $Y \in A_1 \Rightarrow \forall X \in C_1 \ X \cap Y \neq \emptyset$

CS347
Notes09
5

- Quorums can be implemented with vote assignments

To commit  $\geq 3$   
 To abort  $\geq 2$



Votes to commit + votes to abort > total votes

CS347
Notes09
6

- Commit protocol must enforce quorum
  - a .
  - b
  - c
  - d
- If node knows transaction could have committed (aborted), if cannot abort (commit) even if abort (commit) quorum available
- All commit protocols are blocking (with network partitions)
  - can we do anything about it?

CS347 Notes09 7

### 3PC Example

- To make commit decision: commit quorum
- To make abort decision: abort quorum
- Example:
  - votes for commit  $V_c = 3$
  - votes for abort  $V_a = 3$

CS347 Notes09 8

- Coordinator could NOT have committed
- Have abort quorum
- ⇒ Try to abort!

CS347 Notes09 9

- Note: need to go to Prepare to Abort state (PA), analogous to Prepare to Commit state (PC)

(1) w	w	PA	PA	A
(1) w	PA	PA	PA	A
(1) w	PA	PA	A	A

time →

CS347 Notes09 10

- What if new group has following states?
  - (1) w
  - (1) PC
  - (1) PA
- Can this happen? How?
- Could transaction have aborted?
- Could transaction have committed?
- What do we do? Block?

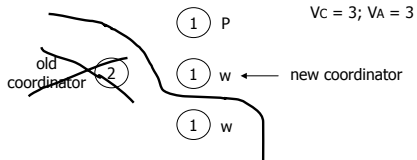
CS347 Notes09 11

### Another 3PC Example

- Coordinator could not have aborted
- Have commit quorum
- ⇒ Try to commit!

CS347 Notes09 12

### Yet Another 3PC Example



- Not enough votes!  
⇒ Block!

CS347

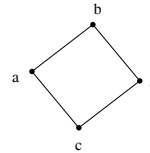
Notes09

13

### Not all quorums can be implemented via votes

$$C_2 = \{\{a,b\}, \{c,d\}\}$$

$$A_2 = \{\{a,c\}, \{a,d\}, \{b,c\}, \{b,d\}\}$$

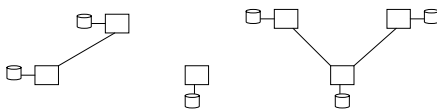


CS347

Notes09

14

### Partitions and data replication



#### Options:

- (1) all copies required for updates
- (2) Group may update, but at most one (at a time)
- (3) Any group may update

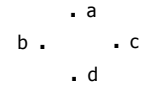
CS347

Notes09

15

### Updates by at most one group

#### Coterie



$$C_1 = \{\{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}\}$$

$$C_2 = \{\{a,b\}, \{a,c\}, \{a,d\}, \{b,c,d\}\}$$

$$X_1 = \{\{a,b\}, \{c,d\}\} \text{ not valid}$$

Important property:

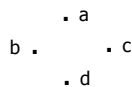
$$S \in C \Rightarrow \forall G \in C, S \cap G \neq \emptyset$$

CS347

Notes09

16

### Reading replicated data



$$C_1 = \{\{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}\}$$

$$R_1 = \{\{a,b\}, \{a,c\}, \{a,d\}, \{b,c\}, \{b,d\}, \{c,d\}\}$$

$$C_2 = R_2 = \{\{a,b\}, \{a,c\}, \{a,d\}, \{b,c,d\}\}$$

CS347

Notes09

17

### Reading replicated data - Votes

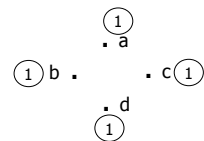
$$C_1 = \{\{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}\}$$

$$R_1 = \{\{a,b\}, \{a,c\}, \{a,d\}, \{b,c\}, \{b,d\}, \{c,d\}\}$$

to write get 3 votes ( $V_w$ )

to read get 2 votes ( $V_R$ )

( $2 V_w > T, V_w + V_R > T$ )



CS347

Notes09

18

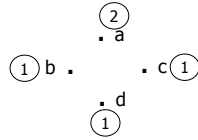
### Reading replicated data - Votes

$$C_2 = R_2 = \{\{a,b\}, \{a,c\}, \{a,d\}, \{b,c,d\}\}$$

to write get 3 votes ( $V_w$ )

to read get 3 votes ( $V_R$ )

( $2 V_w > T, V_w + V_R > T$ )



CS347

Notes09

19

### Note

- Note: not all coterie have vote assignments

CS347

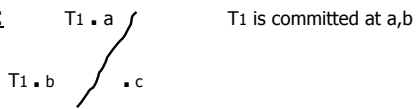
Notes09

20

### A Problem

Example: a 3 node system, 1 vote each node, replicated data

Now:

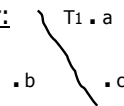


CS347

Notes09

21

Later:



T2 reads at c (not seeing T1); then writes and commits at a, c

CS347

Notes09

22

### Solution

- each node keeps list of committed transactions
- compare list at read site with those at write sites
- update sites that missed transactions

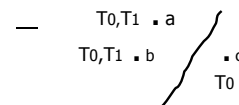
CS347

Notes09

23

### Example Revisited

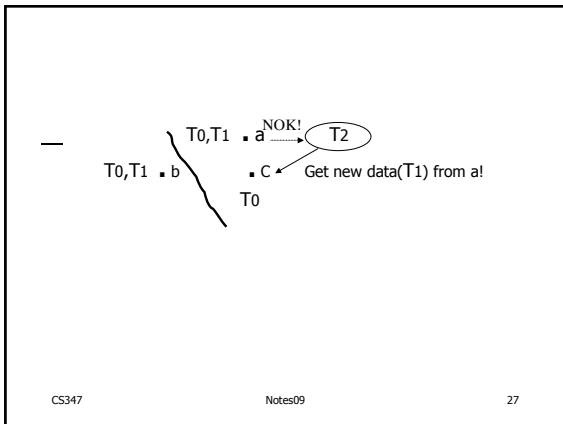
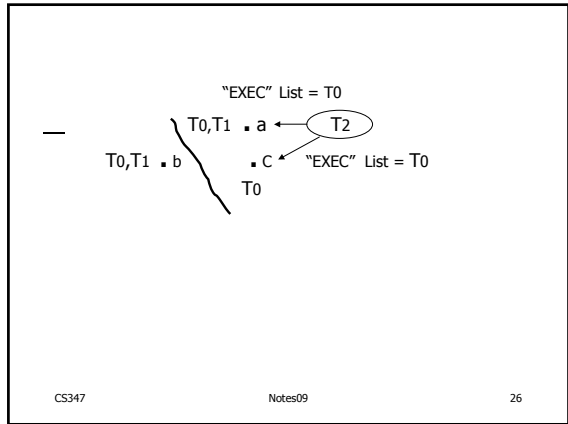
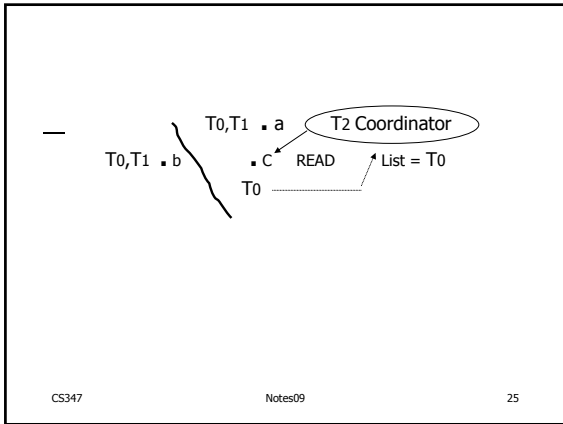
Initially



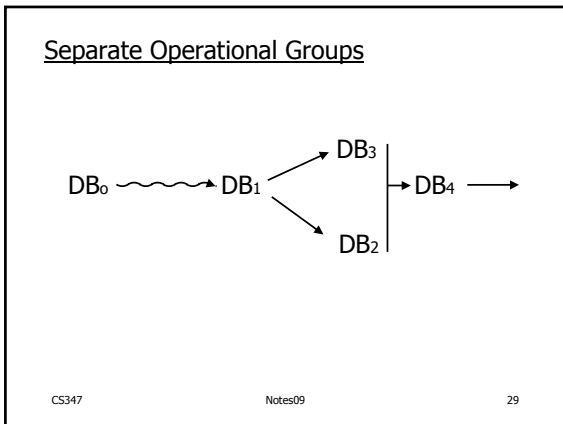
CS347

Notes09

24



- Each node must keep updates for transactions until all nodes have seen them
    - interesting problem...
- CS347 Notes09 28



- For integration
- (1) Compensate transactions to make schedules match
  - (2) Data-patch: semantic fix
- CS347 Notes09 30

Example: compensation

CS347 Notes09 31

- Say T1 commutes with T3 and T4  
E.g.: no conflicting operations
- At DB2:  
Schedule = T0, T3, T4, T1  
(equivalent to T0, T1, T3, T4)

CS347 Notes09 32

- At DB3:  
Schedule = T0, T1, T2, T2<sup>-1</sup>, T3, T4  
(equivalent to T0, T1, T3, T4)

CS347 Notes09 33

- In general:  
Based on characteristics of transactions,  
can “merge” schedules

CS347 Notes09 34

Example: Data Patch

- forget schedules!
- integrate differing values via “rules”

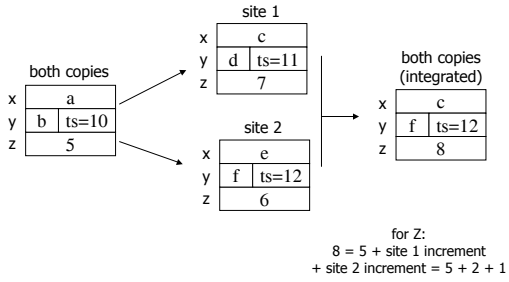
CS347 Notes09 35

Simple rules

- For X: site 1 wins
- For Y: latest timestamp wins
- For Z: add increments

CS347 Notes09 36

For X: site 1 wins  
 For Y: latest timestamp wins  
 For Z: add increments



CS347

Notes09

37

## Network Partitions: Summary

- No replicated data
  - abort, commit quorums
  - commit protocols
- Replicated data
  - at most one operational group
    - coterie
    - propagation of updates
  - multiple operational groups

CS347

Notes09

38