Operating Systems

Deadlocks

Mindset

Think: Listen, read, explore, try, think

Overview of Deadlocks

- Deadlock Definition
- Necessary Conditions for Deadlocks
- Dealing with deadlocks
  - Prevention
  - Avoidance
  - Detection
  - Recovery

Review of Resource Sharing

- Several Processes
- Finite Resources
- Resource Use
  - Request
  - Use
  - Release
- Issues
  - Resolve race conditions with various mechanisms (e.g. mutex)
  - **Deadlocks** are possible

Example of Deadlock: Bank Transfer Scenario

	Banker Transfer of Money
Entry Section	**lock_account**(account_a); **lock_account**(account_b);

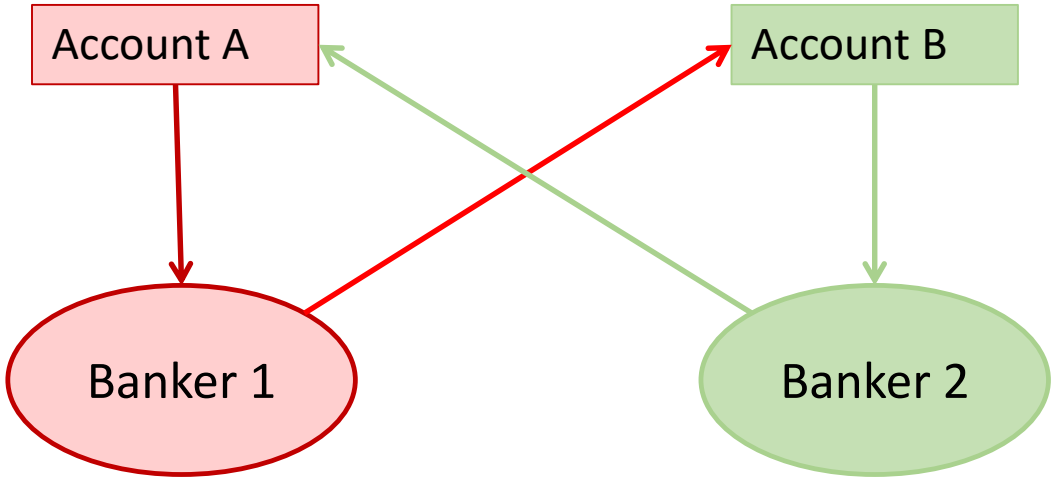
Critical Section	var balance_a = account_a_balance
	var balance_b = account_b_balance
	if (balance_a > transfer_amount)
	{
	balance_a = balance_a – transfer_amount
	balance_b = balance_b + transfer_amount
	}

Exit Section	**release_account(account_a);** **release_account(account_b);**

Example of Deadlock: Deadlock Scenario

Banker Transfer of Money

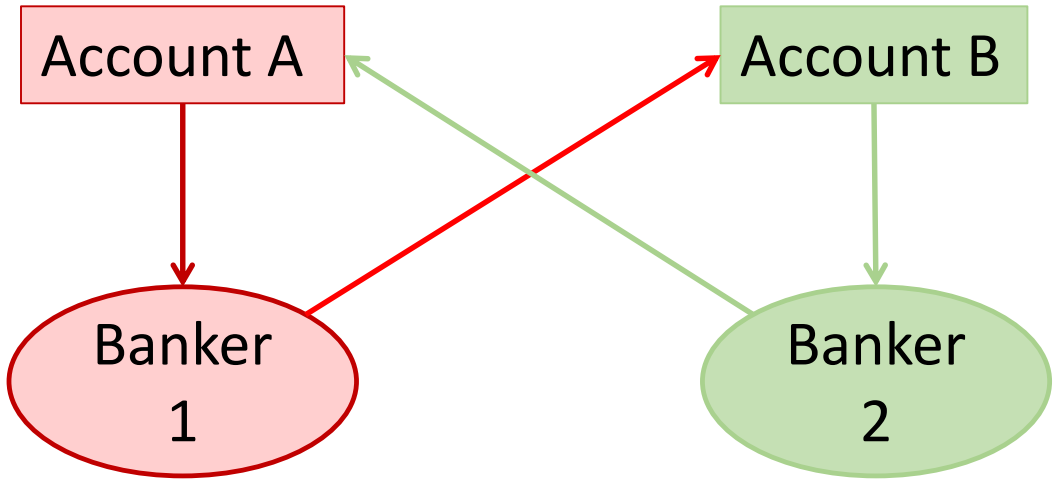
```
lock_account(account_a);
lock_account(account_b);
var balance_a = account_a_balance
var balance_b = account_b_balance
if (balance_a > transfer_amount)
{
        balance_a = balance_a – transfer_amount
        balance_b = balance_b + transfer_amount
}
release_account(account_a);
release_account(account_b);
```



Transfer from Account A to Account B

Transfer from Account B to Account A

Example of Deadlock: Conventions



P = {Banker1, Banker 2}
R = { AccountA, Account B}
E = {
AccountA->Banker1,
AccountB->Banker2,
Banker2 ->AccountA,
Banker1 ->AccountB
}

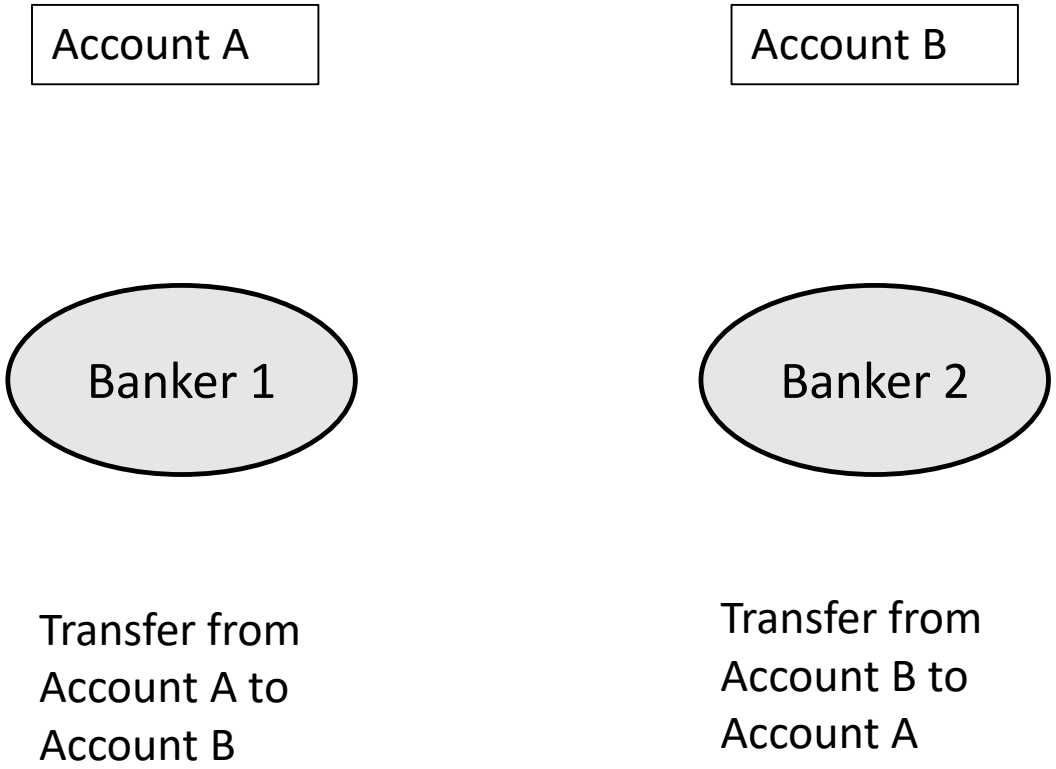
Transfer from Account A to Account B

Transfer from Account B to Account A

Example of Deadlock: 2 Bankers

Banker Transfer of Money

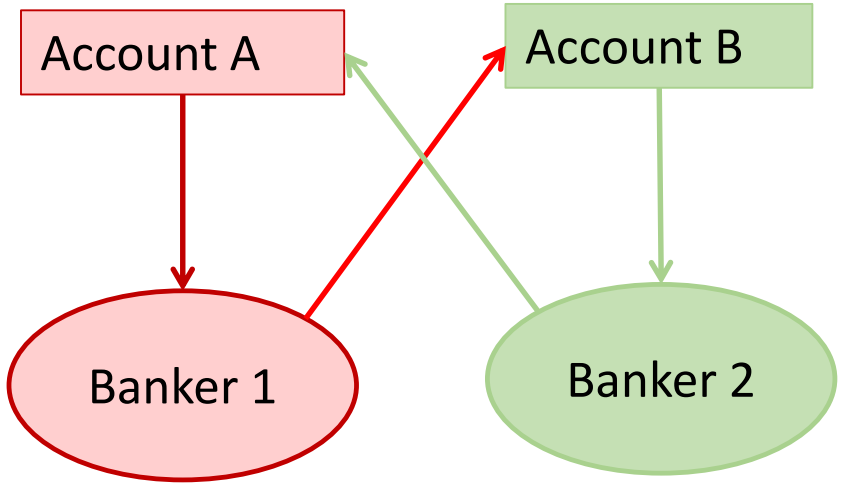
```
lock_account(account_a);
lock_account(account_b);
var balance_a = account_a_balance
var balance_b = account_b_balance
if (balance_a > transfer_amount)
{
        balance_a = balance_a – transfer_amount
        balance_b = balance_b + transfer_amount
}
release_account(account_a);
release_account(account_b);
```



Conditions for Deadlock

- Other Prerequisites
  - Multiple Resources
  - Multiple Processes
- Mutual Exclusivity
- Hold and Wait
- No Preemption
- Circular Wait

Example of Deadlock: Necessary Conditions



Transfer from Account A to Account B

Transfer from Account B to Account A

P = {Banker1, Banker 2}

R = { AccountA, Account B}

E = {

AccountA->Banker1, Banker1 hold AccountA

AccountB->Banker2, Banker2 hold AccountB

Banker2->AccountA, Banker2 waits for AccountA

Banker1->AccountB Banker1 waits for AccountB

}

Note: mutex means single arrow from resource

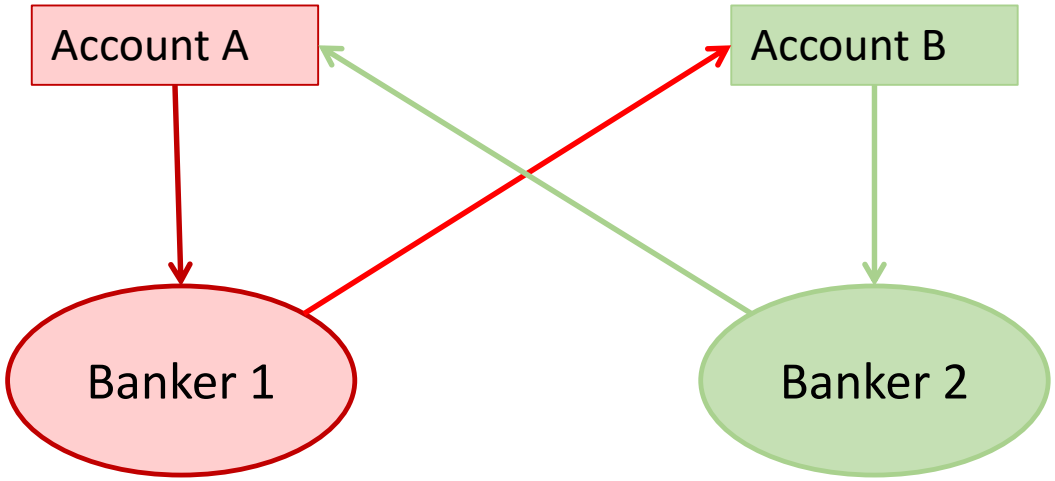
Dealing with Deadlocks

- Prevention
- Avoidance
- Detection
- Recovery

- Other Issues

Dealing with Deadlocks

- Prevention
  - Mutual Exclusivity
    - Use sharing is possible
  - Hold and Wait
    - Request all before starting
    - Hold none before request
  - No Preemption
    - Implicit yield on unsatisfied request
    - Waiting processes can have resources preempted
  - Circular Wait
    - Assign strict request order for resources



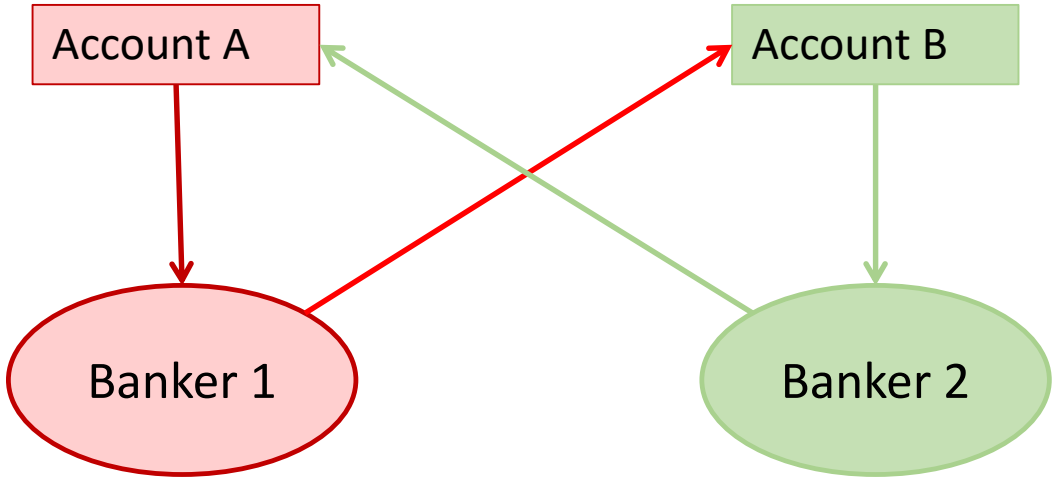
Transfer from Account A to Account B

Transfer from Account B to Account A

Dealing with Deadlocks

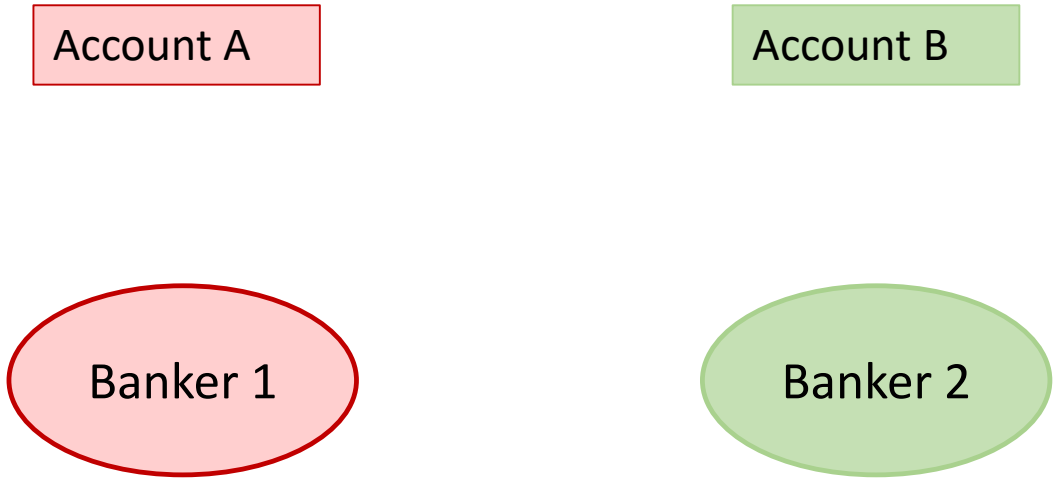
- Avoidance
  - Safe State
    - Safe States and Safe Sequences
  - Resource Allocation Graph Algorithm
    - Graph with potential requests (claim edges)
  - Banker’s Algorithm
    - Algorithm to track maximum, allocated, needed, and available resources

Safe States and Safe Sequences



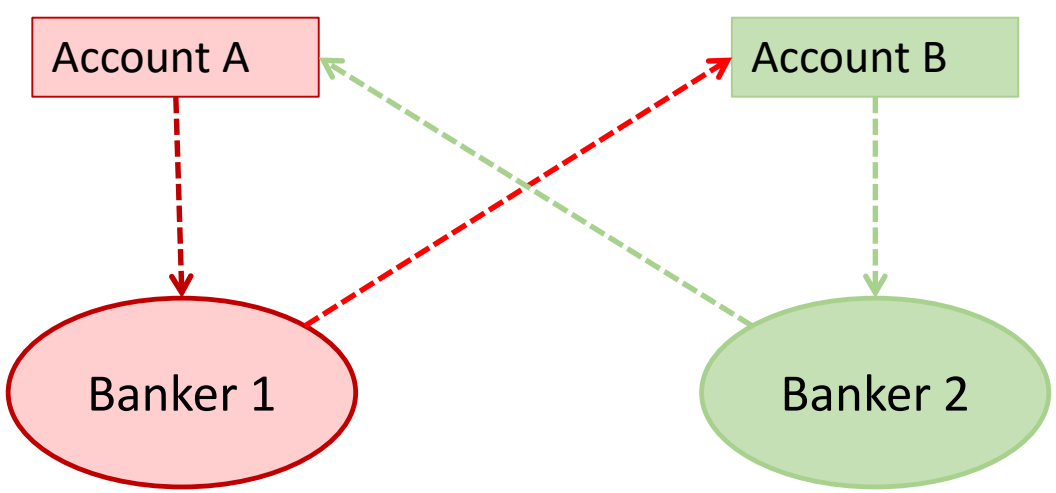
Transfer from Account A to Account B

Transfer from Account B to Account A



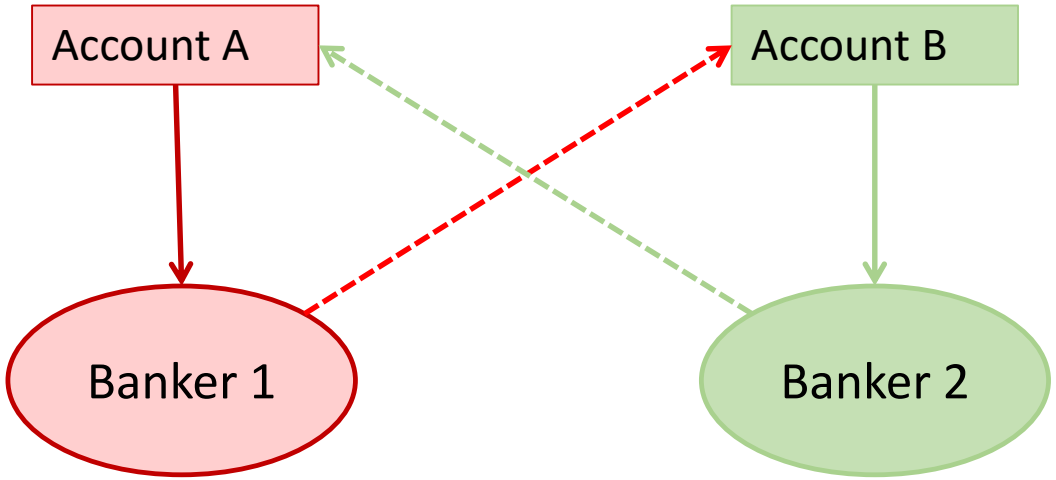
Safe Sequence
Account A -> Banker 1
Account B -> Banker 1
Banker 1 releases Account A,Account B
Account A -> Banker 2
Acouunt B -> Banker 2
Banker 2 releases Account A,Account B

Example: Claim Edges



Transfer from Account A to Account B

Transfer from Account B to Account A



Transfer from Account A to Account B

Transfer from Account B to Account A

Create a graph with “claim edges”

Banker’s Algorithm

	Maximum			Allocated		
	A	B	C	A	B	C
Process 1						
Process 2						
Process 3						

Available

A	B	C

Banker’s Algorithm

BANKER'S ALGORITHM

	Maximum			Allocated			Needed		
	A	B	C	A	B	C	A	B	C
Process 1	6	6	4	0	0	0	6	6	4
Process 2	1	7	4	0	0	0	1	7	4
Process 3	4	0	6	0	0	0	4	0	6

Available		
A	B	C
8	9	10

Total		
A	B	C
8	9	10

Is this a safe state?

Can we find a safe sequence to prove we are in safe state?

If we are not in safe state, can we prove that as well?

Practice Question: Bankers algorithm

BANKER'S ALGORITHM

	Maximum			Allocated			Needed		
	A	B	C	A	B	C	A	B	C
Process 1	6	6	4	5	0	0	1	6	4
Process 2	1	7	4	0	1	0	1	6	4
Process 3	4	0	6	2	0	4	2	0	2

Available		
A	B	C
1	8	6

Total		
A	B	C
8	9	10

Is this a safe state?

Practice Question: Bankers algorithm

BANKER'S ALGORITHM

	Maximum			Allocated			Needed		
	A	B	C	A	B	C	A	B	C
Process 1	6	6	4	3	0	0	3	6	4
Process 2	1	7	4	1	1	3	0	6	1
Process 3	4	0	6	3	0	4	1	0	2

Available		
A	B	C
1	8	3

Total		
A	B	C
8	9	10

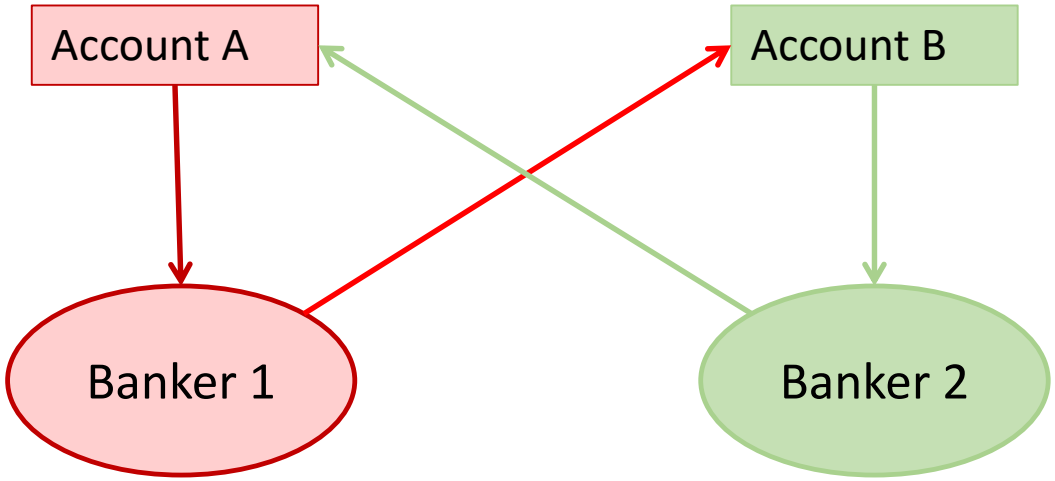
Is this a safe state?

Dealing with Deadlocks

- Detection
  - Wait-For Graphs
    - Collapse resource-allocation graph to wait-for graph
  - Issues
    - When to check for deadlocks

Example: Wait-For Graphs

Changing Resource Allocation Graph to Wait for Graph

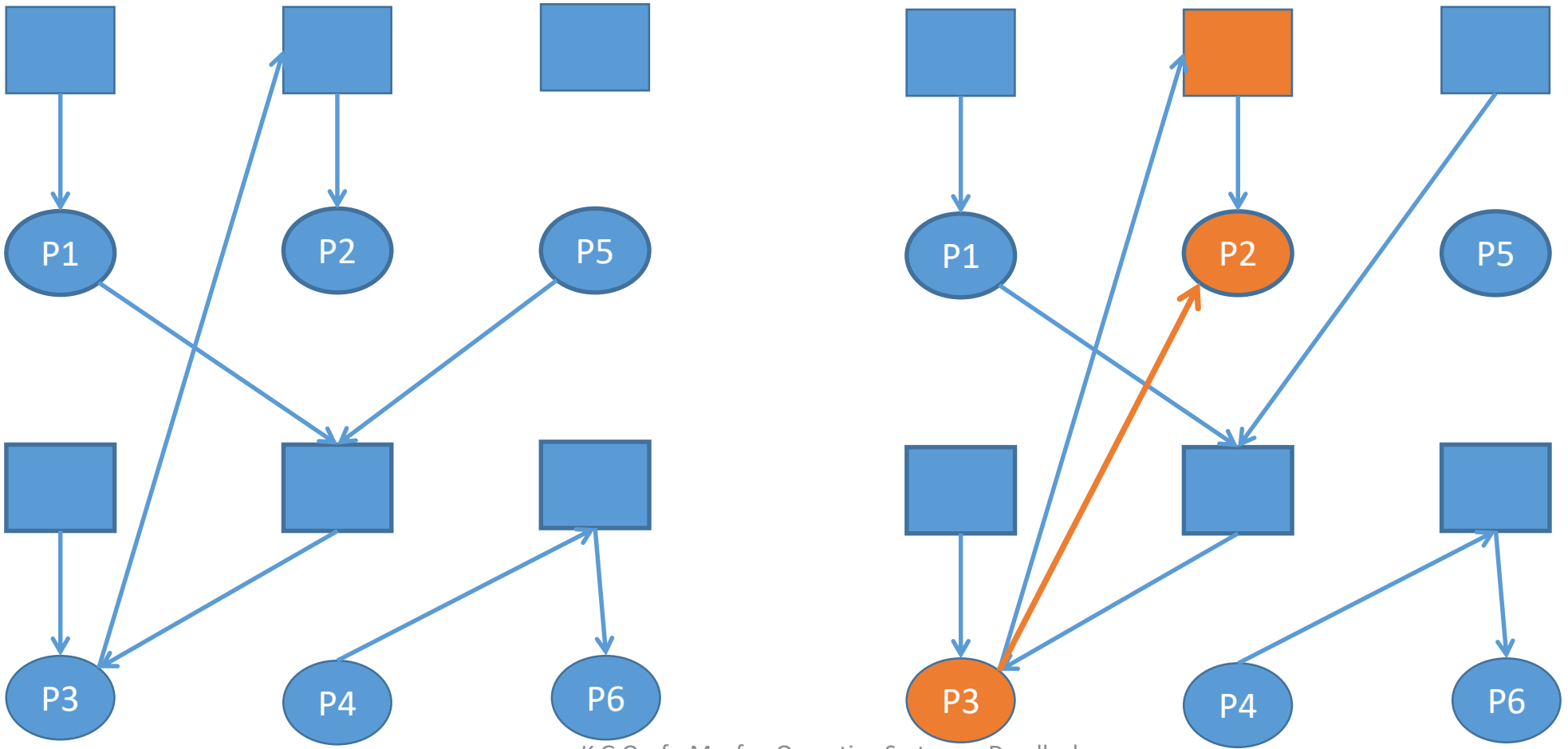


Transfer from Account A to Account B

Transfer from Account B to Account A

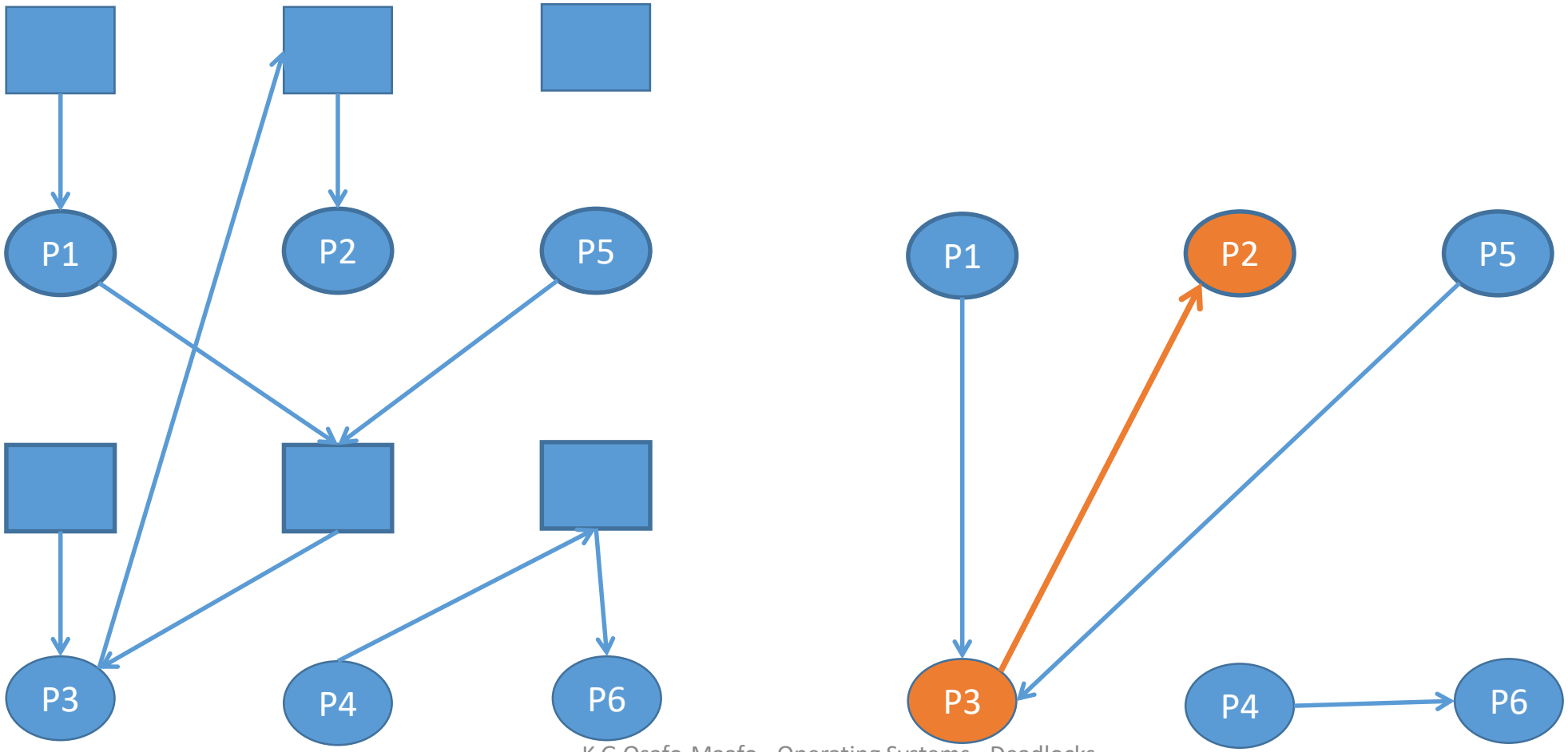
Example: Wait-For Graphs

Changing Resource Allocation Graph to Wait for Graph



Example: Wait-For Graphs

Changing Resource Allocation Graph to Wait for Graph



Dealing with Deadlocks

- Deadlock Recovery
  - Approach is about abort (then restart) Processes
    - Abort all deadlocked processes
    - Abort processes (one at a time) until deadlock is resolved
  - Issue
    - Should detect deadlocks
    - Choose a victim process (e.g. using minimum cost)

Other Issues with Deadlock Resolution

- Minimum Cost
  - Priority of Process
  - Resources used (extra: Resources needed)
  - Time used (extra: time remaining)
  - Type of process (interactive or batch)
  - Different types of resources (and if they easy to preempt)

Other Issues with Deadlock Resolution

- Resource Preemption
  - Select a Process (victim)
  - Rollback
  - Starvation

Conclusion

- Deadlocks defined
- Deadlock Conditions
- Dealing with Deadlocks
  - Prevention
  - Avoidance
  - Detection
  - Recovery
- What next?
  - Most programming languages have locks (or similar)
  - You can implement any solution or extend on any ideas discussed