Bakery Algorithm

Code for entry section:

```
Choosing[i] := true
Number[i] := max{Number[0], …, Number[n-1]} + 1
Choosing[i] := false
for j := 0 to n-1 (except i) do
    wait until Choosing[j] = false
    wait until Number[j] = 0 or
     (Number[j],j) > (Number[i],i)
endfor
```

Code for exit section:

```
Number[i] := 0
```

2-Processor Mutex Algorithm

Code for entry section:

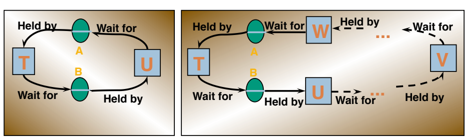
```
1  W[i] := 0
2  wait until W[1-i] = 0 or Priority = i
3  W[i] := 1
4  if (Priority = 1-i) then
5       if (W[1-i] = 1) then goto Line 1
6  else wait until (W[1-i] = 0)
```

Code for exit section:

```
7  Priority := 1-i
8  W[i] := 0
```

Deadlocks

- Necessary conditions for deadlocks
  - Non-shareable resources (locked objects)
  - No preemption on locks
  - Hold & Wait
  - Circular Wait (Wait-for graph)

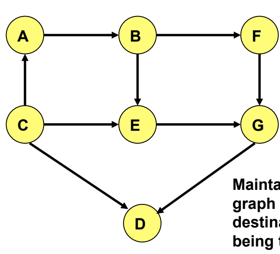


Validation of Transactions

Backward validation of transaction T_v
    boolean valid = true;
    for (int T_i = *startTn*+1; T_i <= *finishTn*; T_i++){
        if (read set of T_v intersects write set of T_i) valid = false;
    }

Forward validation of transaction T_v
    boolean valid = true;
    for (int T_{id} = *active1*; T_{id} <= *activeN*; T_{id}++){
        if (write set of T_v intersects read set of T_{id}) valid = false;
    }

Link Reversal Algorithm



Links are bi-directional

But algorithm imposes logical directions on them

Maintain a directed acyclic graph (DAG) for each destination, with the destination being the *only sink*

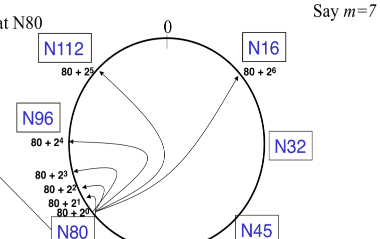
This DAG is for *destination node D*

Peer pointers (2): finger tables

Say m=7

Finger Table at N80

i	$ft[i]$
0	96
1	96
2	96
3	96
4	96
5	112
6	16



ith entry at peer with id n is first peer with id >= $n+2^i(\bmod 2^m)$