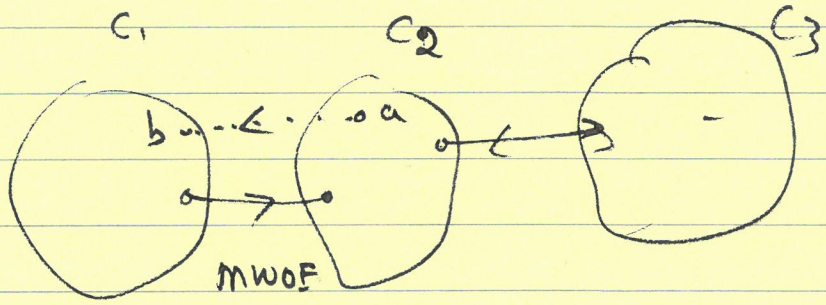
10-15-19

① Asynchronous MST or
Asynch GHS

Complications because of asynchrony:

A. Some Components are "too fast"
Compared to others.

All proceses of a Combined Component need to know about new Component id.



C_1 absorbed into C_2

C_2 ⊛ Combined with C_3
(C_2 & C_3 merged)

Later a sends a test message on (a, b) to see if (a, b) is an outgoing edge

b may "erroneously" send an accept reply.

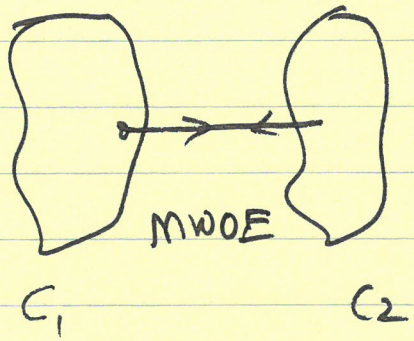
B. "Big" Component finds MWOE to a "small" Component
This results in "big" message Complexity

(2) How to take care of these Complications?

We have levels for Components. (initially level 0, ~~0~~ for each process).

(a)

merge

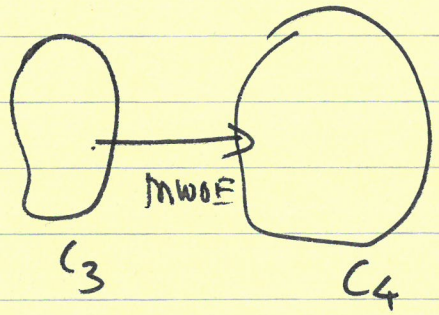


C_1 at Level L & C_2 at Level L Combine (merge) using Common MWOE

$C_1 \cup C_2 \cup$ {Common MWOE} is at level L+1

(b)

absorb



MWOE (C_3) ~~[illegible]~~ Connects to C_4 and Level (C_3) < Level (C_4)

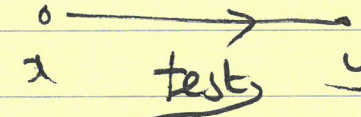
All processes in C_3 take on C_4's Component id & C_4's level.

Initially, each process is a Component by itself and at level 0.

Finding MWOE of a Component

Leader broadcasts an initiate (id, Level) message

x needs to find MWOE Candidate incident on x.



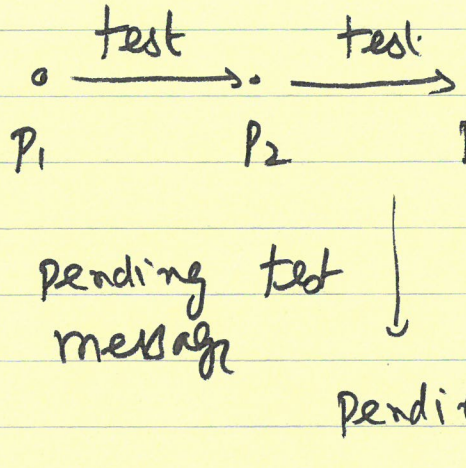
(Component id, Level)

wait for a reply

y sends a reply if Level of y $\geq$ Level of x. and send an accept or reject

else { // Level(x) > Level(y):
defer reply until y's Level is greater than or equal to x
}

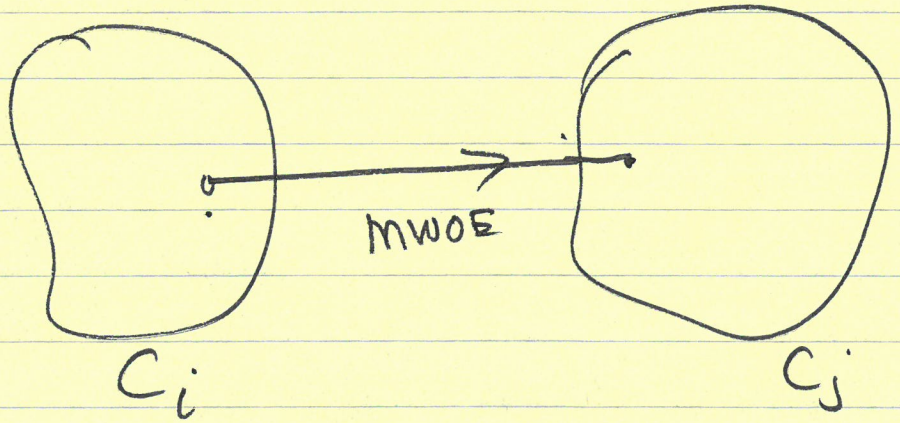
④ Deadlock possible because of these pending replies?



these lead to successively smaller levels; no cycle; no deadlocks.

All processes of a component find MWOE

Combine with component at other end of MWOE



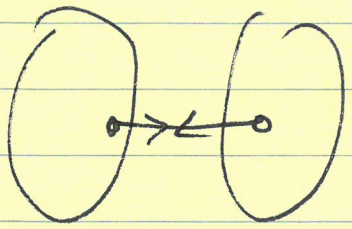
(5)

Two cases

(a) merge : $MWOE(C_i) = MWOE(C_j)$

$Level(C_i) = Level(C_j)$

Common MWOE is the core edge.

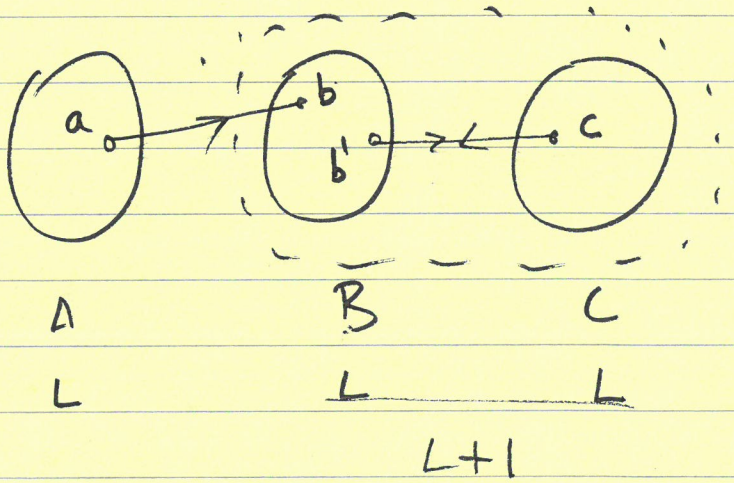


$C_i \cup C_j \cup MWOE(C_i \text{ or } C_j)$ C_i C_j

Is at level $Level(C_i) + 1$

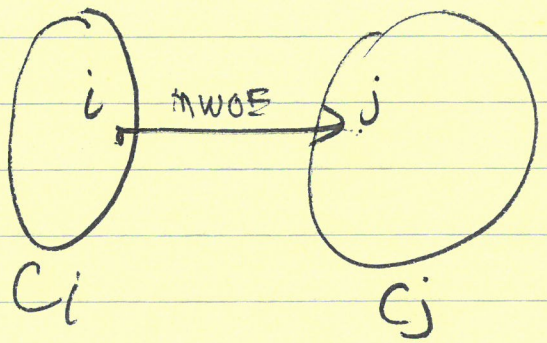
One end point of Common MWOE is the Cendra of Combined Component

edge weight of Common MWOE is the new (Combined) Component's id.



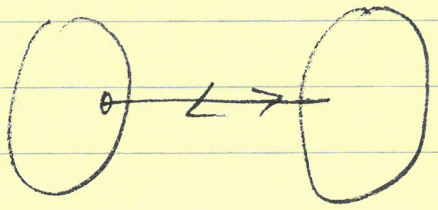
(b) ~~Ab~~ Absorb

⑥ Absorb operation.



$MWOE(C_j) \neq MWOE(C_i)$

$\Rightarrow$ Level (C_i) < Level (C_j) at some some time in the future, and, at that time C_i will be absorbed into C_j



$L_x < L_y$

When C_i at Level L_i is absorbed into C_j at Level L_j ($L_j > L_i$) Processes in C_i take on C_j's Component id and Level L_j

6

C_i's procs participate in finding MWOE of $C_i \cup C_j \cup MWOE(C_i)$

- think about the 2 cases
- (i) j already sent the Convergecast reply to C_j's leader and
- (ii) j has NOT yet sent the Convergecast reply to j's parent

Max level: $\lceil \log_2 n \rceil$

min # of procs in any component of level L is 2^L

Message Complexity: $(|E| + n \log n)$ (test/reject)

Time Complexity: $O(n \log n)$

test/accept

Combine, new id. Convergecast