

# Concurrency Groups: A New Way to Look at Real-time Multiprocessor Lock Nesting

---

CATHERINE E. NEMITZ, TANYA AMERT, MANISH GOYAL,  
JAMES H. ANDERSON



THE UNIVERSITY  
*of* NORTH CAROLINA  
*at* CHAPEL HILL

# Nested Resource Access

---

## *Nested lock request*

```
lock(A)
  lock(B)

    // critical section

  unlock(B)
unlock(A)
```

# Nested Resource Access

---

## *Nested lock request*

```
lock(A)
```

```
    lock(B)
```

```
        // critical section
```

```
    unlock(B)
```

```
unlock(A)
```

# Nested Resource Access

---

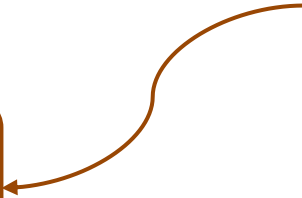
## *Nested lock request*

```
lock(A)  
  lock(B)
```

```
    // critical section
```

```
  unlock(B)  
unlock(A)
```

Both A and B  
are required



# Nested Resource Access

---

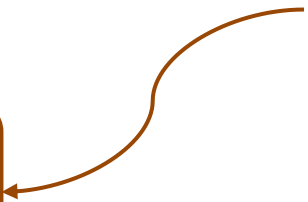
## *Nested lock request*

```
lock(A)  
  lock(B)
```

```
    // critical section
```

```
  unlock(B)  
unlock(A)
```

Both A and B  
are required



**Blocking:** delay caused by one  
or more held resources

# Nested Resource Access

---

## *Nested lock request*

```
lock (A)  
  lock (B)
```

```
// critical section
```

```
  unlock (B)  
unlock (A)
```

Both A and B  
are required

**Blocking:** delay caused by one or more held resources

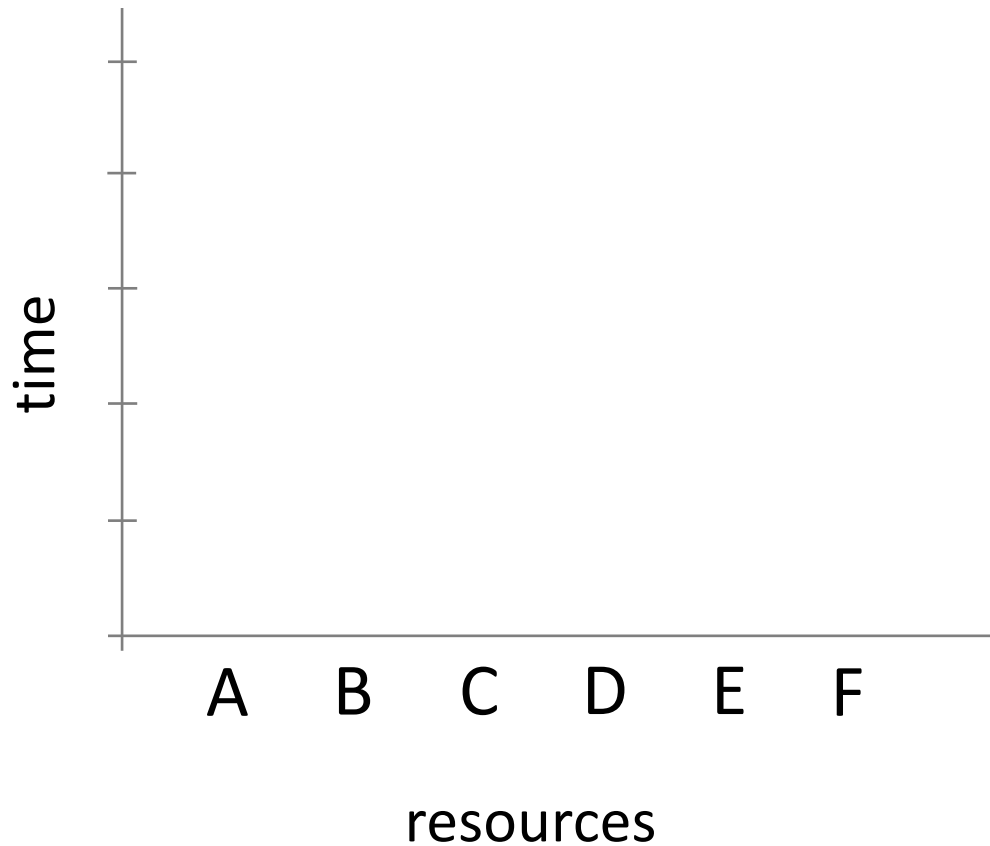
**Overhead:** delay introduced by the execution of the protocol

# Transitive Blocking Chain Problem

---

# Transitive Blocking Chain Problem

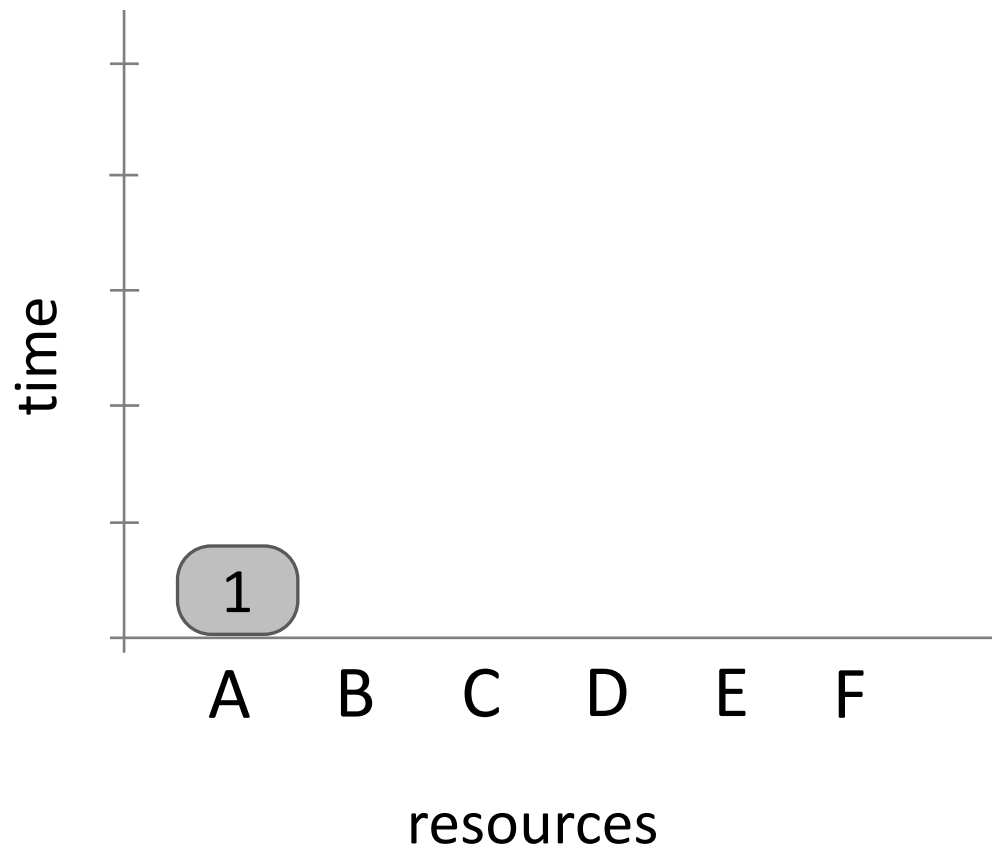
---





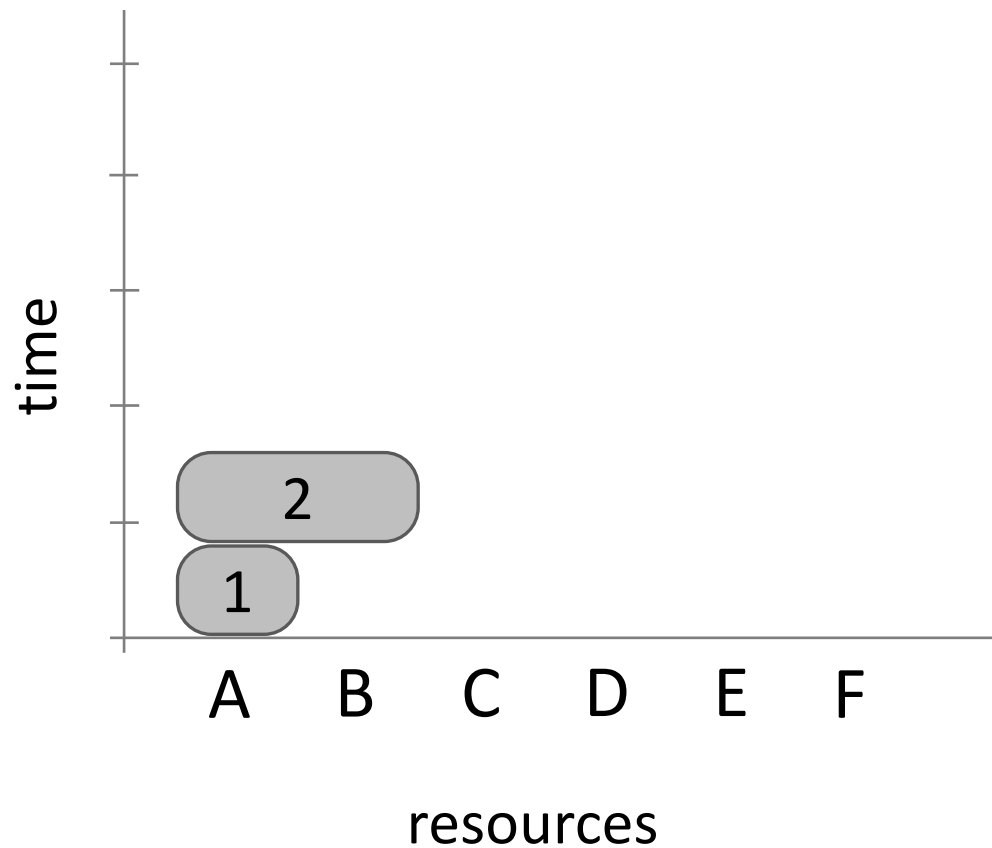
# Transitive Blocking Chain Problem

---



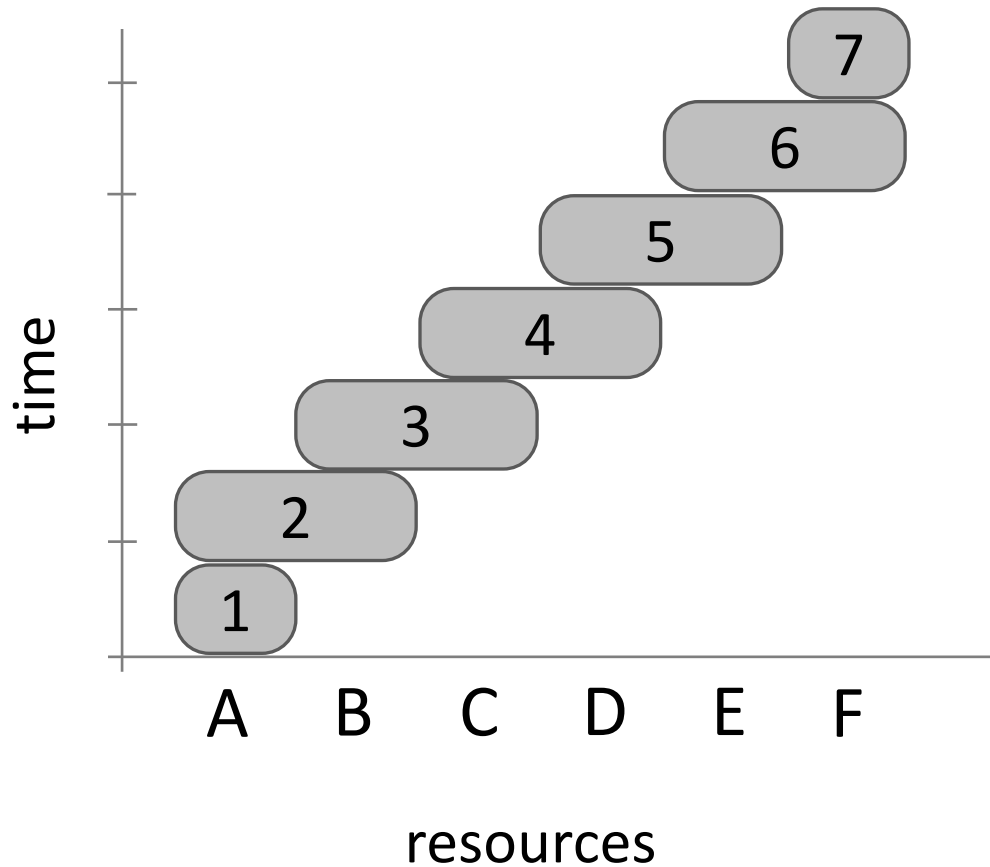
# Transitive Blocking Chain Problem

---



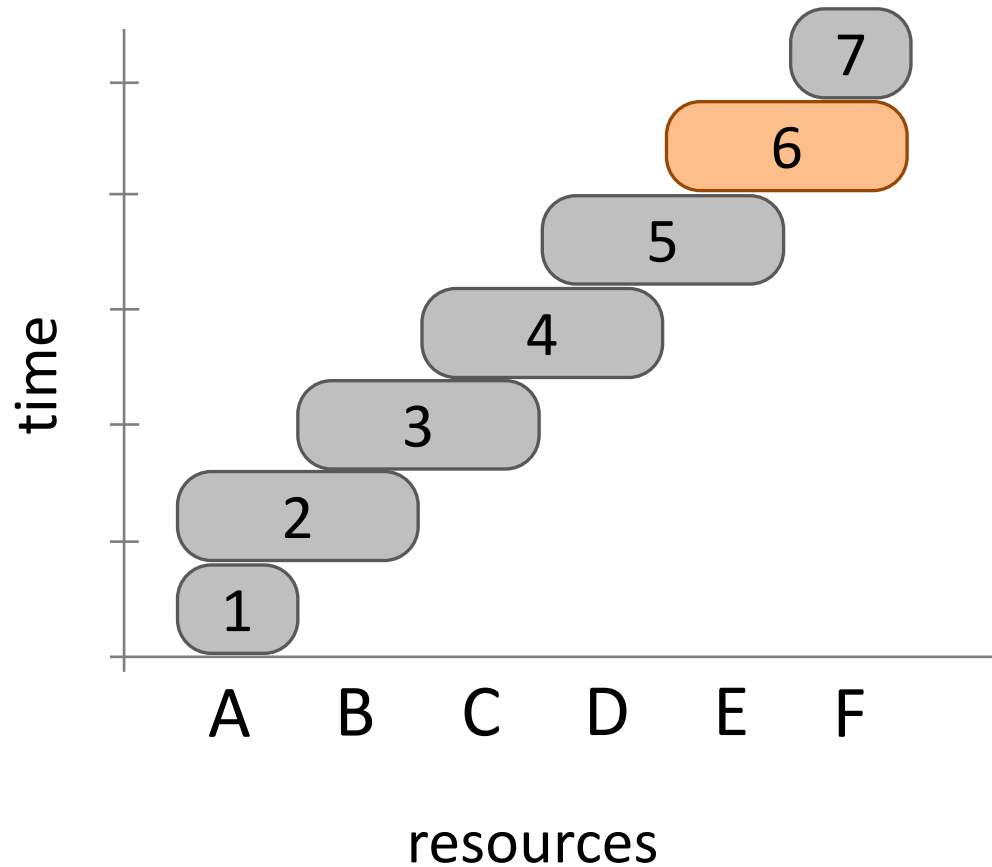
# Transitive Blocking Chain Problem

---



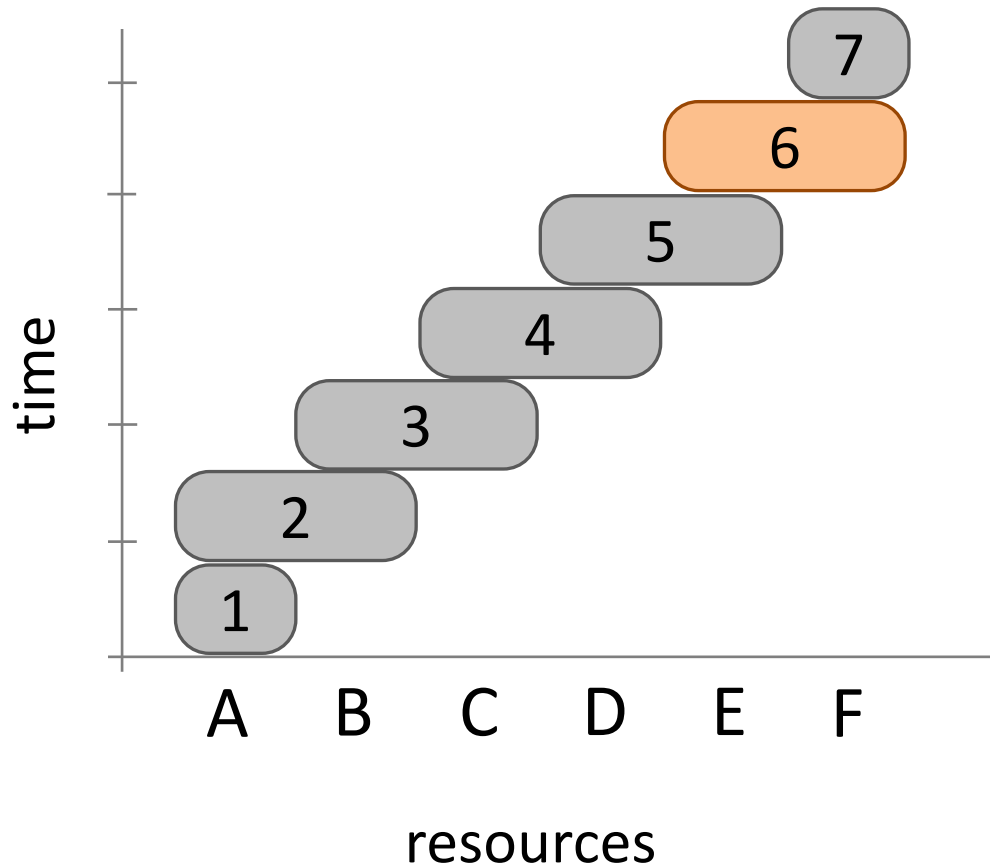
# Transitive Blocking Chain Problem

---



# Transitive Blocking Chain Problem

Requests are blocked even when resources are free

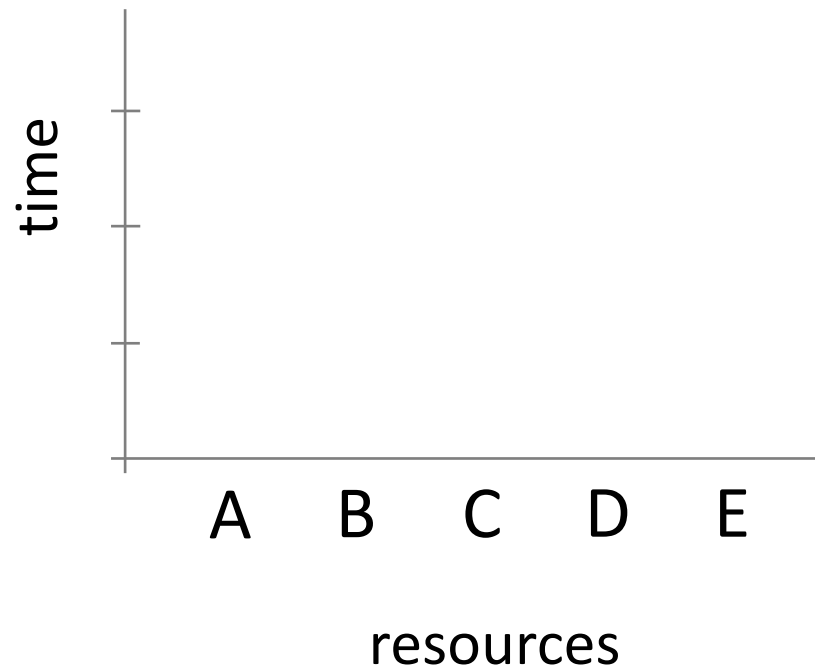


# Request Timing Problem

---

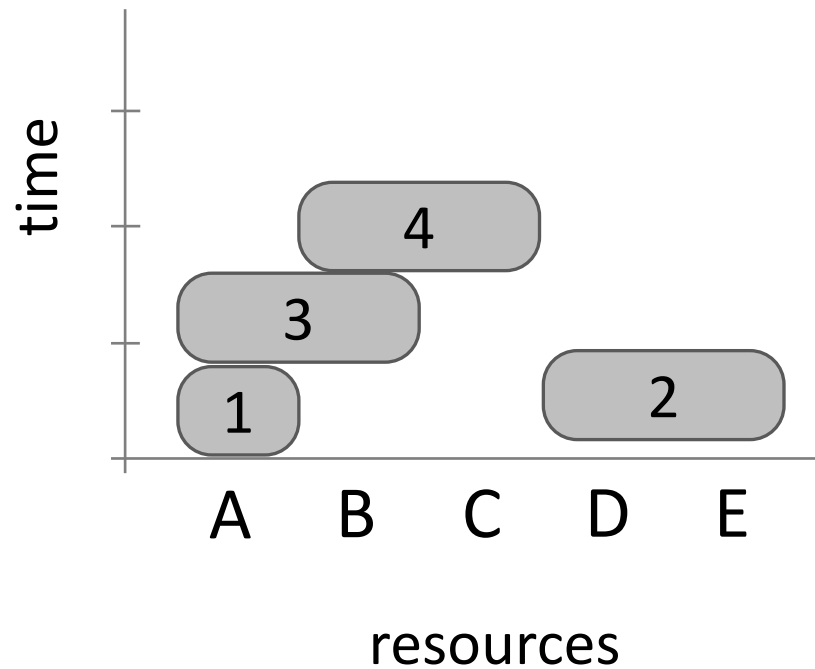
# Request Timing Problem

---



# Request Timing Problem

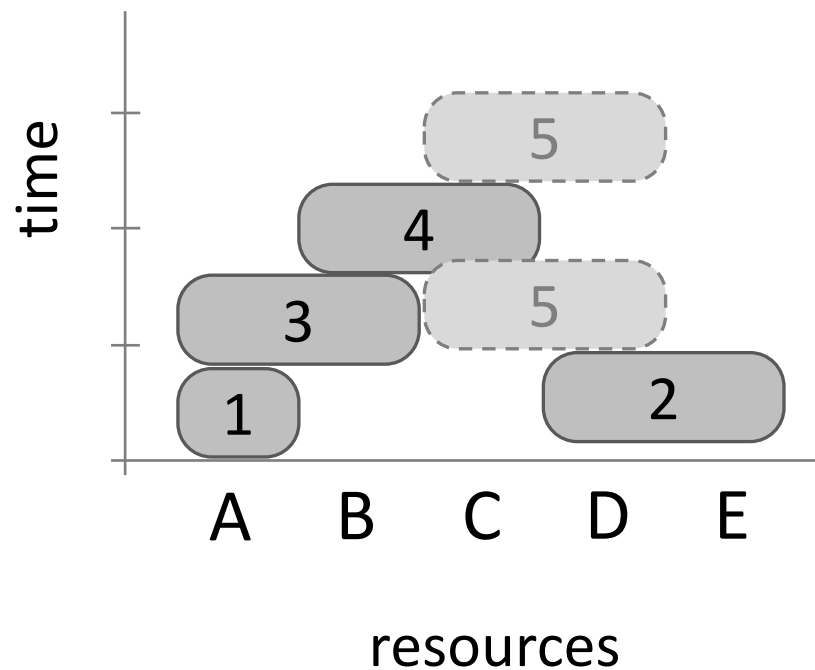
---





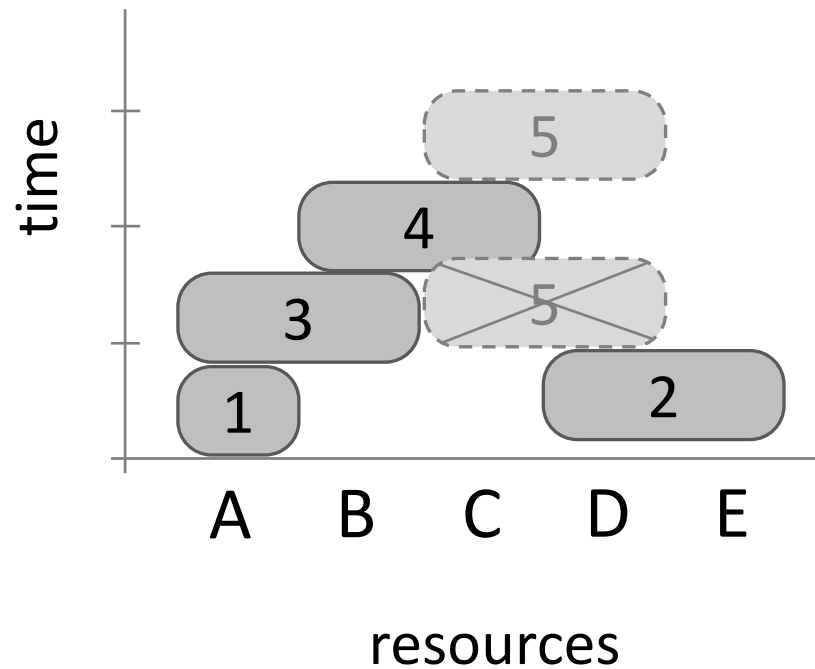
# Request Timing Problem

---



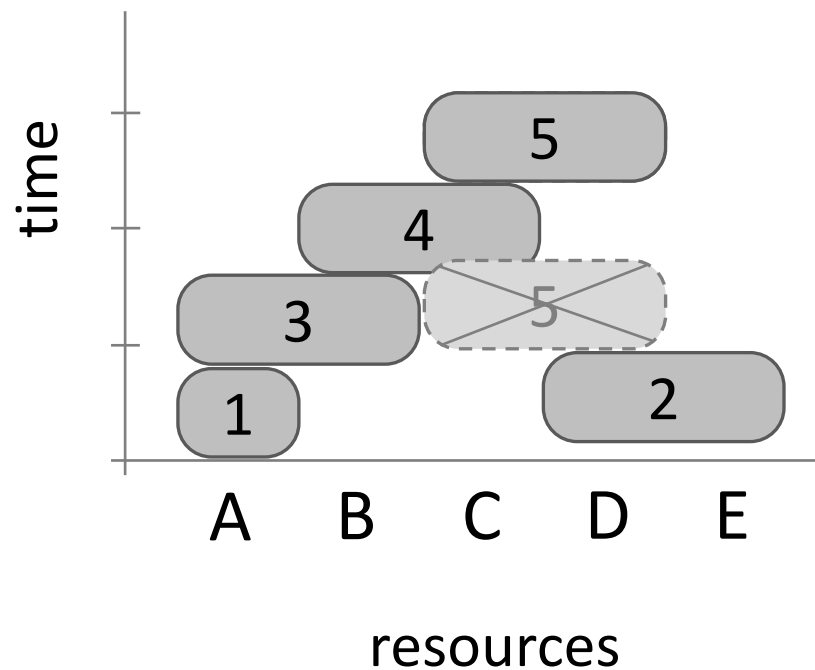
# Request Timing Problem

---



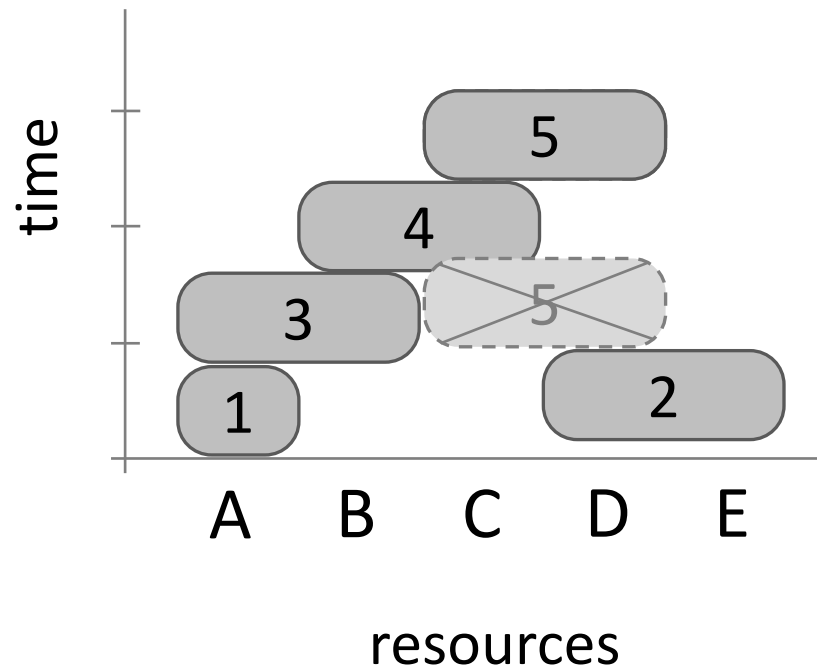
# Request Timing Problem

---



# Request Timing Problem

Timing variations cause “gaps” in underlying queues



# New Approach: Concurrency Groups

---

# New Approach: Concurrency Groups

---

- does not merely *prevent* resources from being accessed concurrently

# New Approach: Concurrency Groups

---

- does not merely *prevent* resources from being accessed concurrently
- instead safely *allows* concurrency

# New Approach: Concurrency Groups

---

## Concurrency Group Locking Protocol (CGLP)

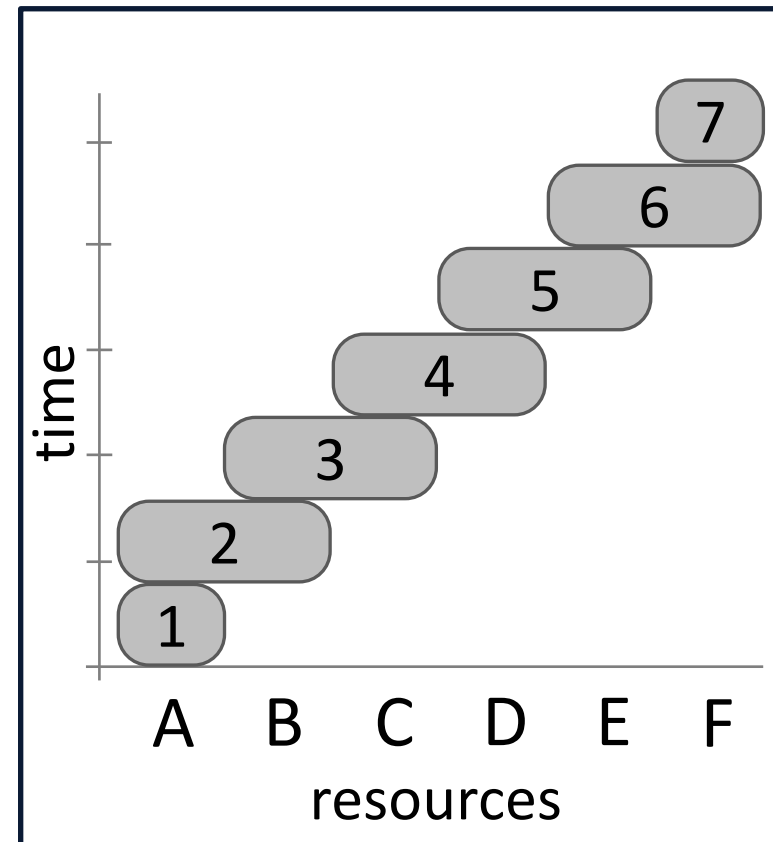
- does not merely *prevent* resources from being accessed concurrently
- instead safely *allows* concurrency



# New Approach: Concurrency Groups

## Concurrency Group Locking Protocol (CGLP)

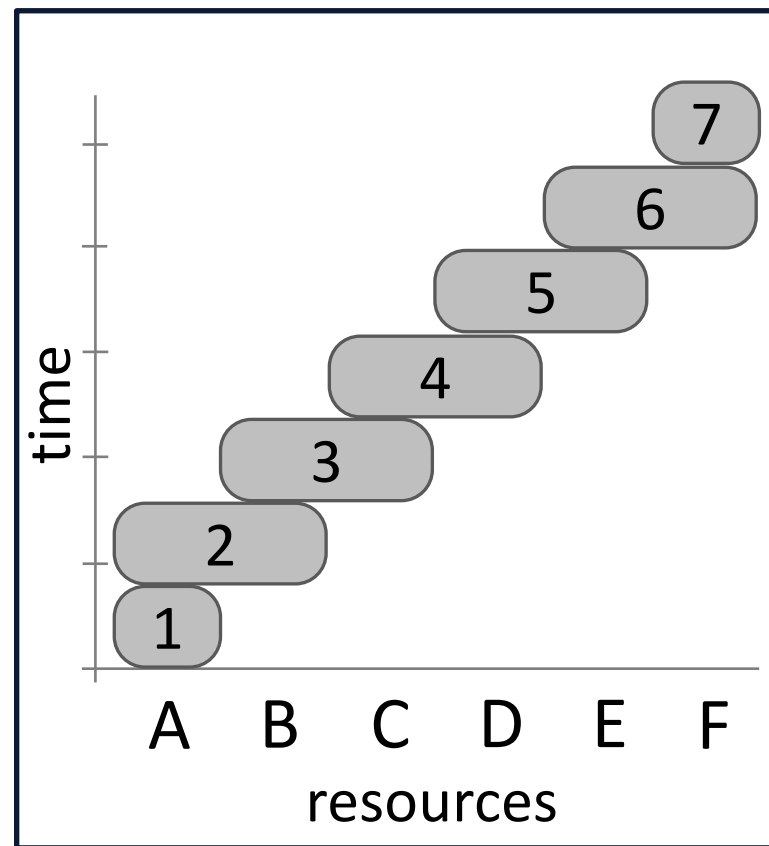
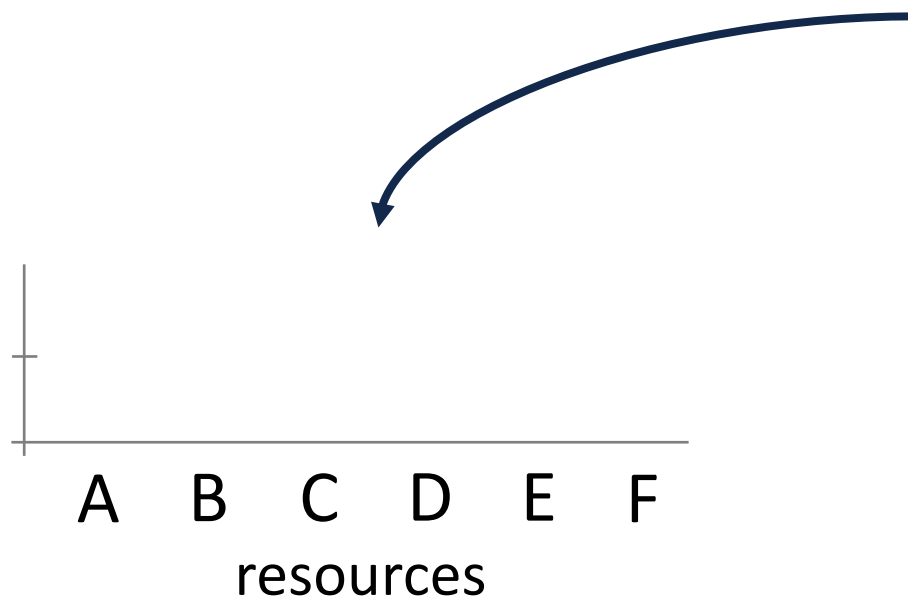
- does not merely *prevent* resources from being accessed concurrently
- instead safely *allows* concurrency



# New Approach: Concurrency Groups

## Concurrency Group Locking Protocol (CGLP)

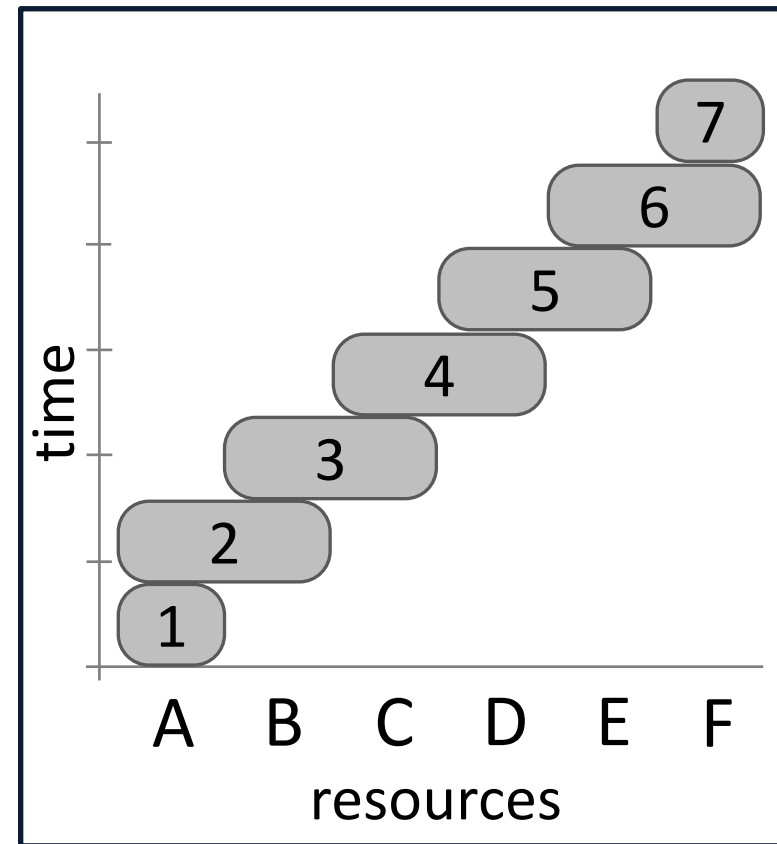
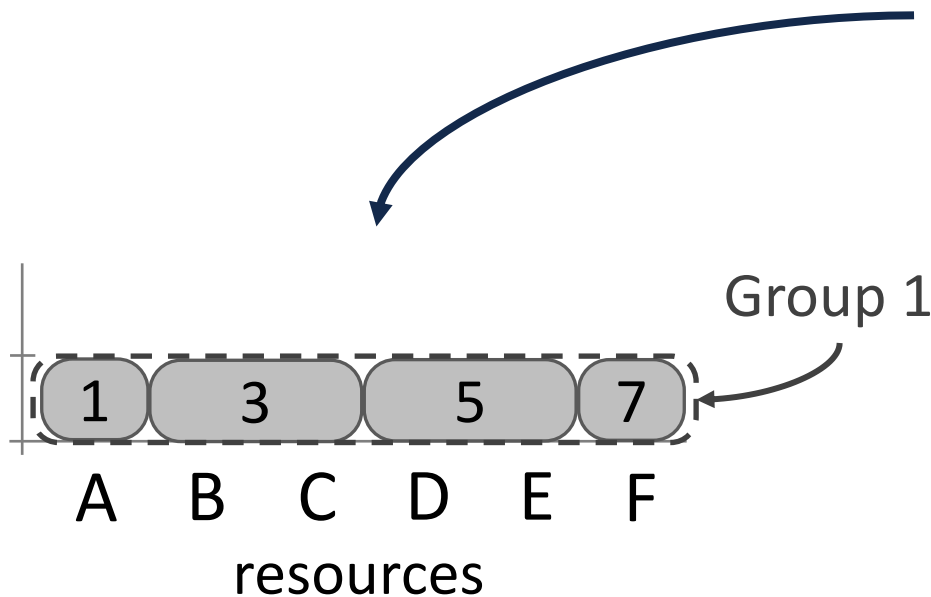
- does not merely *prevent* resources from being accessed concurrently
- instead safely *allows* concurrency



# New Approach: Concurrency Groups

## Concurrency Group Locking Protocol (CGLP)

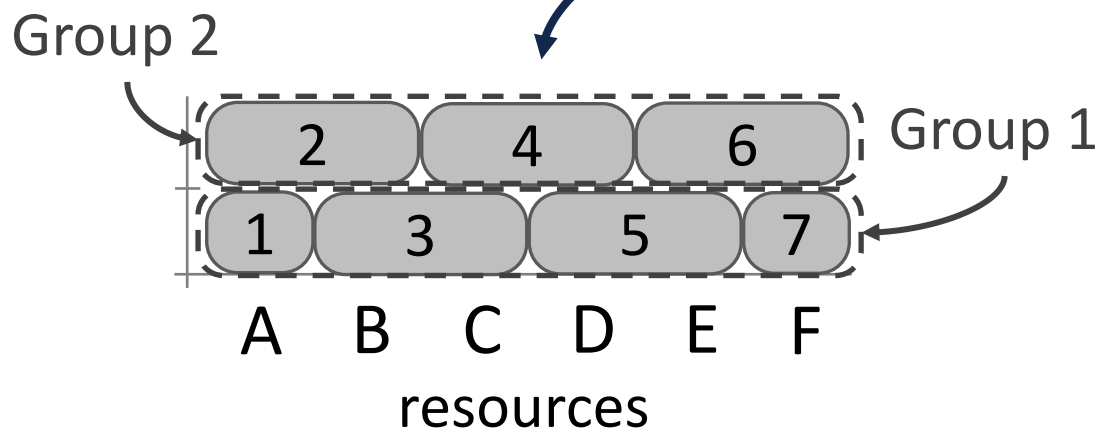
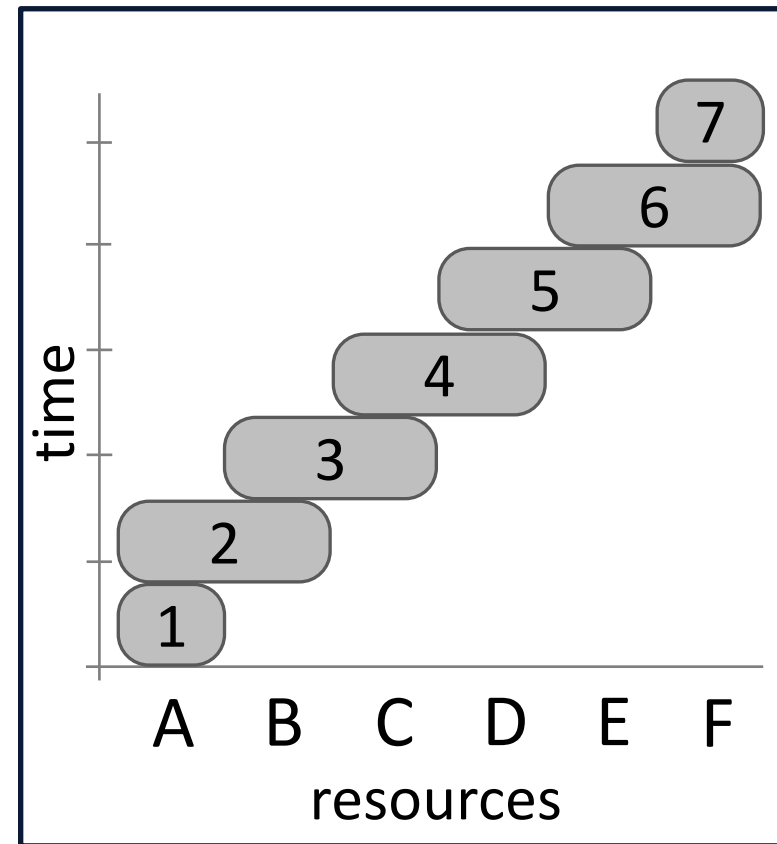
- does not merely *prevent* resources from being accessed concurrently
- instead safely *allows* concurrency



# New Approach: Concurrency Groups

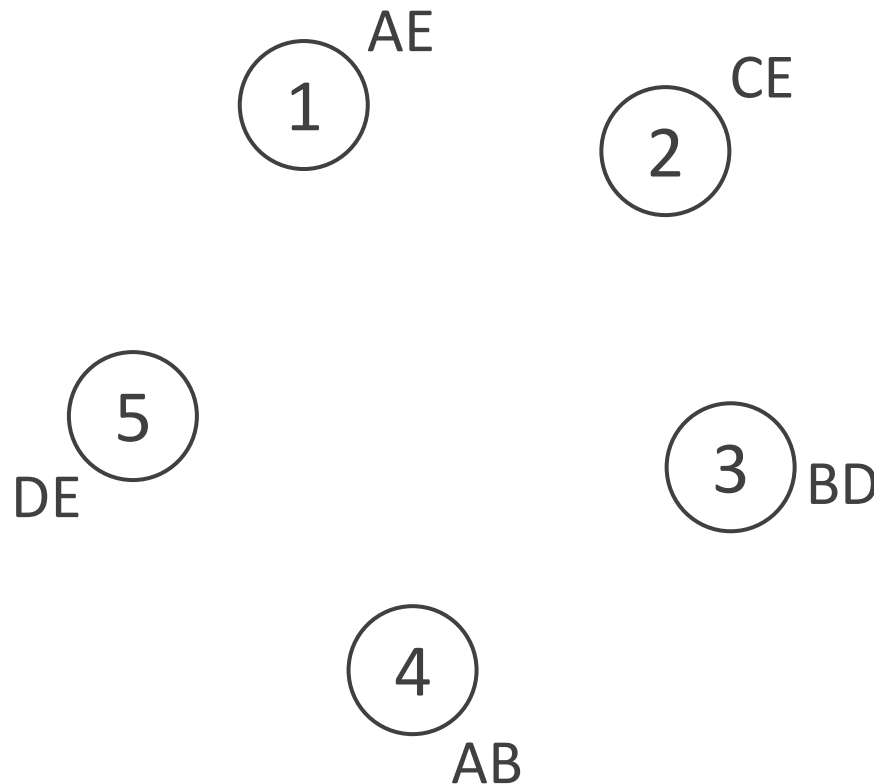
## Concurrency Group Locking Protocol (CGLP)

- does not merely *prevent* resources from being accessed concurrently
- instead safely *allows* concurrency



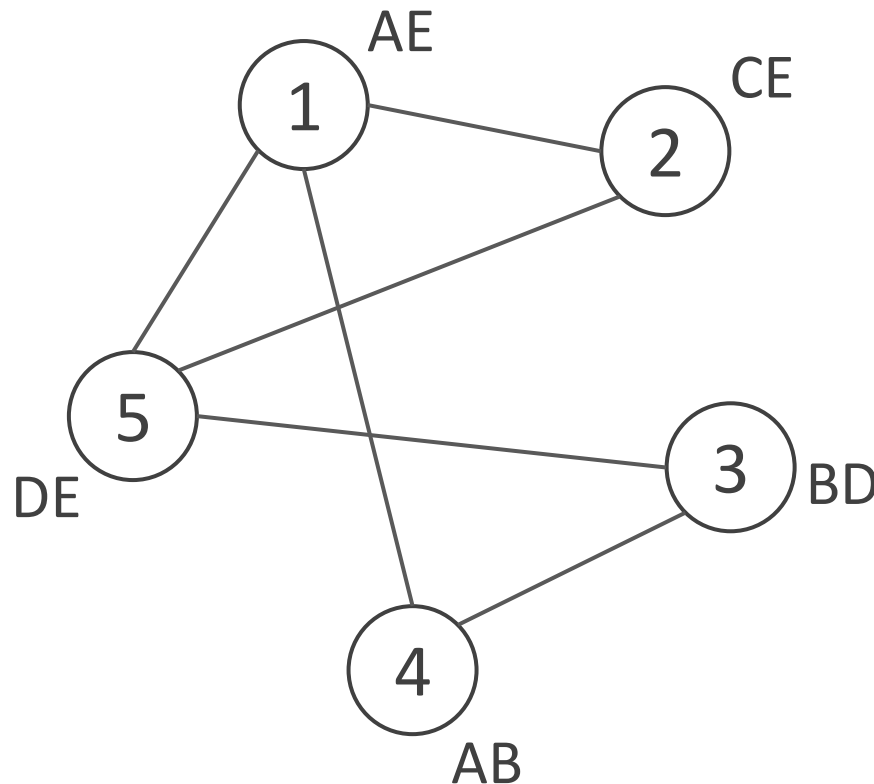
# Offline Group Formation

## Concurrency Group Locking Protocol (CGLP)



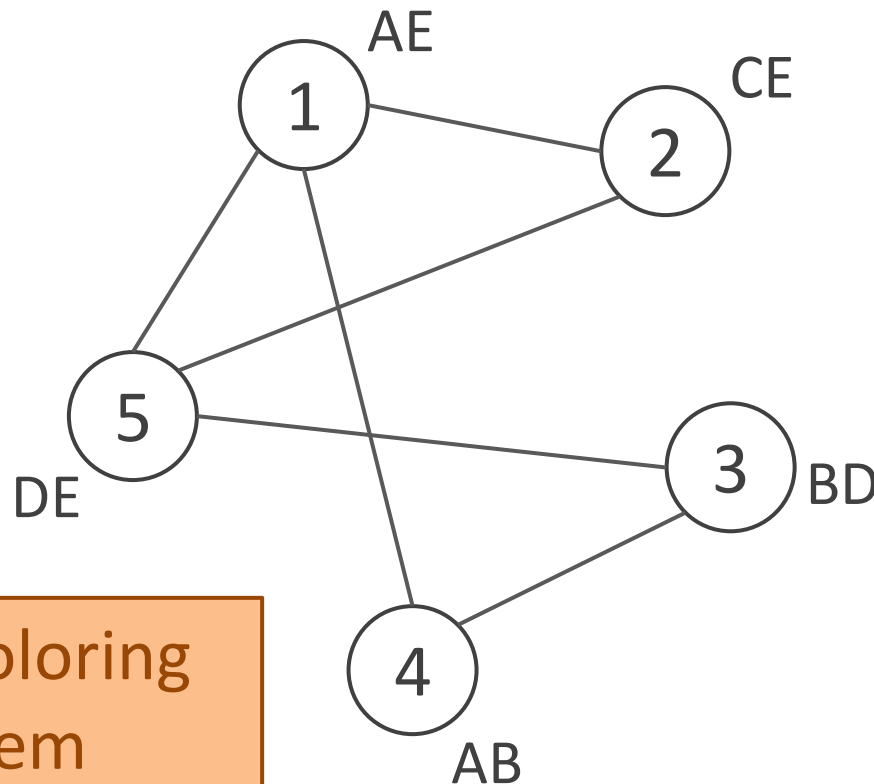
# Offline Group Formation

Concurrency Group Locking Protocol (CGLP)



# Offline Group Formation

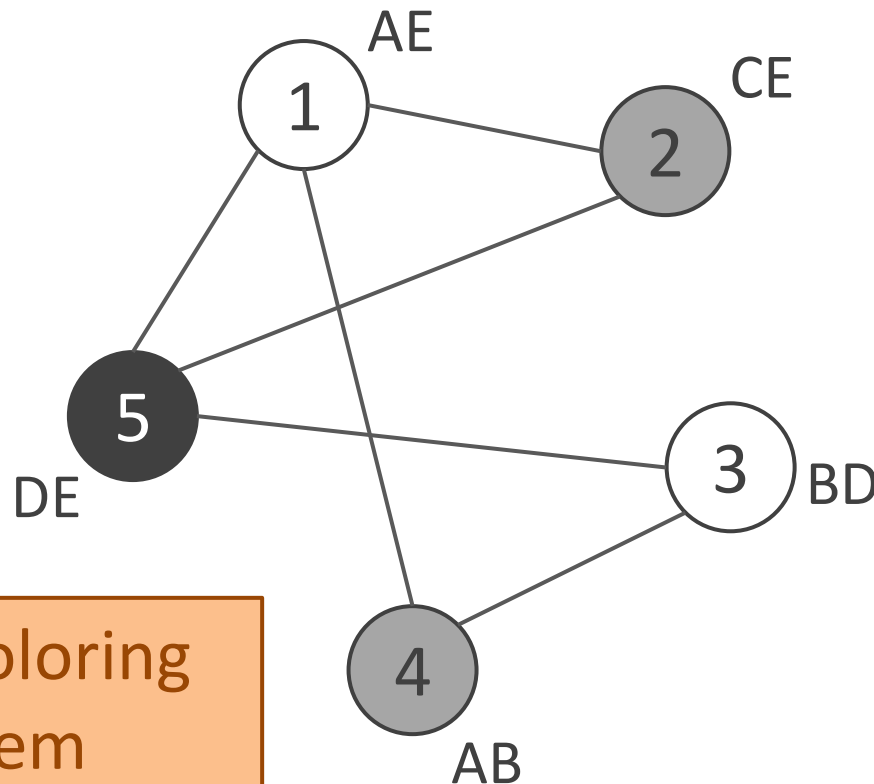
Concurrency Group Locking Protocol (CGLP)



Vertex Coloring  
Problem

# Offline Group Formation

Concurrency Group Locking Protocol (CGLP)

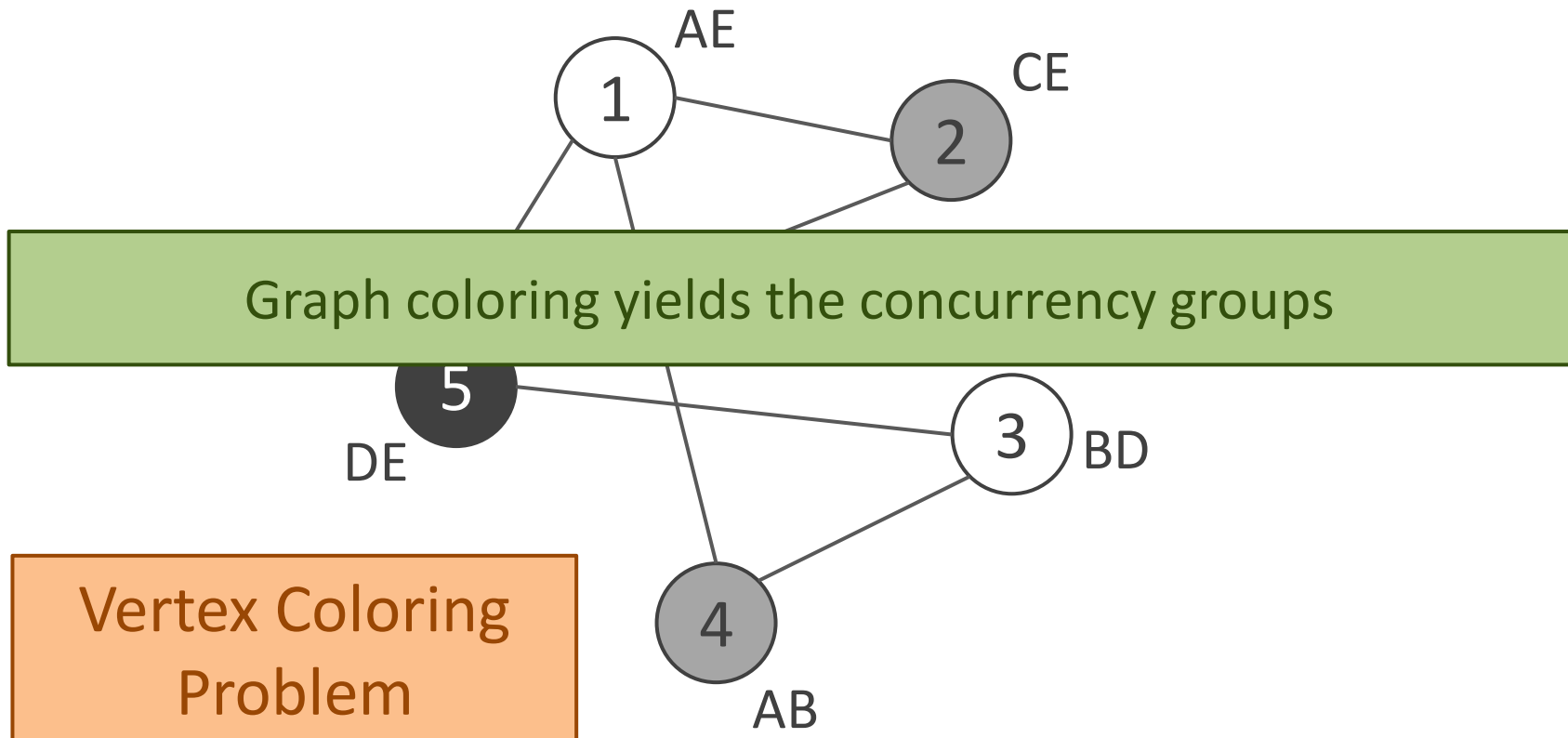


Vertex Coloring  
Problem

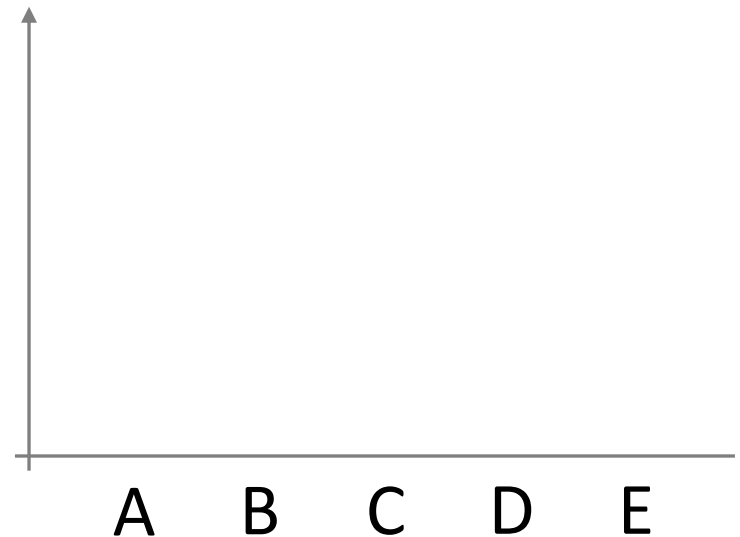
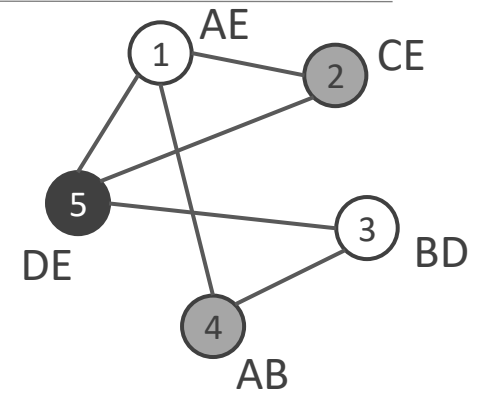
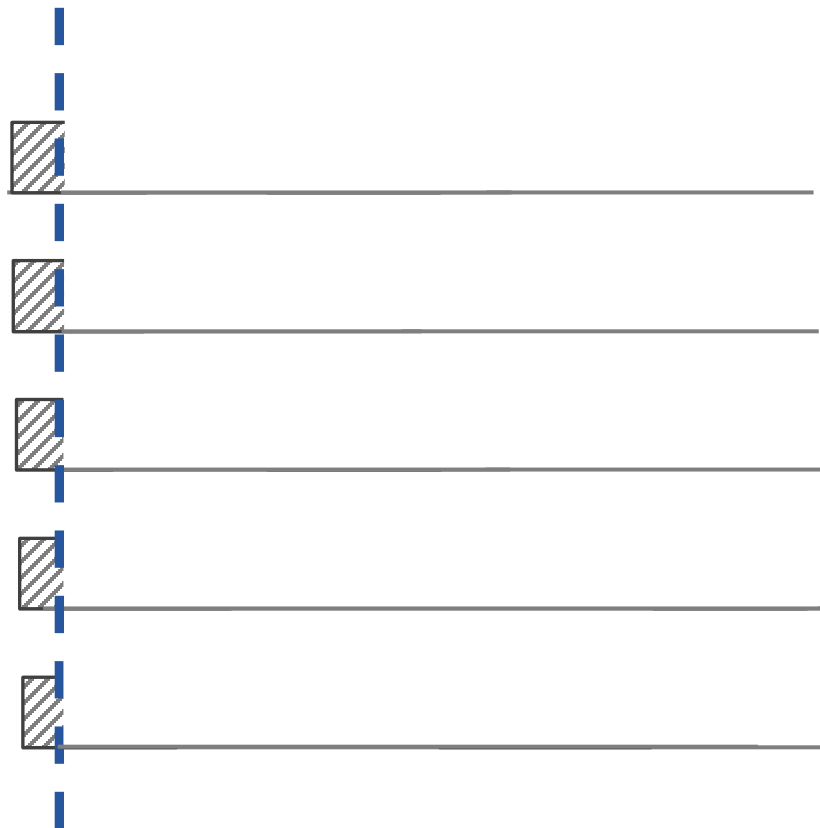


# Offline Group Formation

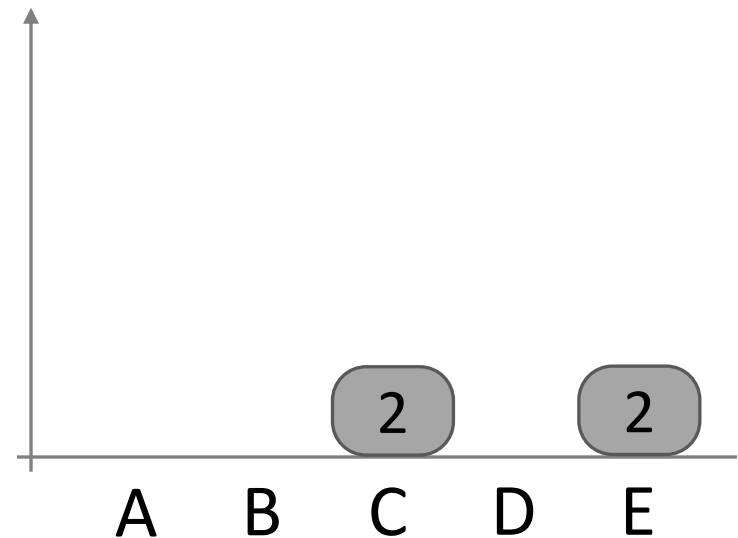
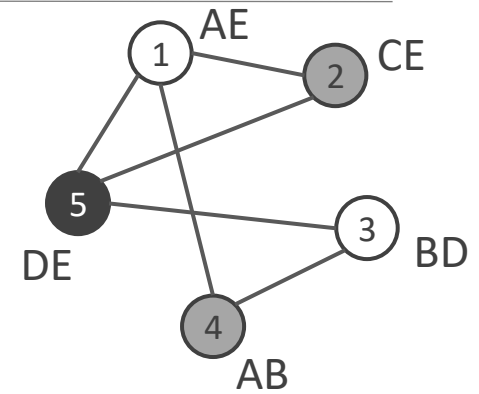
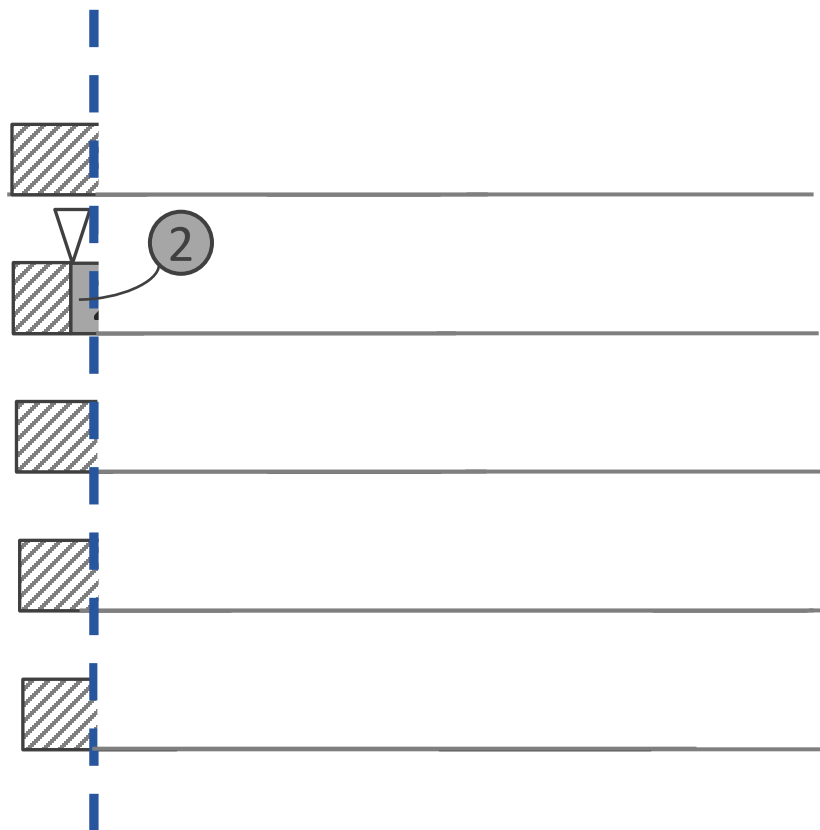
Concurrency Group Locking Protocol (CGLP)



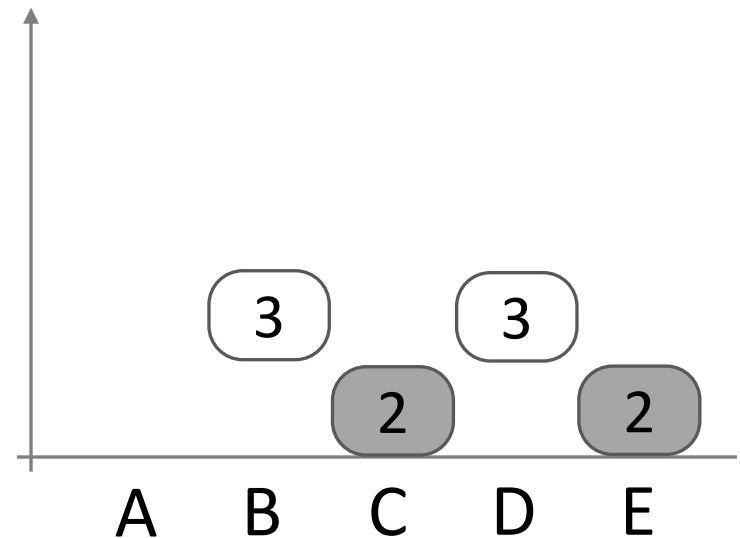
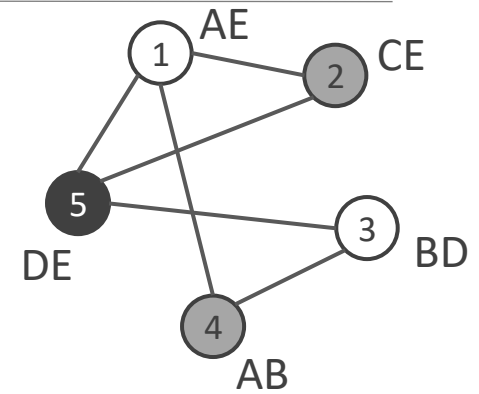
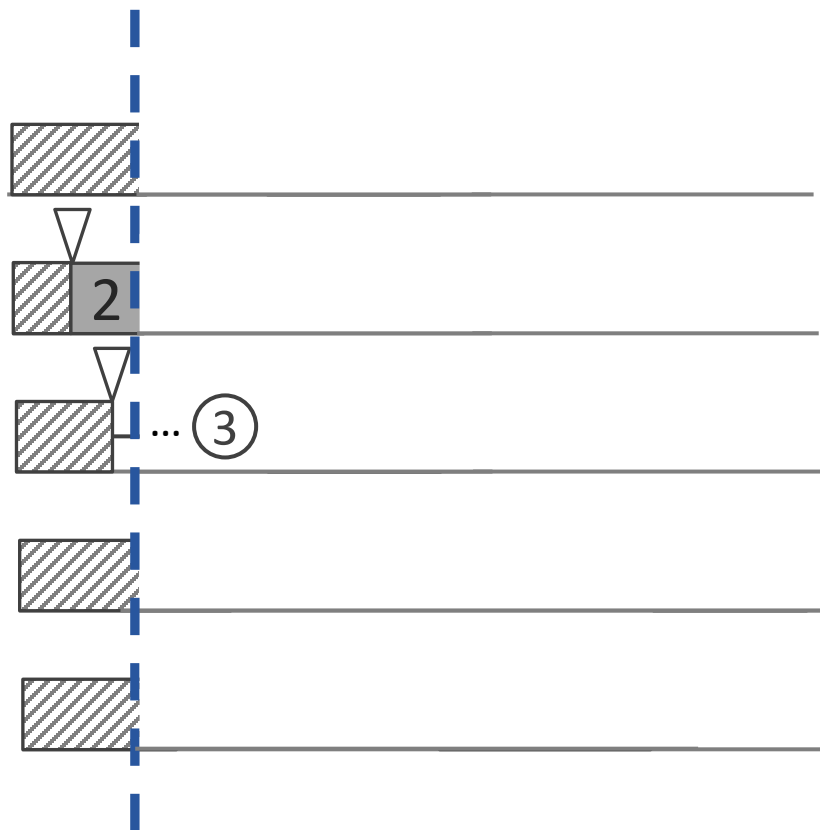
# Online Group Management



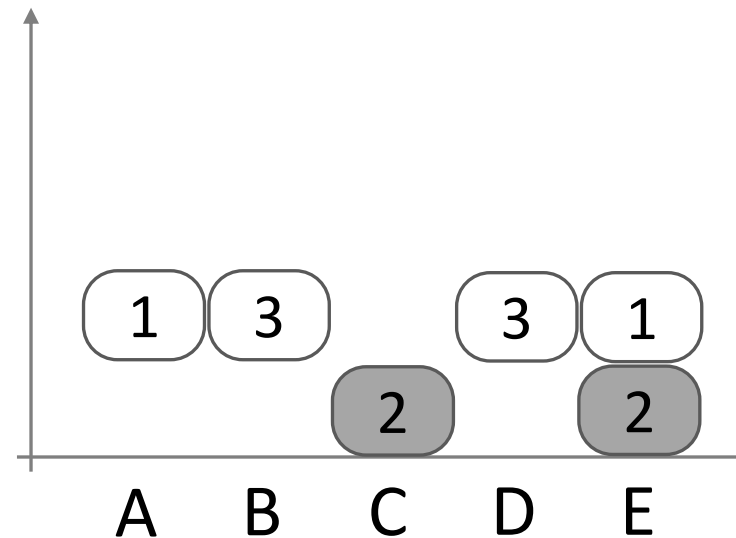
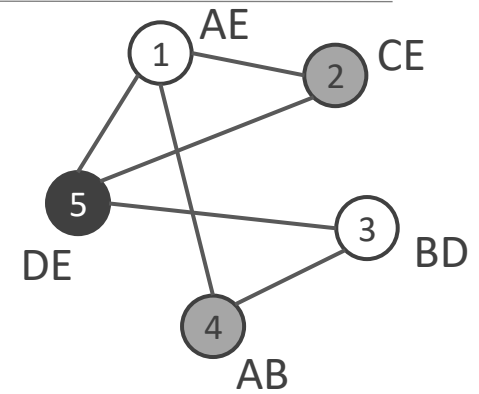
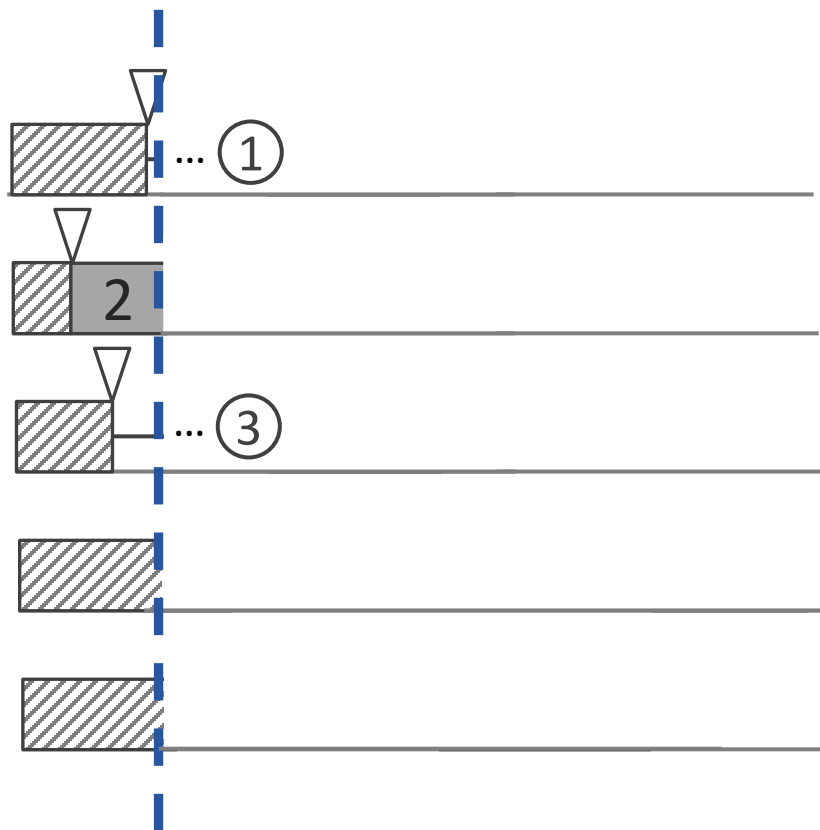
# Online Group Management



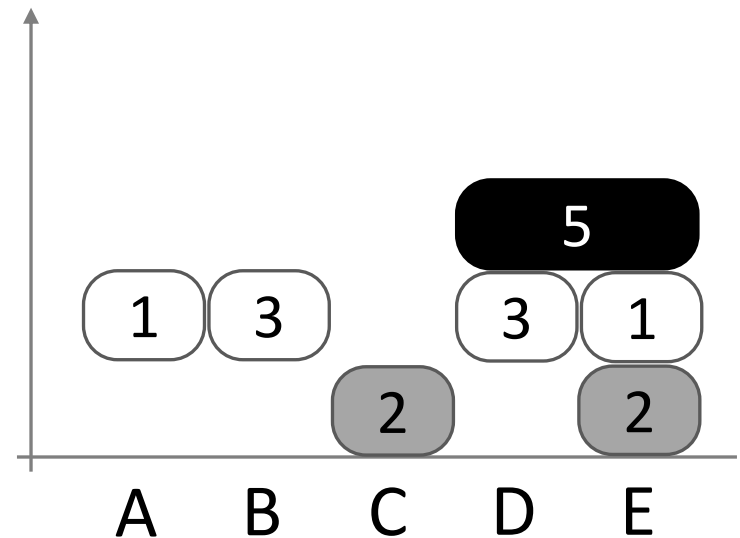
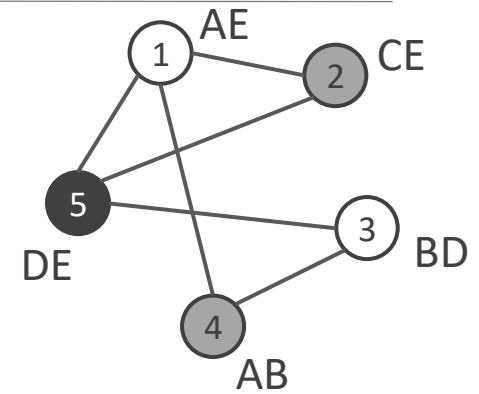
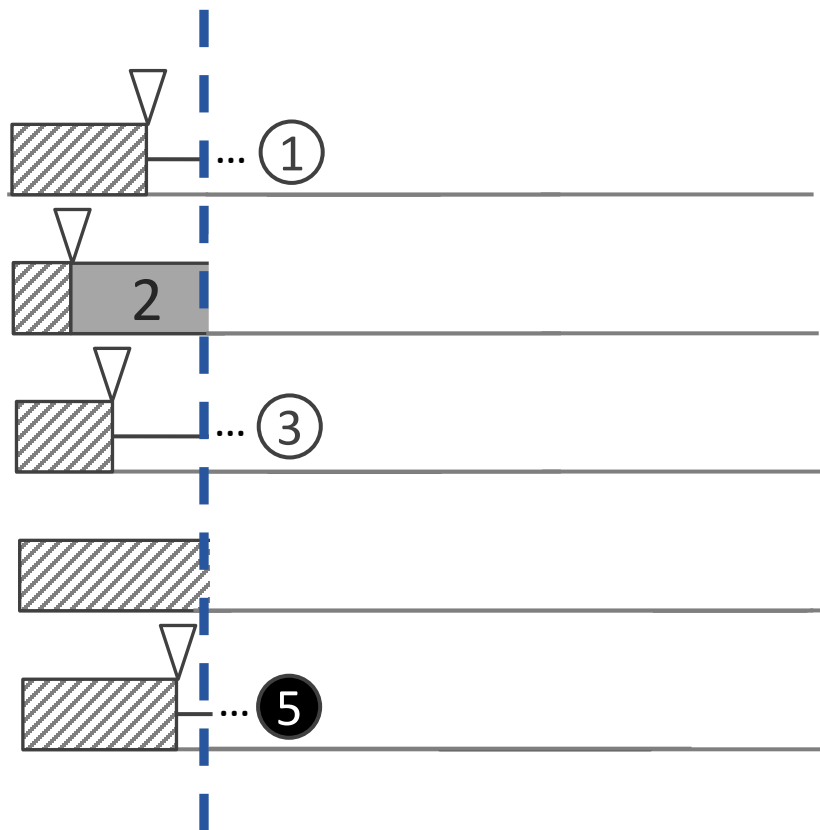
# Online Group Management



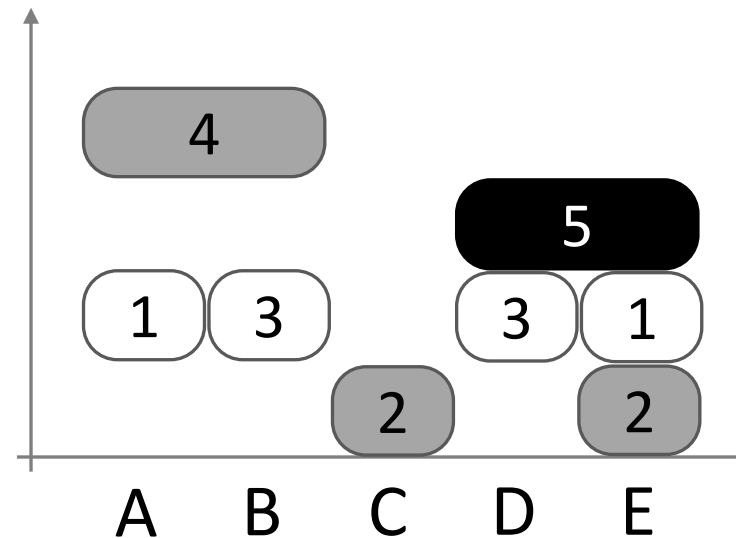
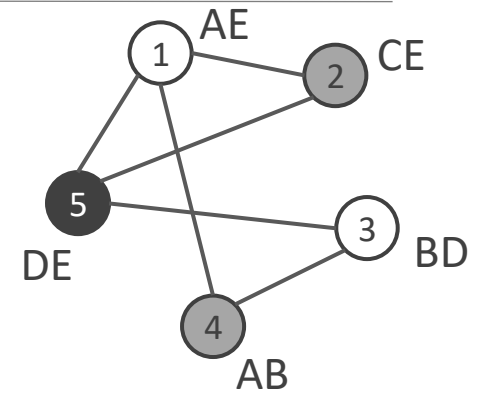
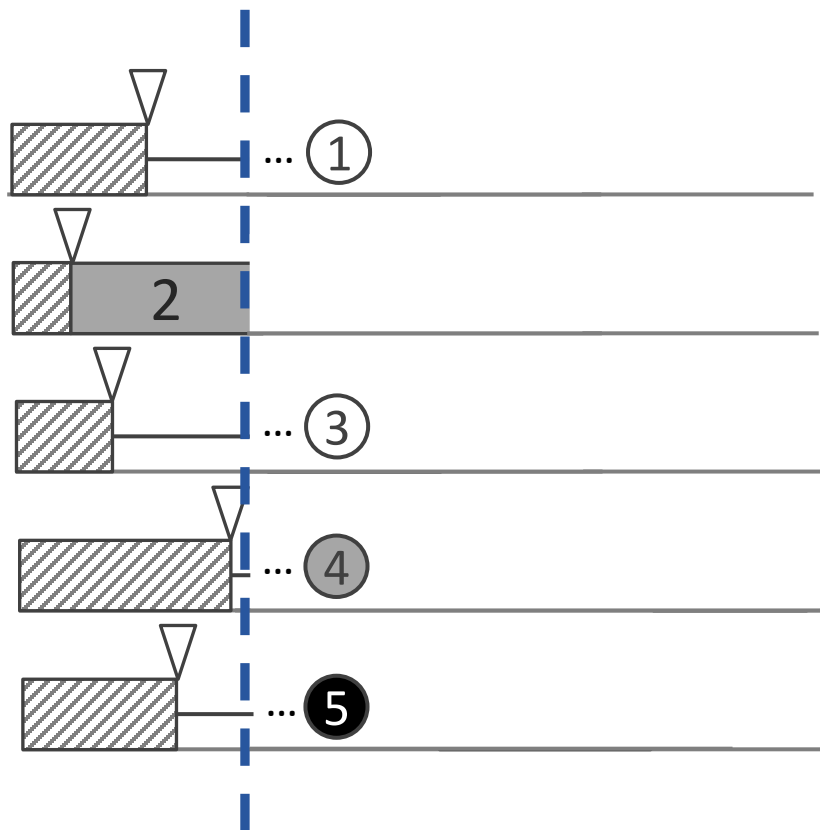
# Online Group Management



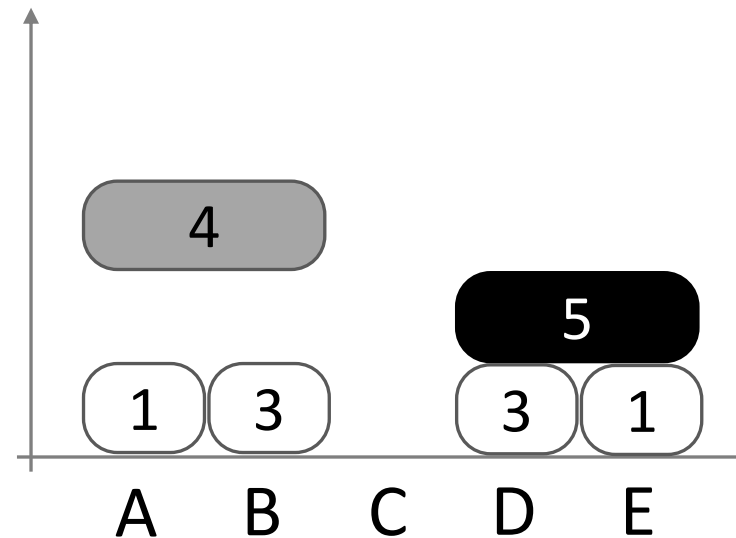
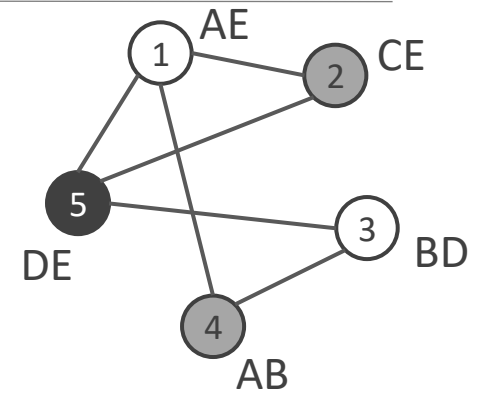
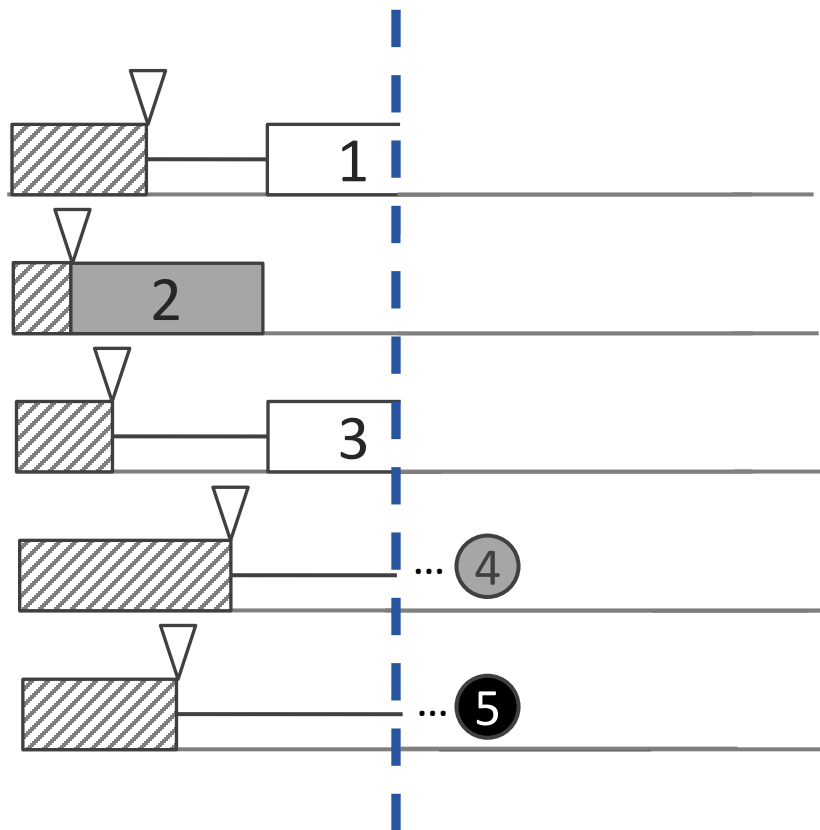
# Online Group Management



# Online Group Management

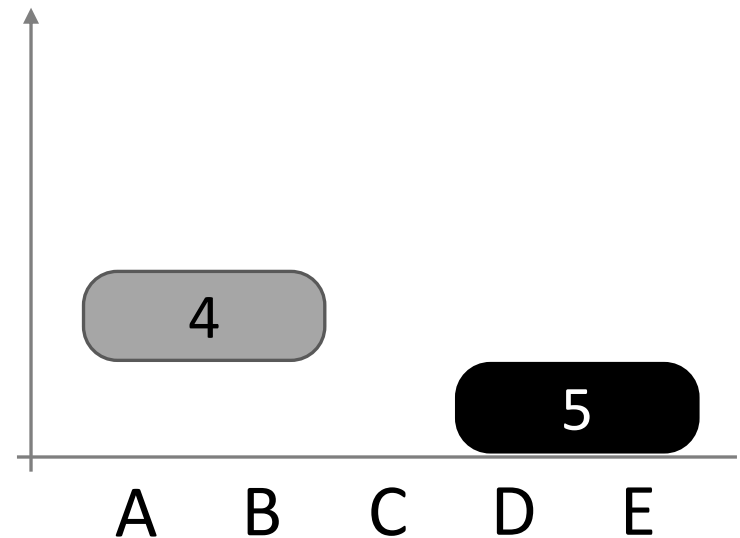
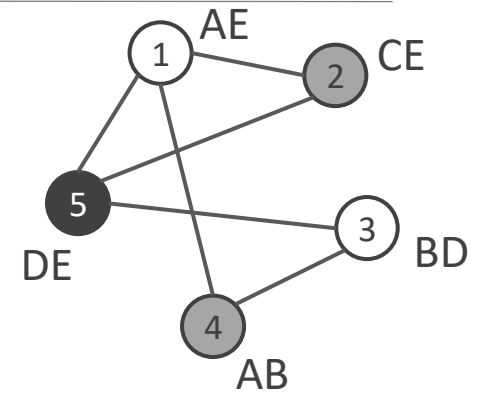
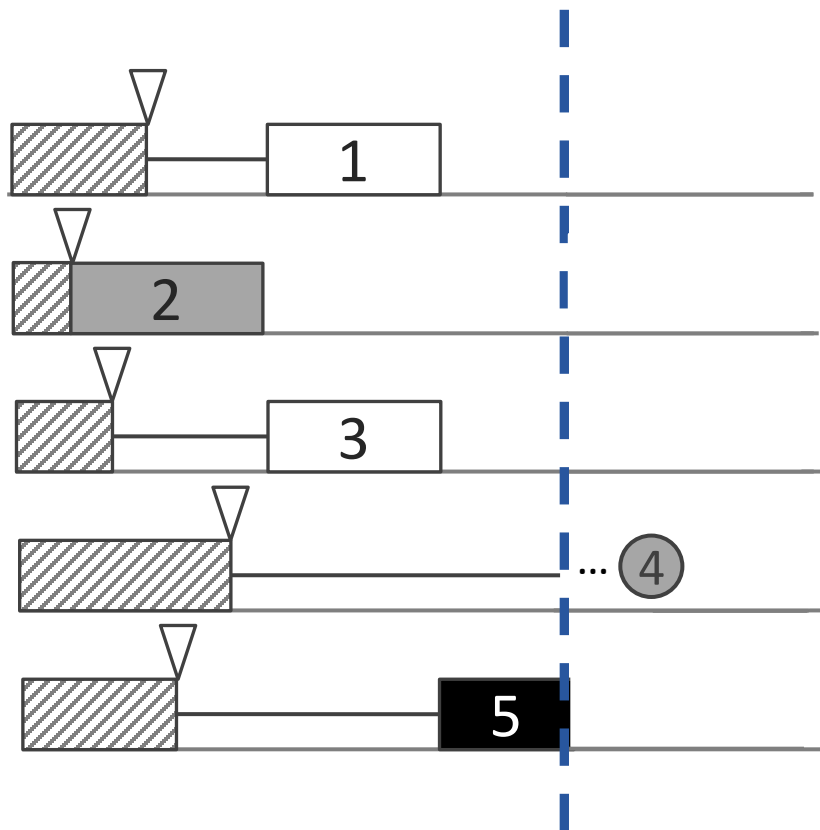


# Online Group Management

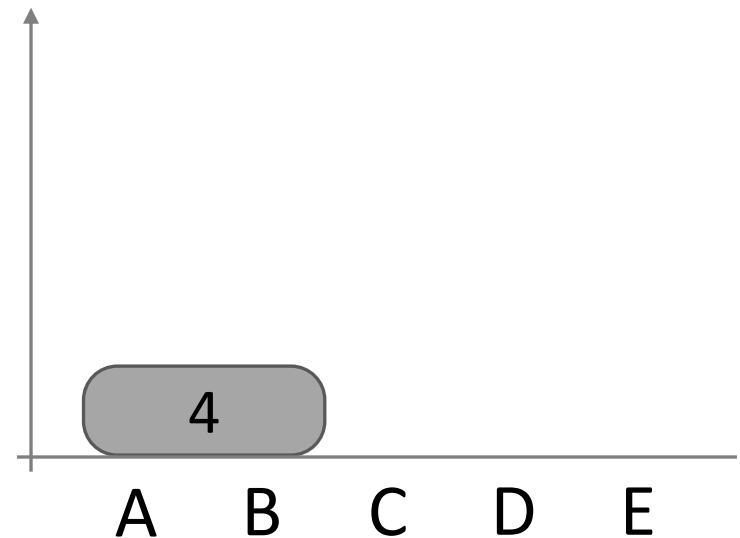
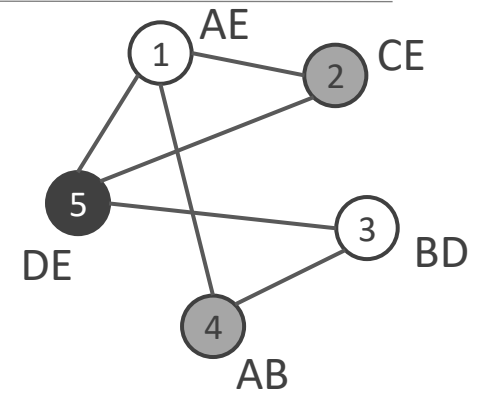
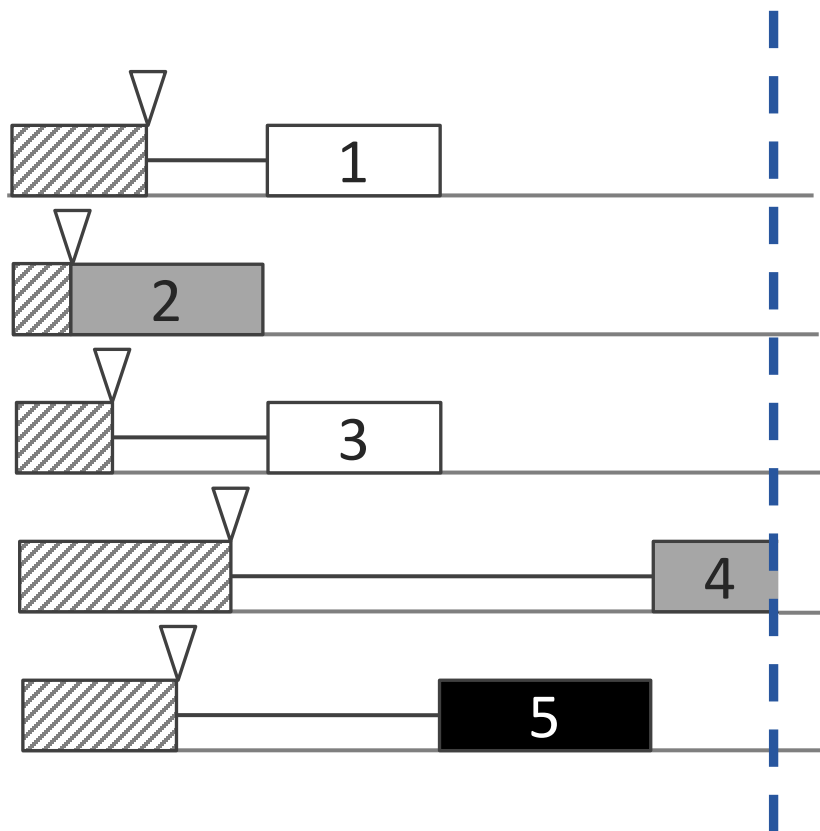




# Online Group Management



# Online Group Management



# Online Group Management

---

# Online Group Management

---

## $R^kLP$

- extension to existing protocols
- $k$ -phased reader locking protocol

# Online Group Management

---

## $R^kLP$

- extension to existing protocols
- k-phased reader locking protocol

Protocol	Overhead ( $\mu s$ )
MCS	0.7
RNLP	13.5
U-C-RNLP	13.0
G-C-RNLP	15.1
CGLP	3.1

# Online Group Management

---

## $R^kLP$

- extension to existing protocols
- k-phased reader locking protocol

Protocol	Overhead ( $\mu s$ )
MCS	0.7
RNLP	13.5
U-C-RNLP	13.0
G-C-RNLP	15.1
CGLP	3.1

# Online Group Management

---

## $R^kLP$

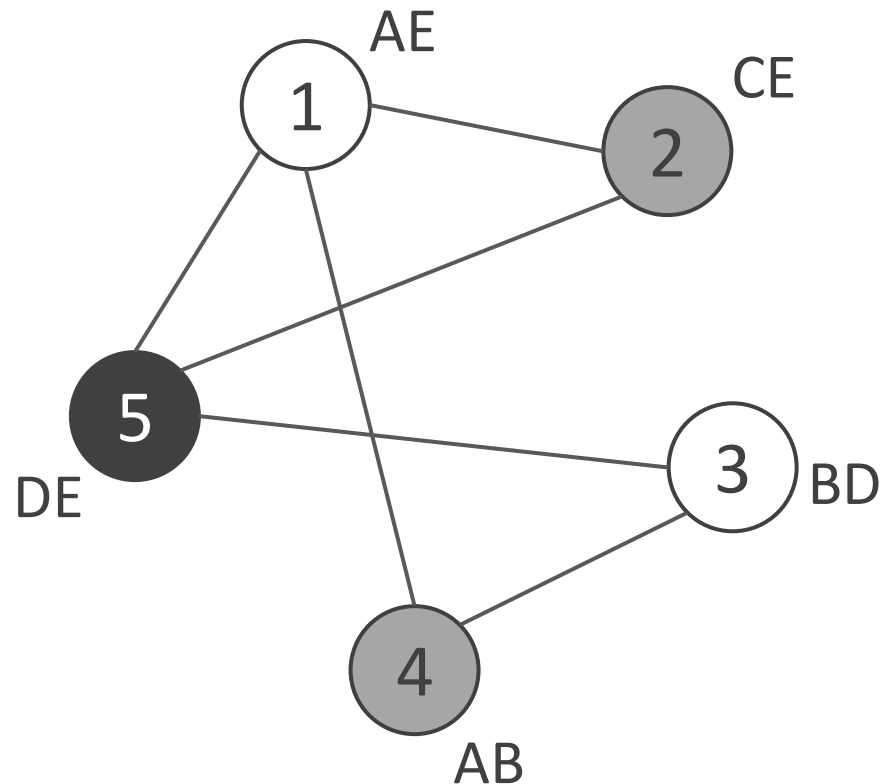
- extension to existing protocols
- k-phased reader locking protocol

Protocol	Overhead ( $\mu s$ )
MCS	0.7
RNLP	13.5
U-C-RNLP	13.0
G-C-RNLP	15.1
CGLP	3.1

Online synchronization with low overhead

# Alternate Coloring Choices

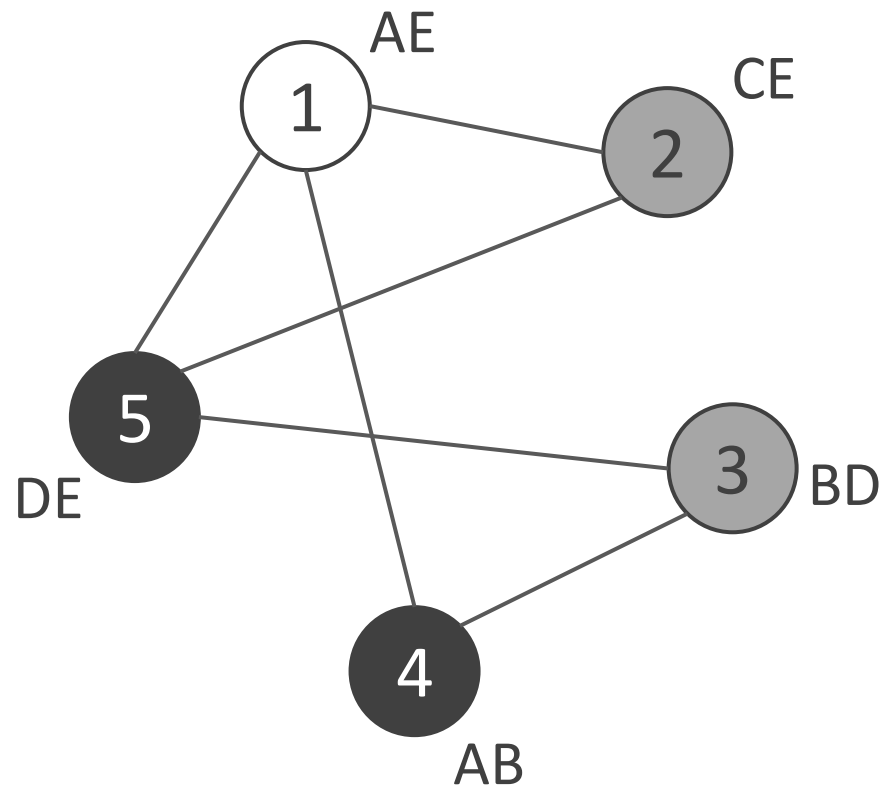
---





# Alternate Coloring Choices

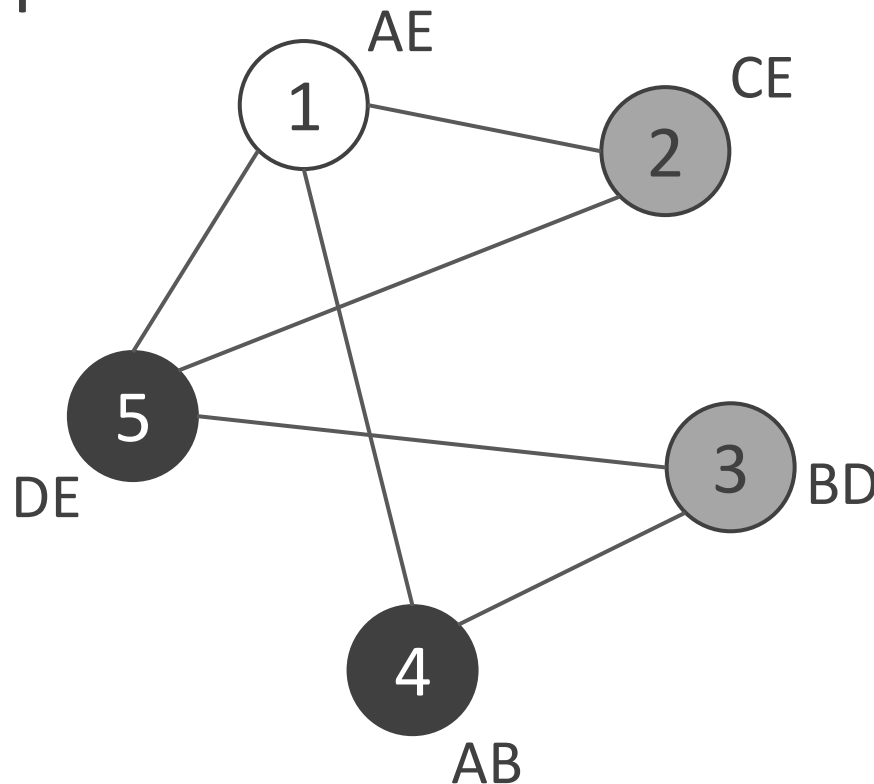
---



# Alternate Coloring Choices

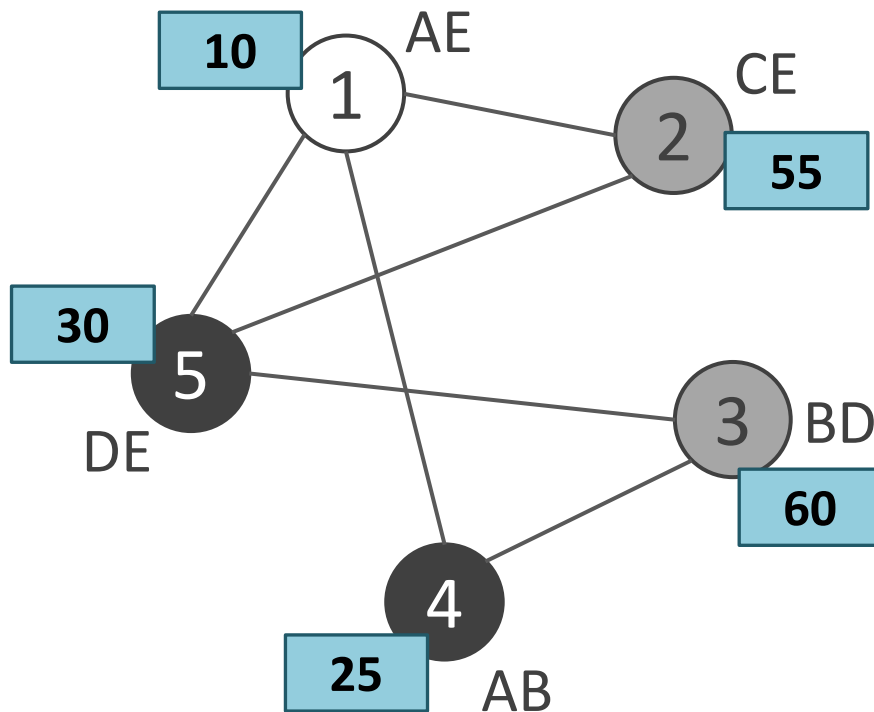
---

**Blocking:** sum of longest resource access in each group



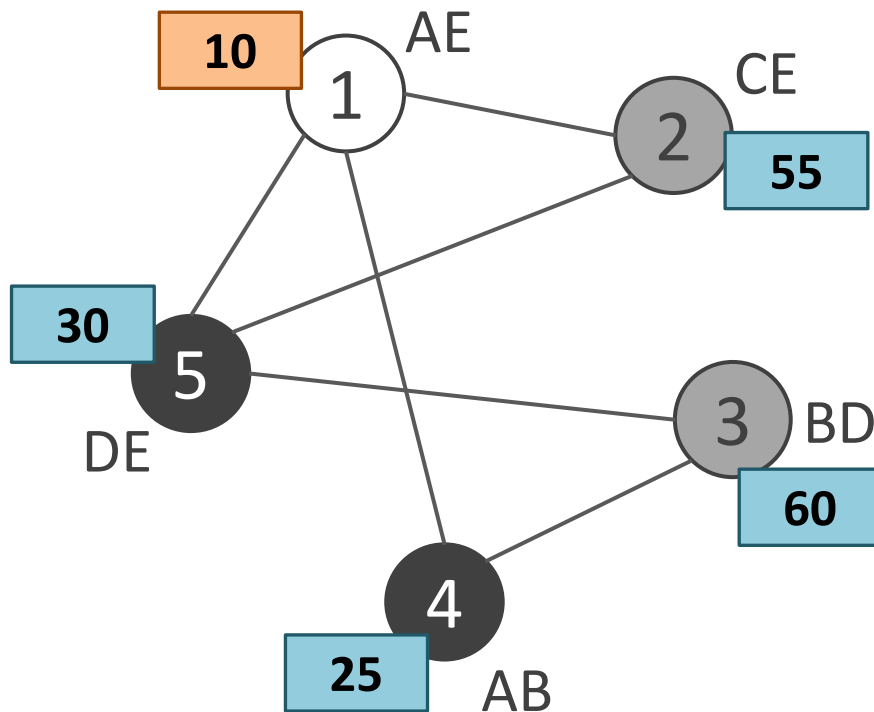
# Alternate Coloring Choices

**Blocking:** sum of longest resource access in each group



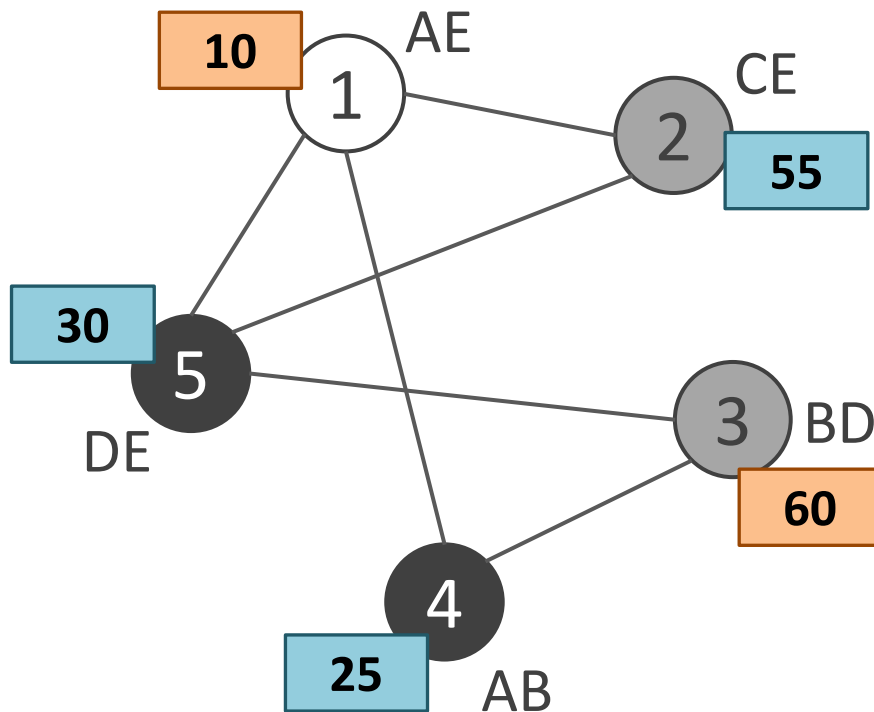
# Alternate Coloring Choices

**Blocking:** sum of longest resource access in each group



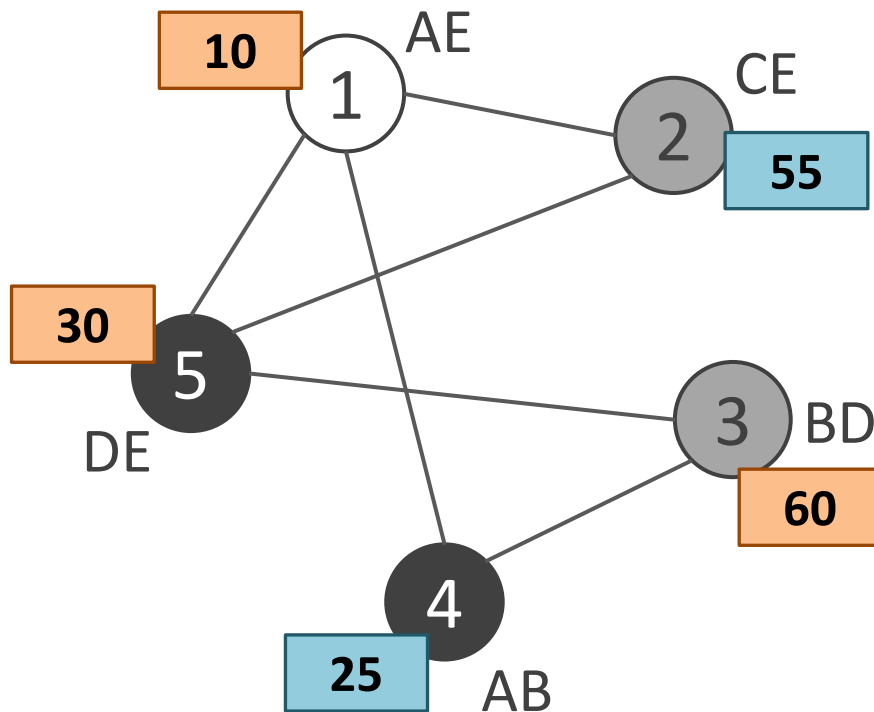
# Alternate Coloring Choices

**Blocking:** sum of longest resource access in each group



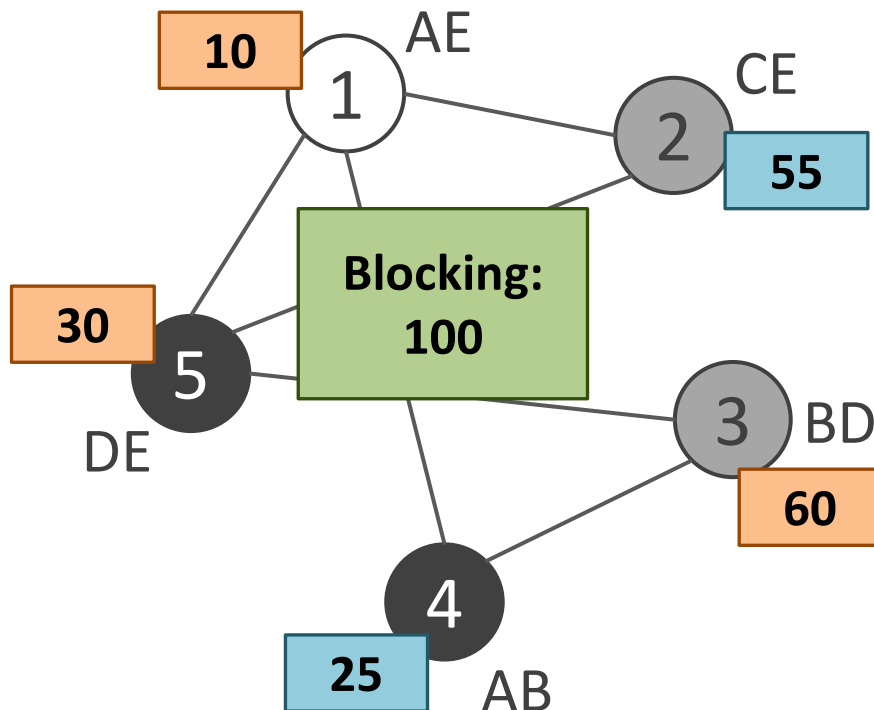
# Alternate Coloring Choices

**Blocking:** sum of longest resource access in each group



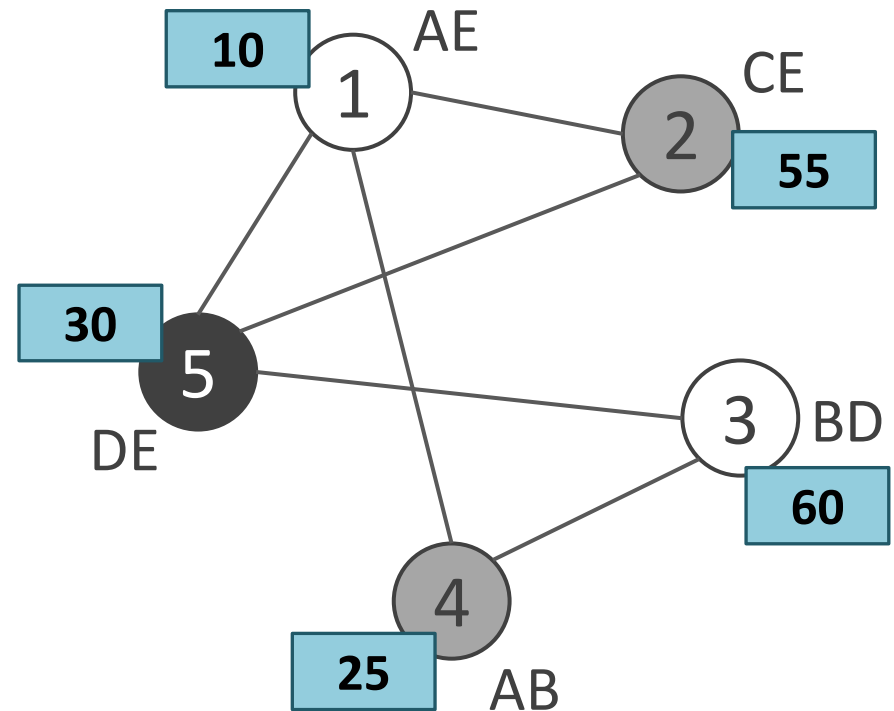
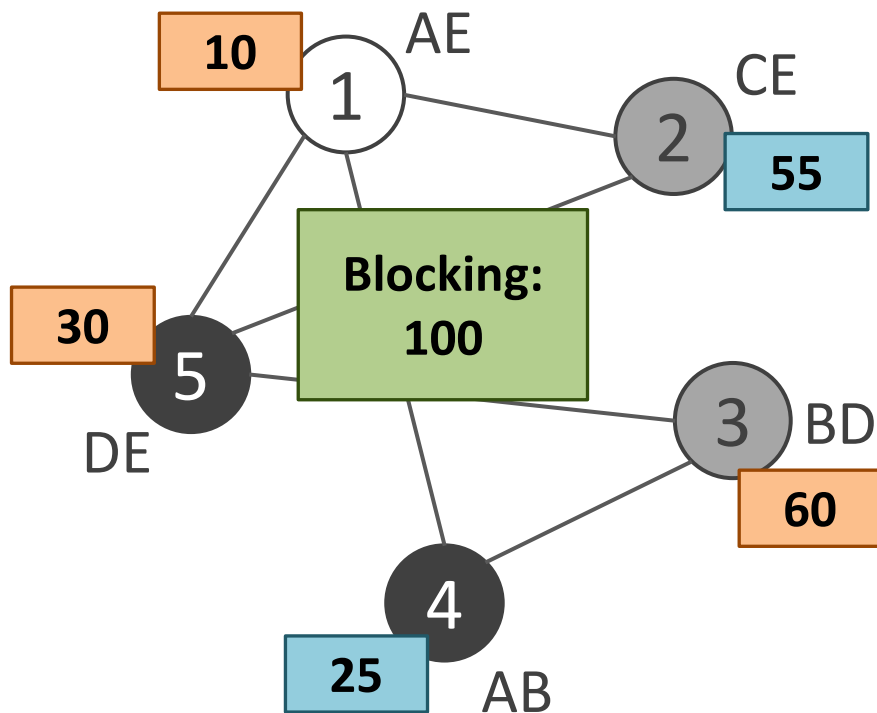
# Alternate Coloring Choices

**Blocking:** sum of longest resource access in each group



# Alternate Coloring Choices

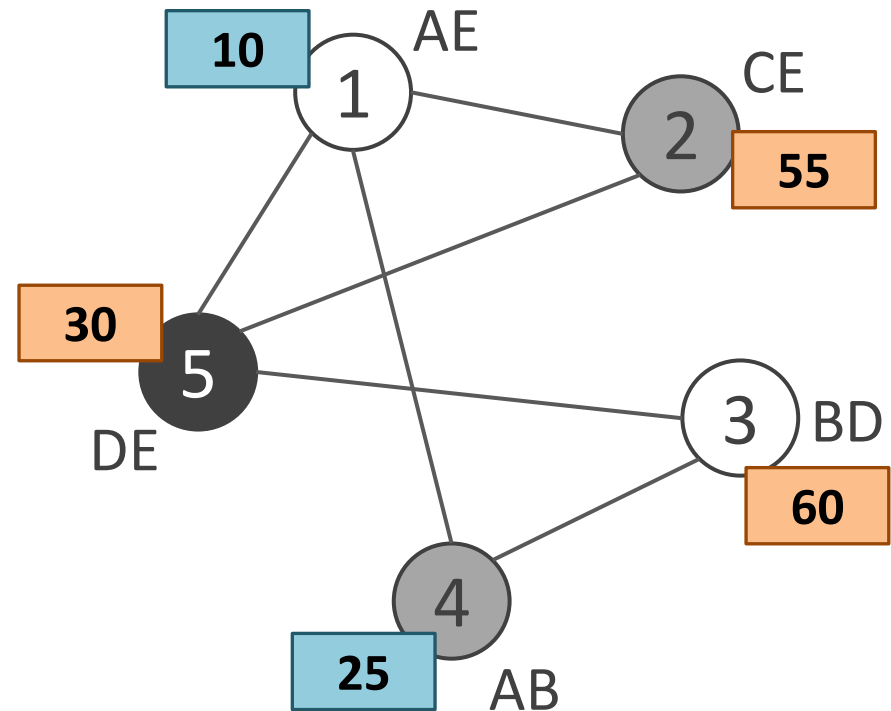
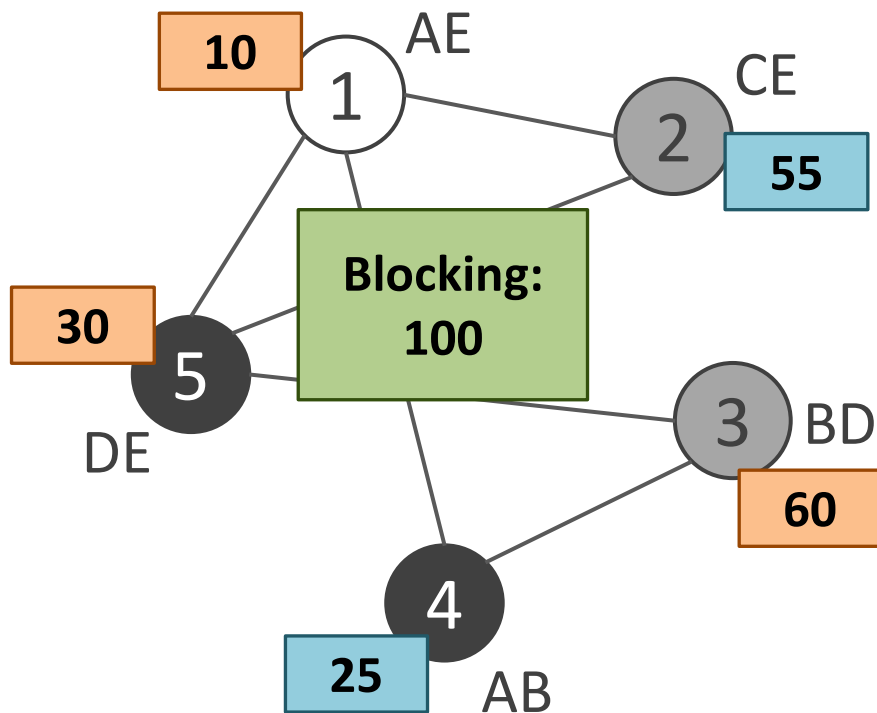
**Blocking:** sum of longest resource access in each group





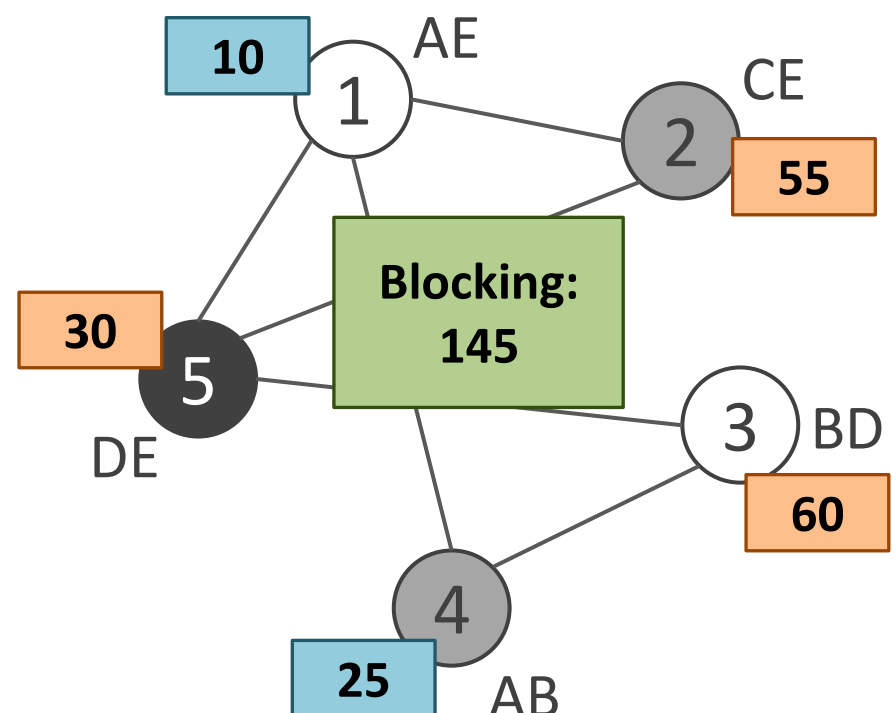
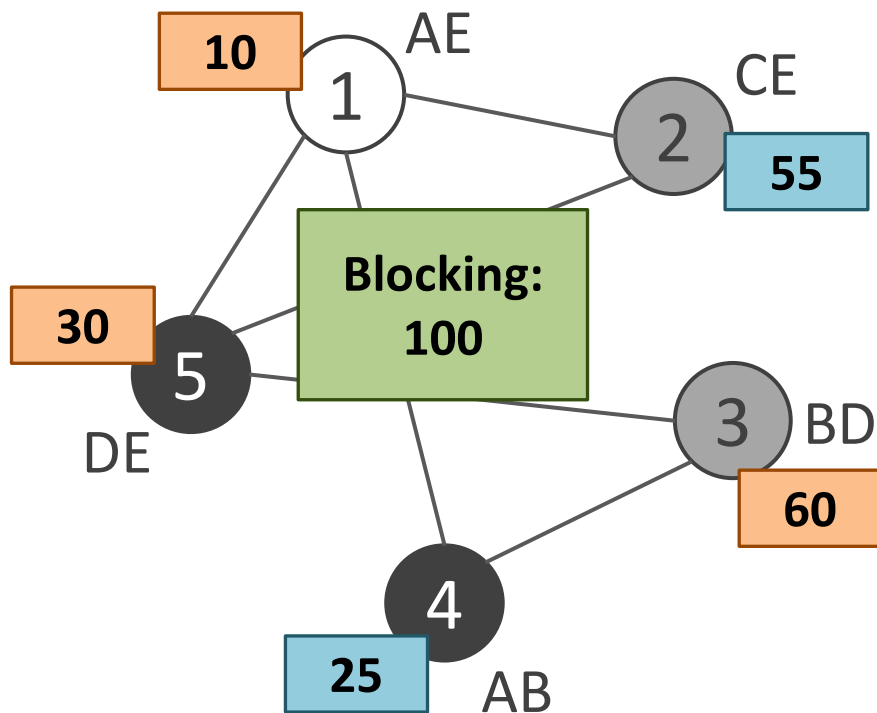
# Alternate Coloring Choices

**Blocking:** sum of longest resource access in each group



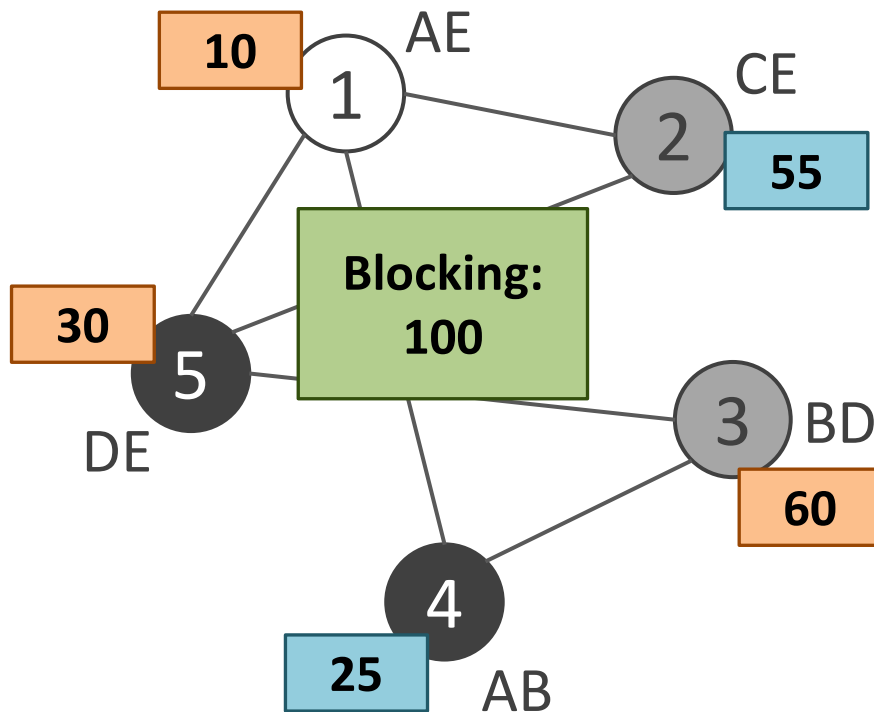
# Alternate Coloring Choices

**Blocking:** sum of longest resource access in each group



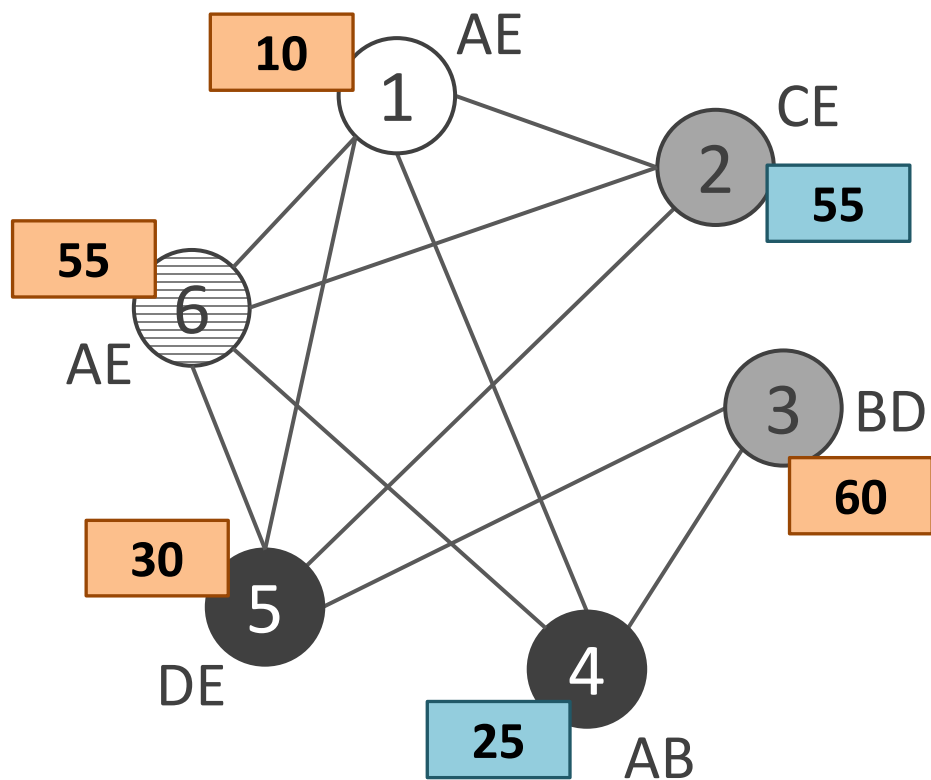
# Hierarchical Organization

---

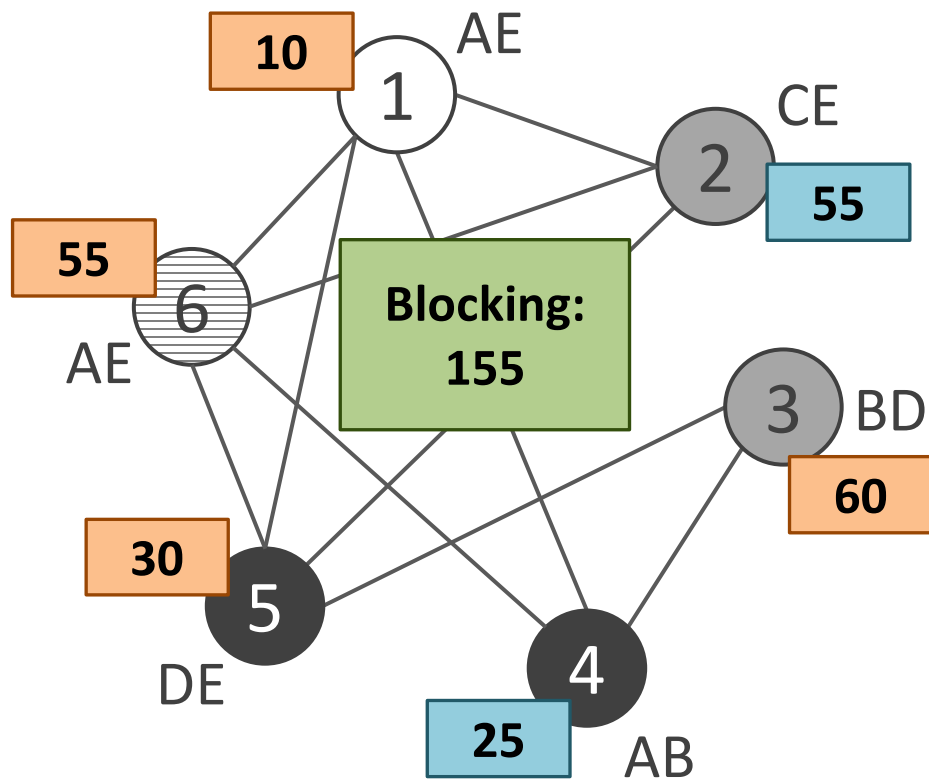


# Hierarchical Organization

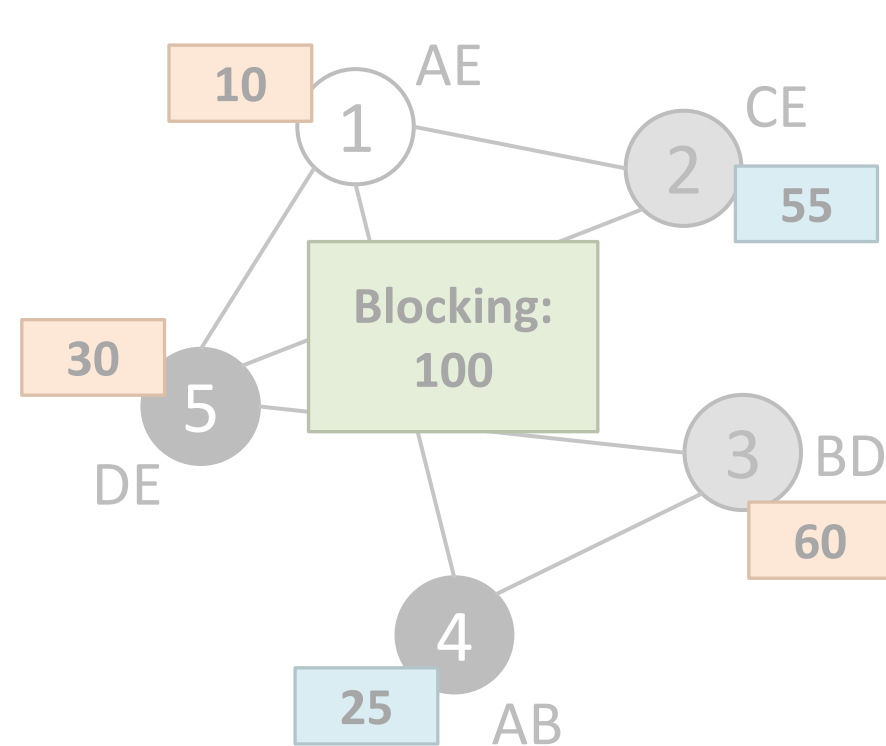
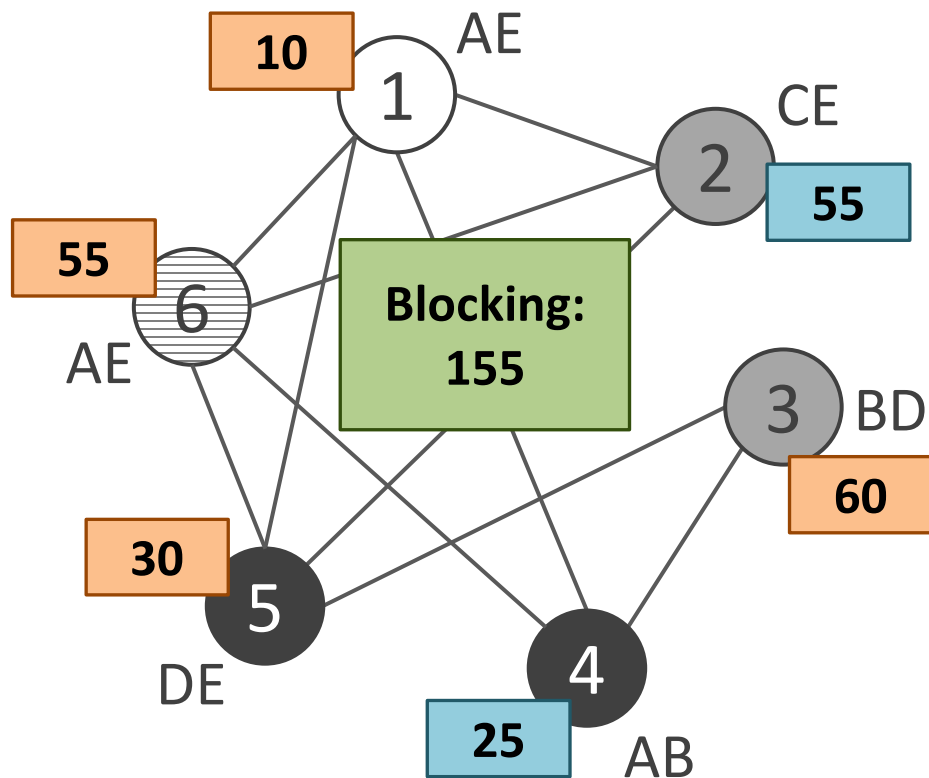
---



# Hierarchical Organization

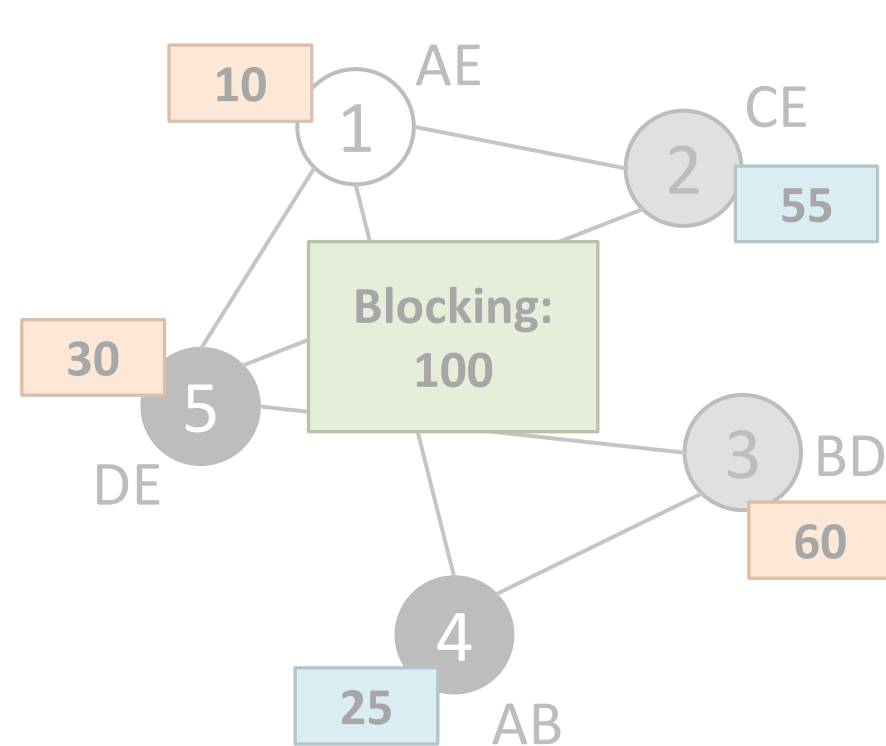
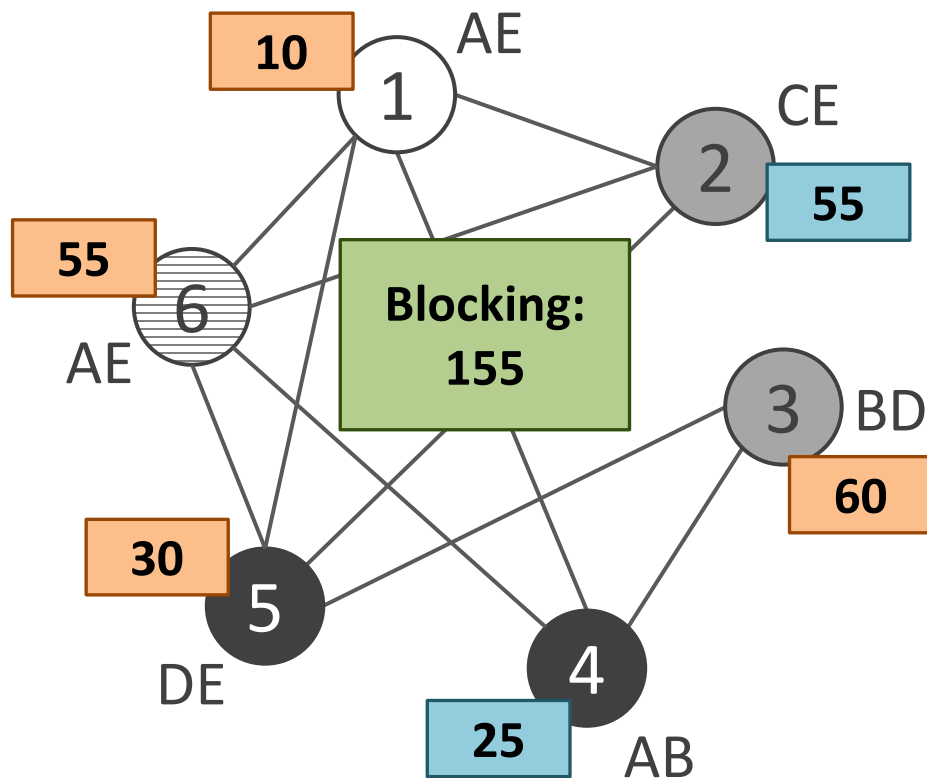


# Hierarchical Organization



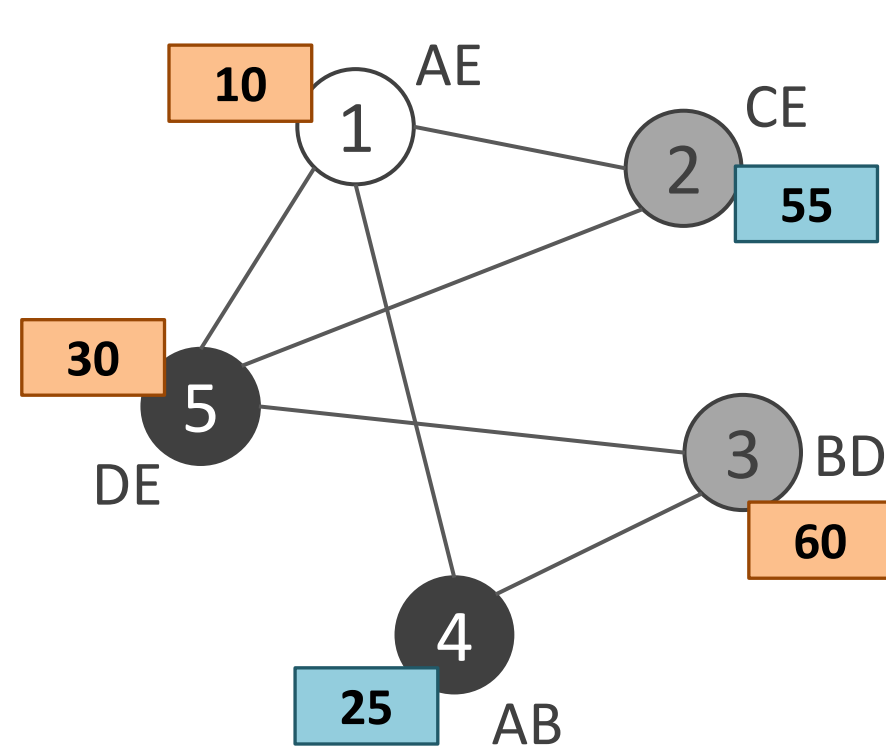
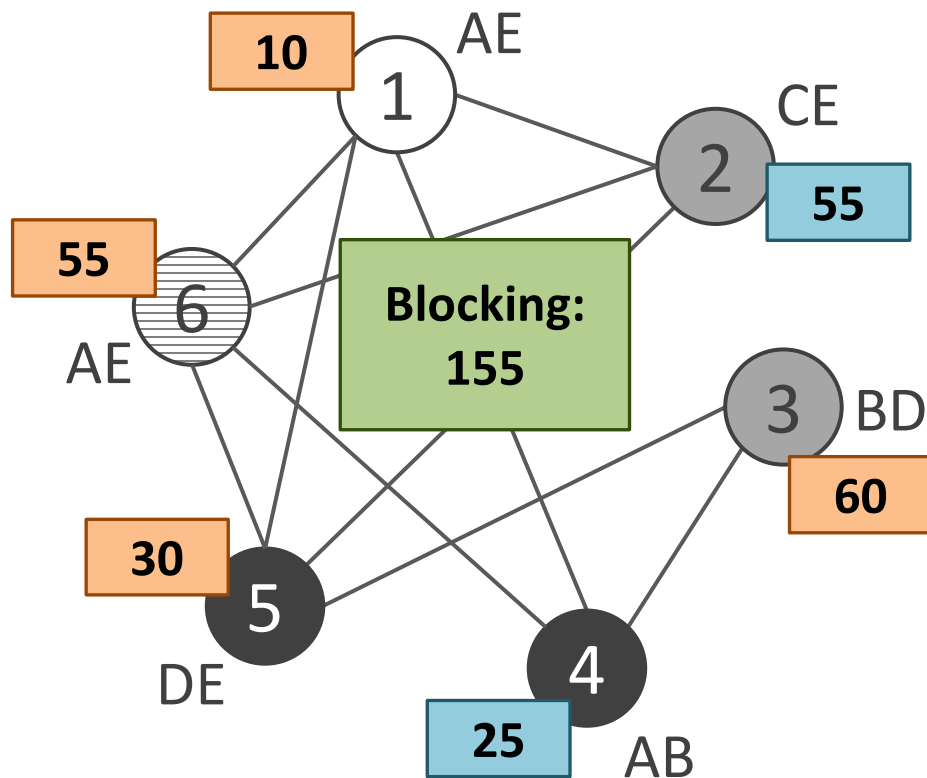
# Hierarchical Organization

Allow one vertex to represent multiple requests



# Hierarchical Organization

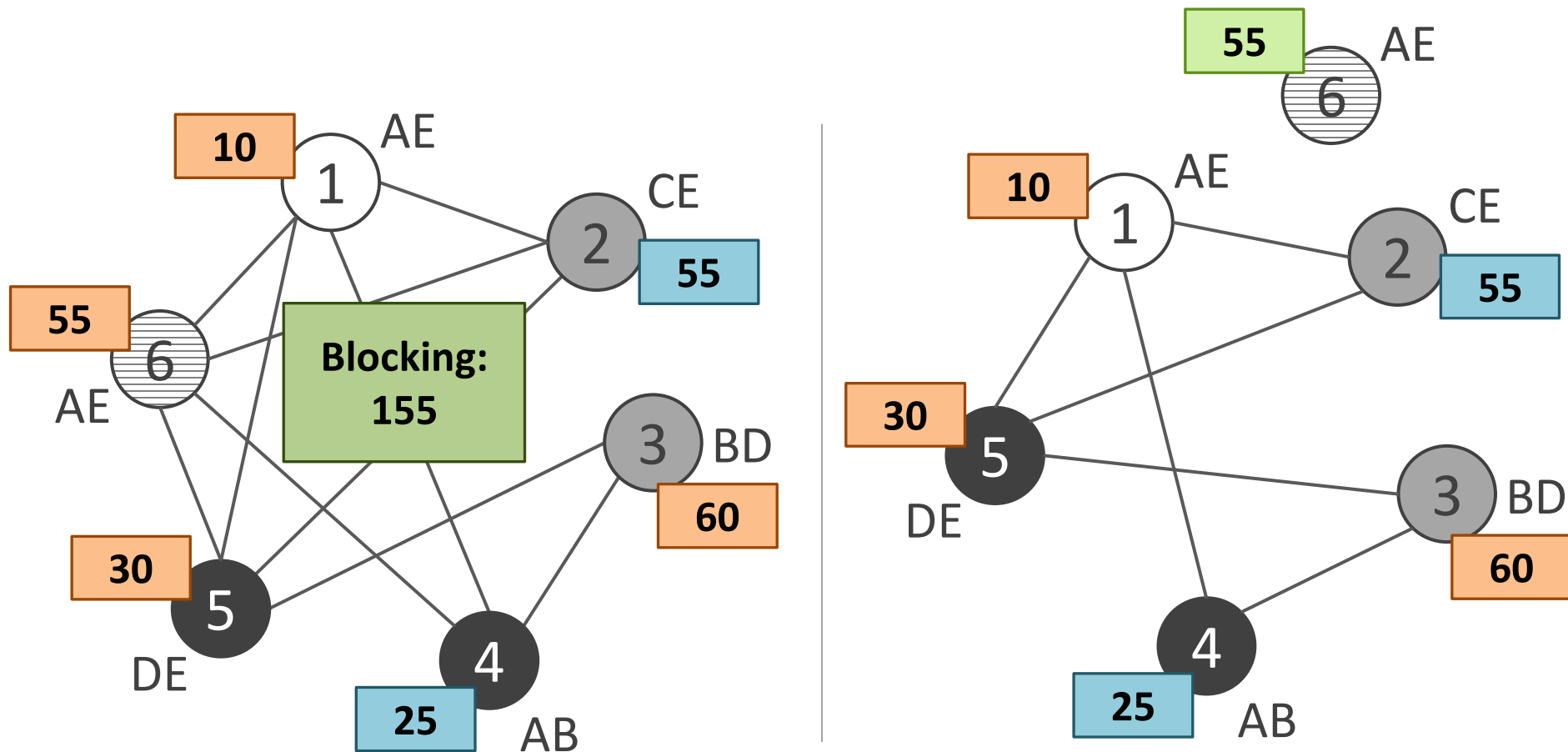
Allow one vertex to represent multiple requests





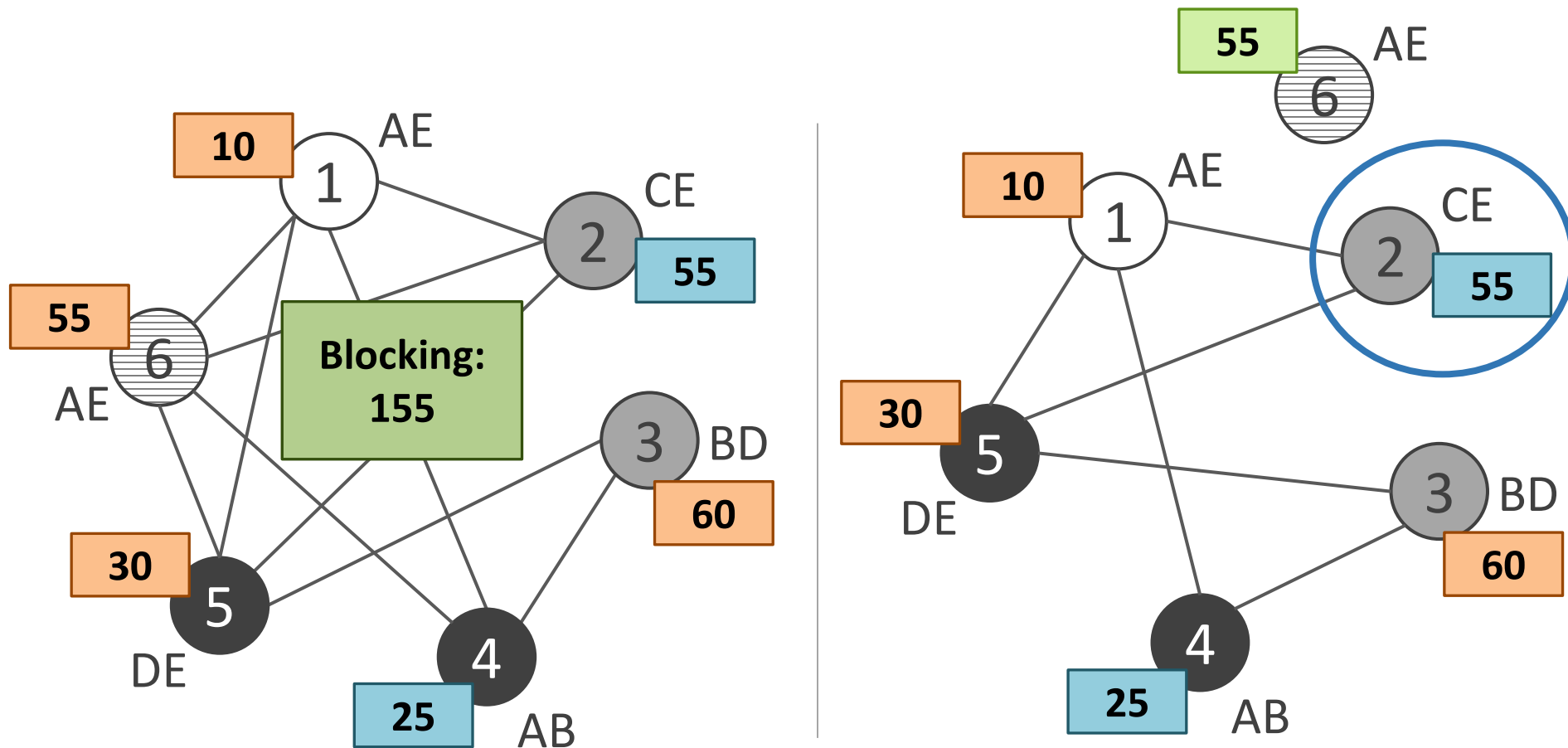
# Hierarchical Organization

Allow one vertex to represent multiple requests



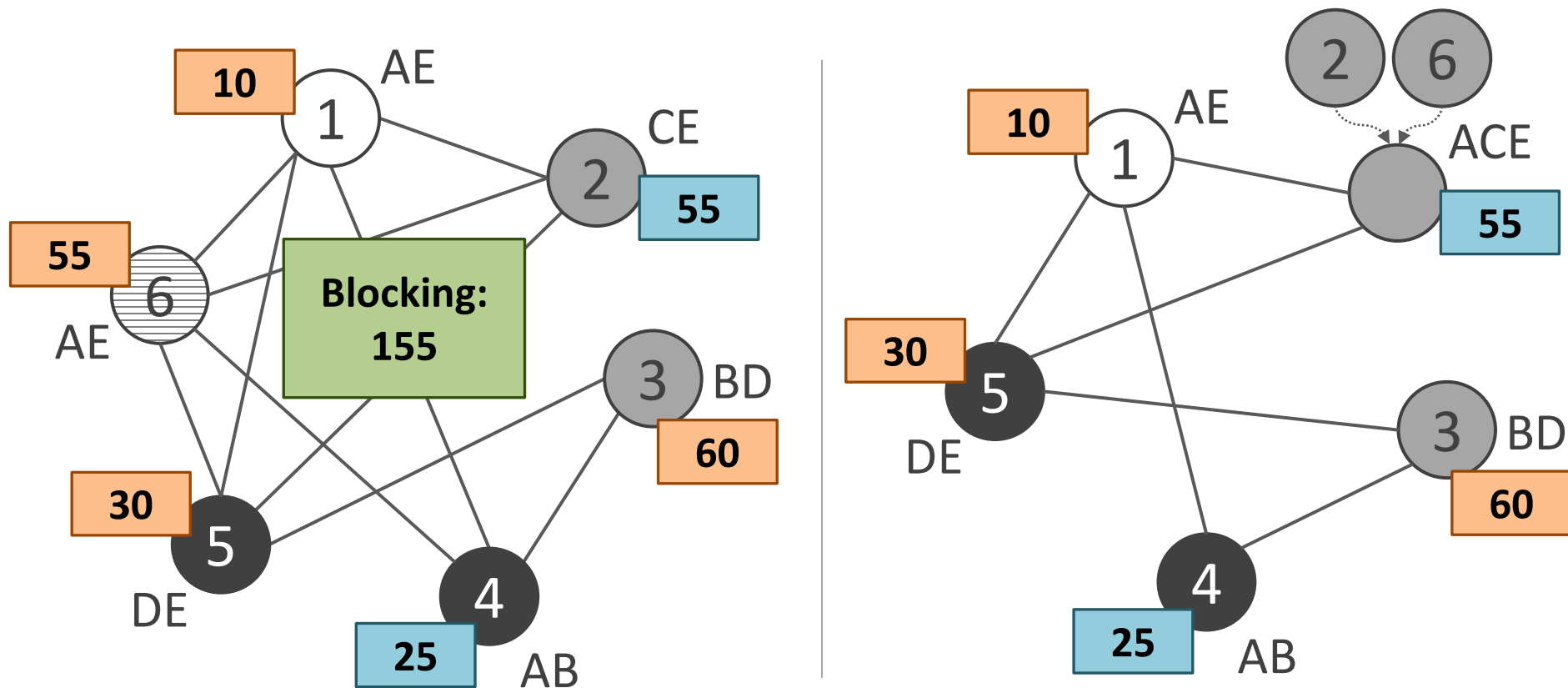
# Hierarchical Organization

Allow one vertex to represent multiple requests



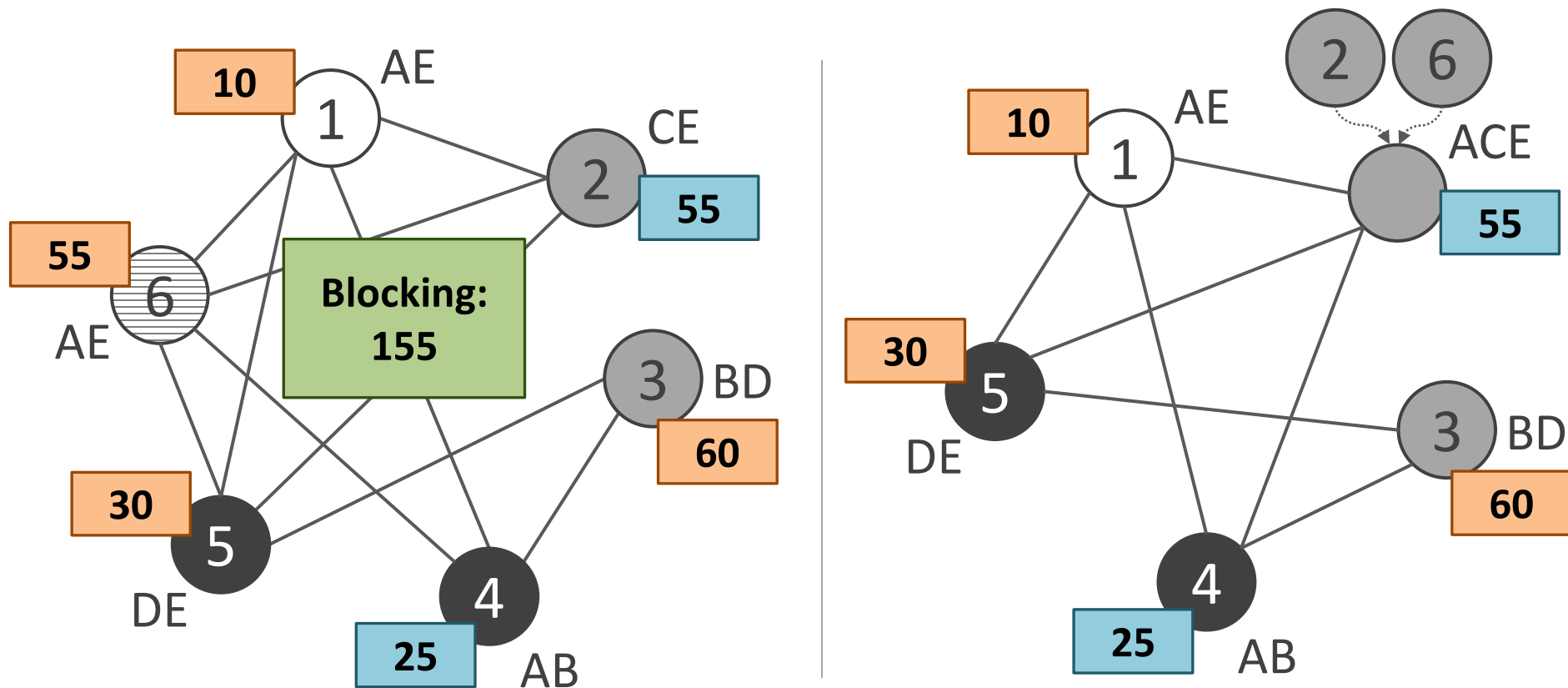
# Hierarchical Organization

Allow one vertex to represent multiple requests



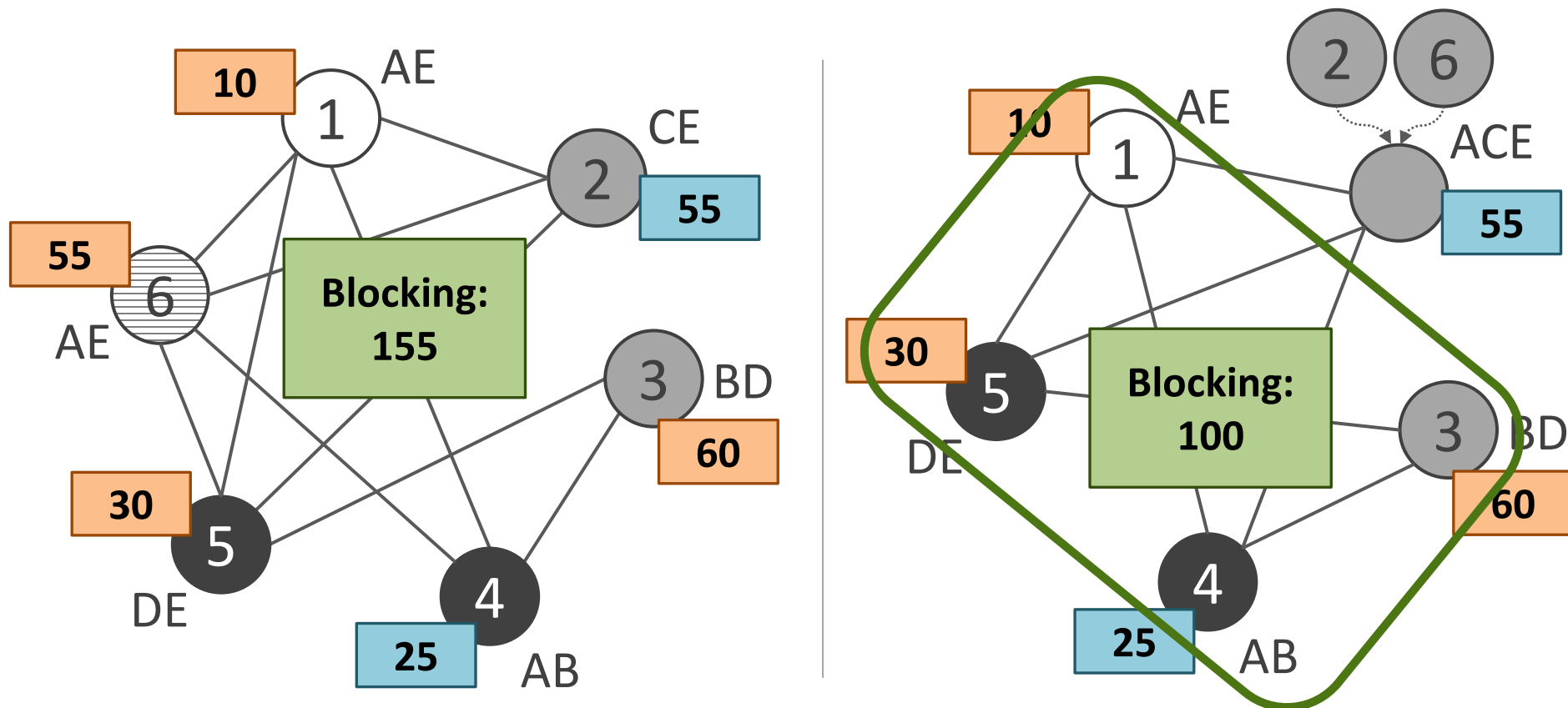
# Hierarchical Organization

Allow one vertex to represent multiple requests



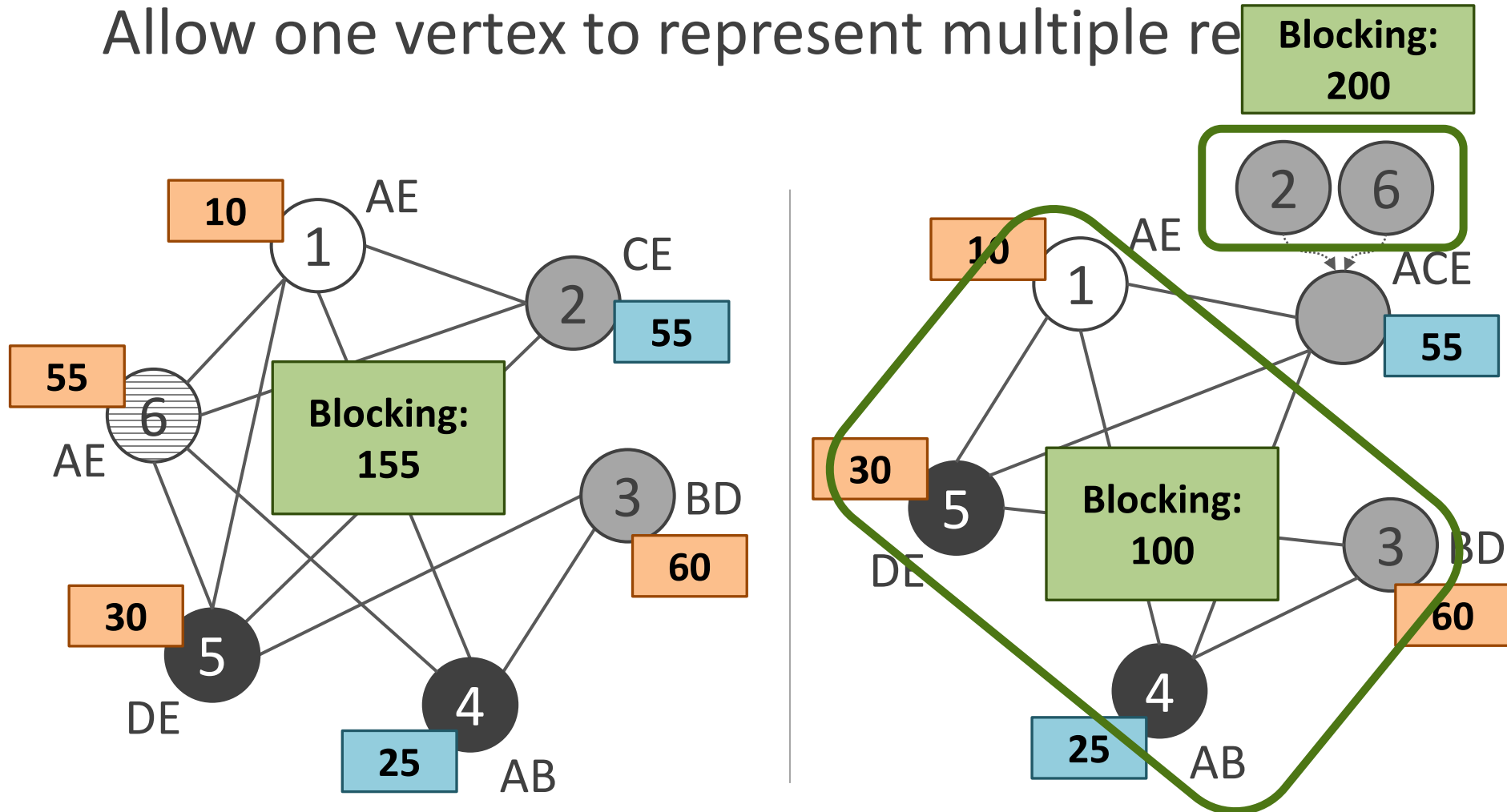
# Hierarchical Organization

Allow one vertex to represent multiple requests



# Hierarchical Organization

Allow one vertex to represent multiple re



# Evaluation: Group Formation

---

# Evaluation: Group Formation

---

- Transformed into Vertex Coloring Problem
  - NP-hard to find minimum coloring



# Evaluation: Group Formation

---

- Transformed into Vertex Coloring Problem
  - NP-hard to find minimum coloring
- Solved with an ILP
- Tested time to find solution to the ILP for different task utilizations

# Evaluation: Group Formation

---

- Transformed into Vertex Coloring Problem
  - NP-hard to find minimum coloring
- Solved with an ILP
- Tested time to find solution to the ILP for different task utilizations

**Task utilization:** Heavy: [0.5, 0.9], Medium: [0.1, 0.4], Medium-Light: [0.01, 0.1]

**Avg. number of vertices:** 23.4, 64.7, 291.6

# Evaluation: Group Formation

---

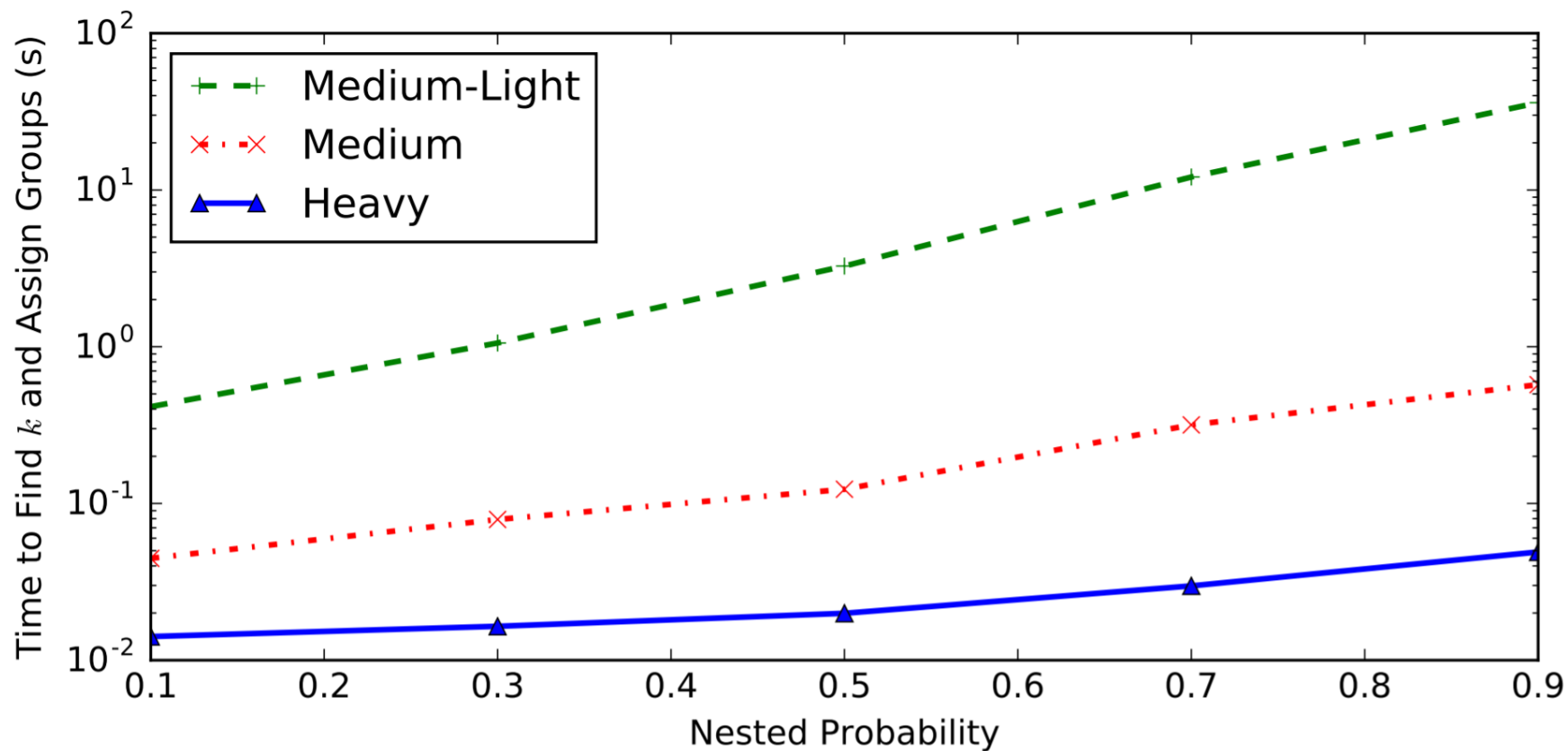
**Task utilization:** Heavy: [0.5, 0.9], Medium: [0.1, 0.4], Medium-Light: [0.01, 0.1]

**Avg. number of vertices:** 23.4, 64.7, 291.6

# Evaluation: Group Formation

**Task utilization:** Heavy: [0.5, 0.9], Medium: [0.1, 0.4], Medium-Light: [0.01, 0.1]

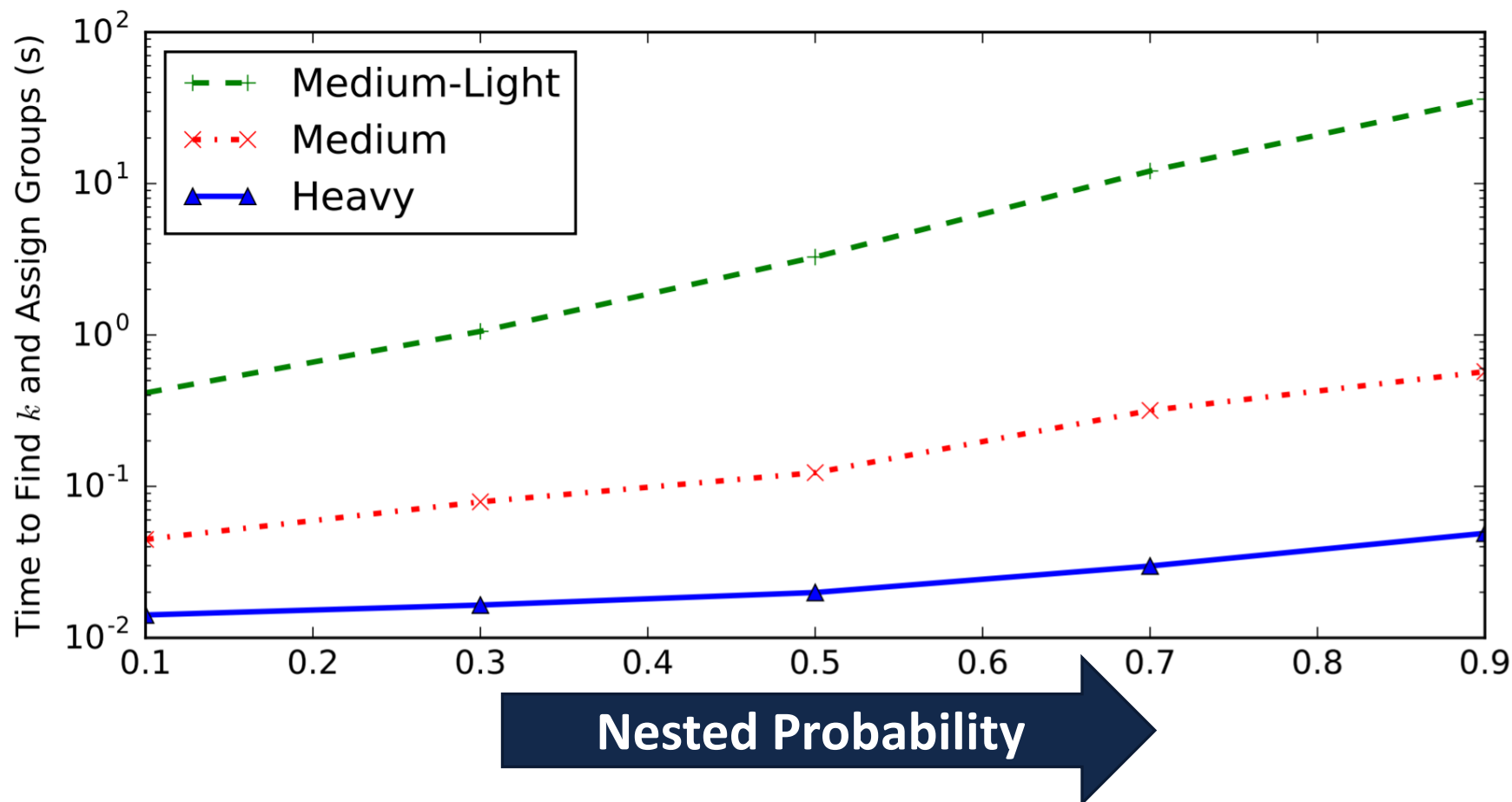
**Avg. number of vertices:** 23.4, 64.7, 291.6



# Evaluation: Group Formation

**Task utilization:** Heavy: [0.5, 0.9], Medium: [0.1, 0.4], Medium-Light: [0.01, 0.1]

**Avg. number of vertices:** 23.4, 64.7, 291.6

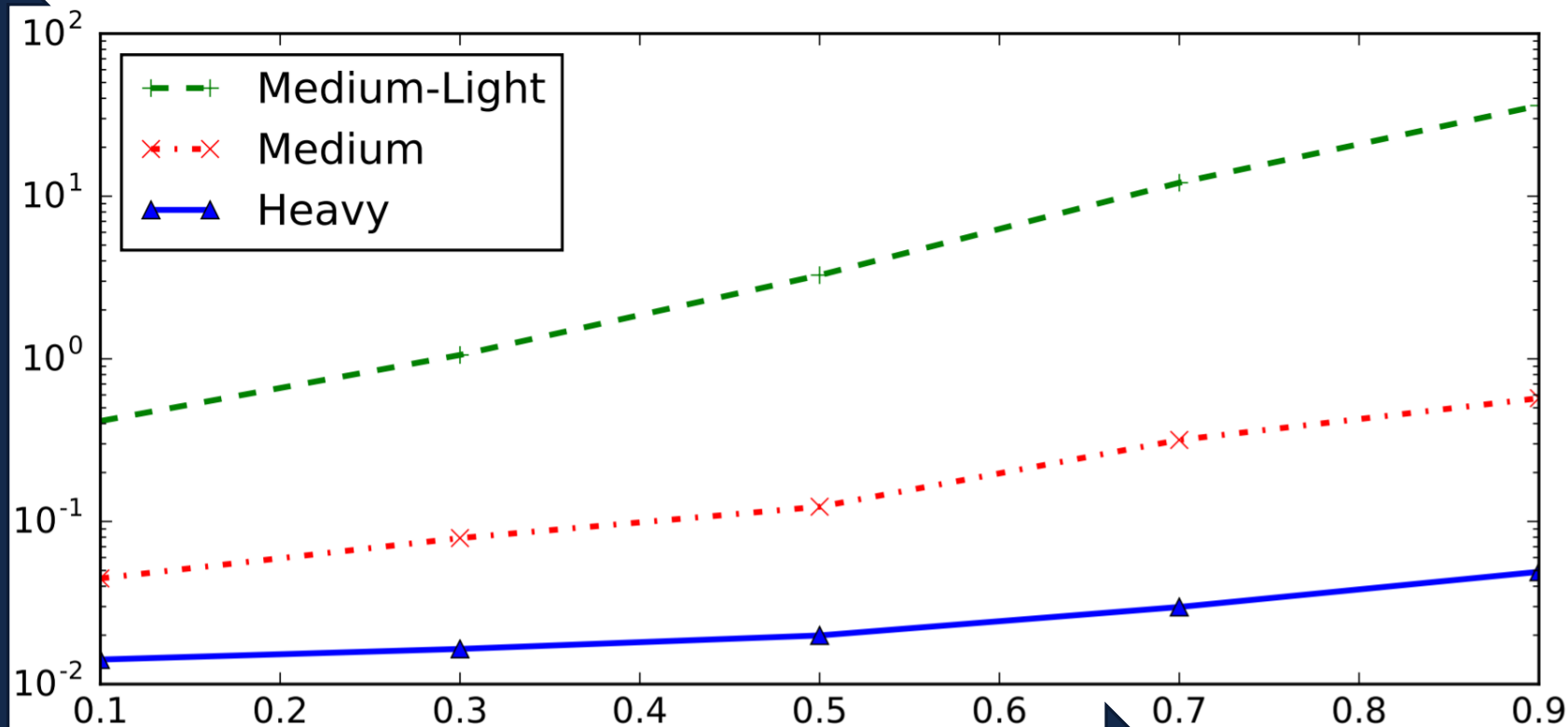


# Evaluation: Group Formation

**Task utilization:** Heavy: [0.5, 0.9], Medium: [0.1, 0.4], Medium-Light: [0.01, 0.1]

**Avg. number of vertices:** 23.4, 64.7, 291.6

Time to Find  $k$  and Assign Groups (s)



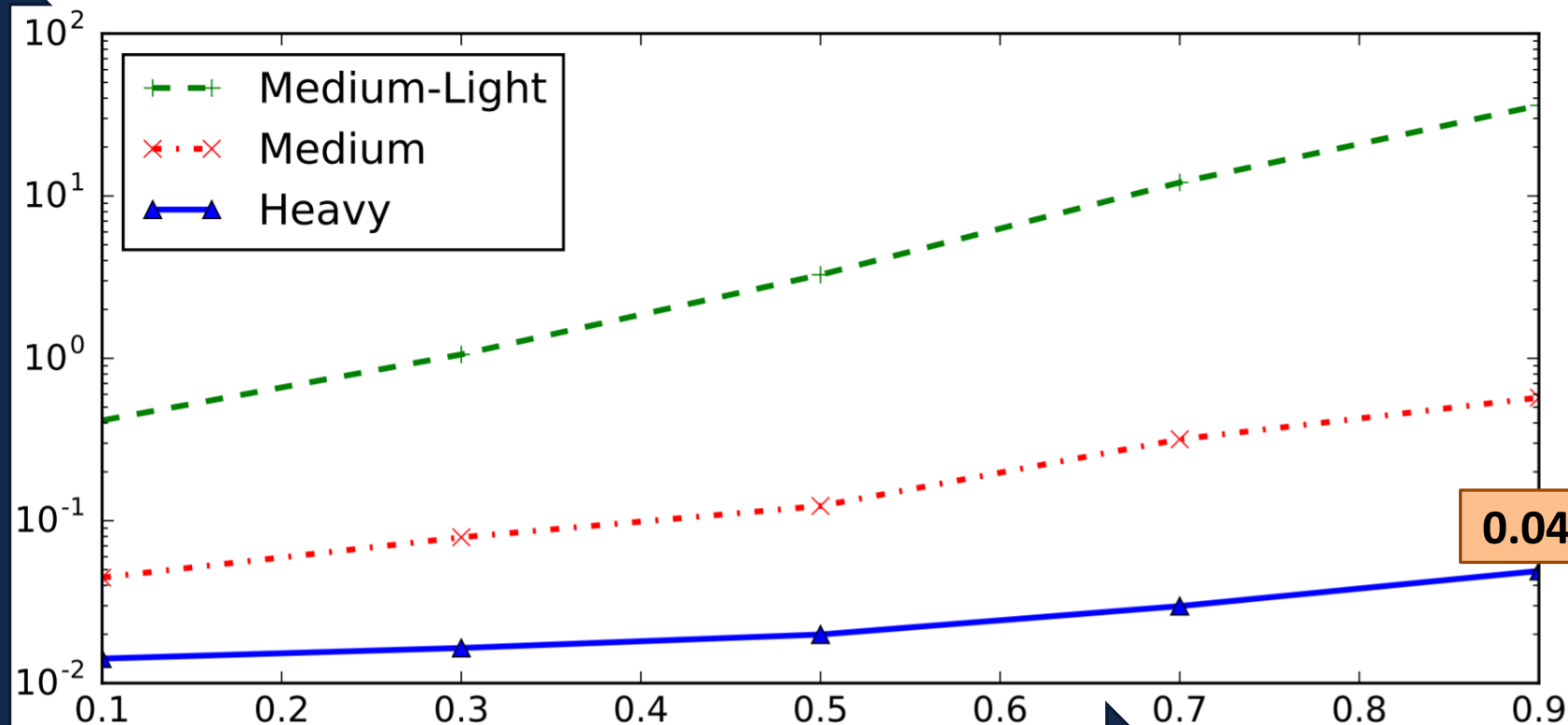
Nested Probability

# Evaluation: Group Formation

**Task utilization:** Heavy: [0.5, 0.9], Medium: [0.1, 0.4], Medium-Light: [0.01, 0.1]

**Avg. number of vertices:** 23.4, 64.7, 291.6

Time to Find  $k$  and Assign Groups (s)



Nested Probability

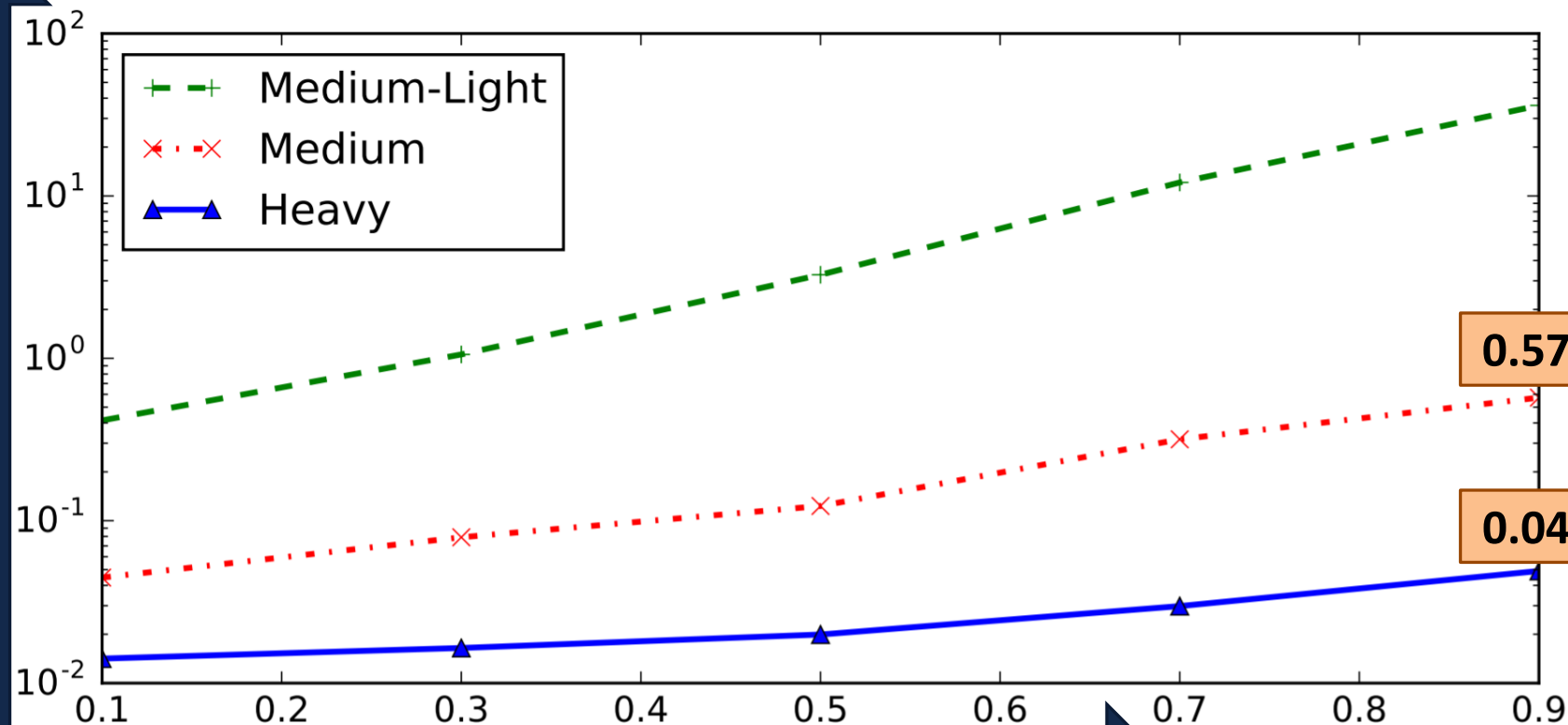
0.049

# Evaluation: Group Formation

**Task utilization:** Heavy: [0.5, 0.9], Medium: [0.1, 0.4], Medium-Light: [0.01, 0.1]

**Avg. number of vertices:** 23.4, 64.7, 291.6

Time to Find  $k$  and Assign Groups (s)



Nested Probability

0.573

0.049

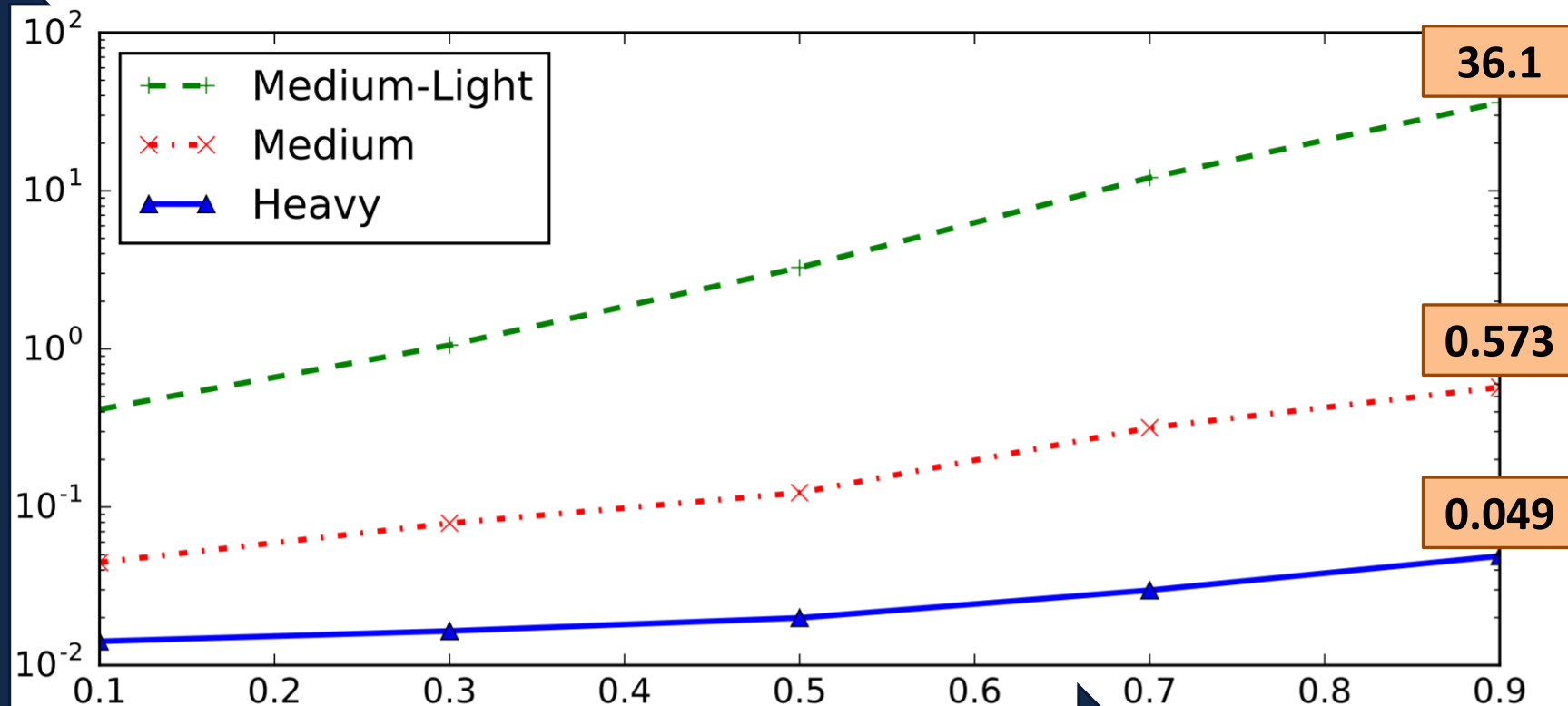


# Evaluation: Group Formation

**Task utilization:** Heavy: [0.5, 0.9], Medium: [0.1, 0.4], Medium-Light: [0.01, 0.1]

**Avg. number of vertices:** 23.4, 64.7, 291.6

Time to Find  $k$  and Assign Groups (s)



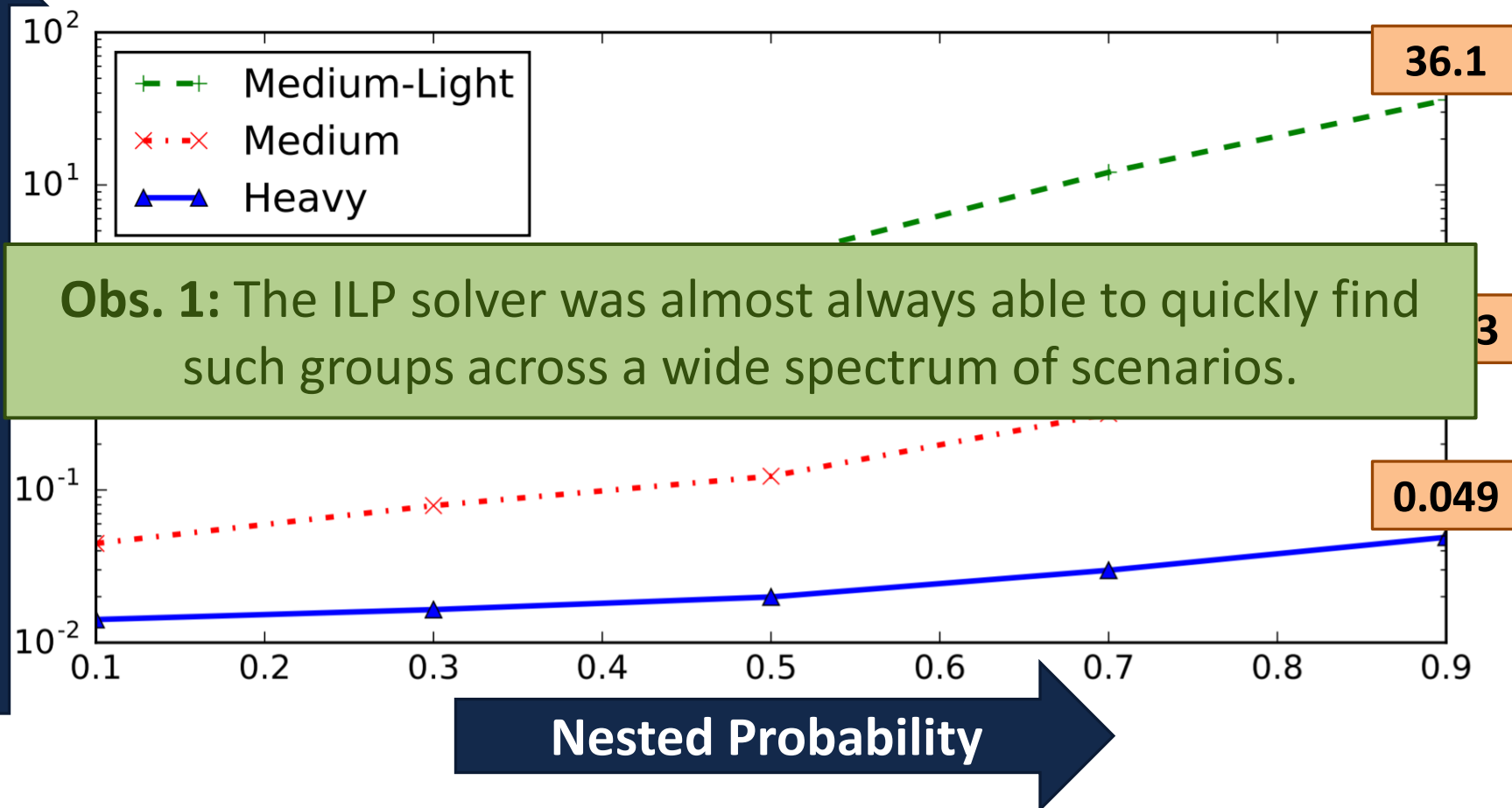
Nested Probability

# Evaluation: Group Formation

**Task utilization:** Heavy: [0.5, 0.9], Medium: [0.1, 0.4], Medium-Light: [0.01, 0.1]

**Avg. number of vertices:** 23.4, 64.7, 291.6

Time to Find  $k$  and Assign Groups (s)



Nested Probability

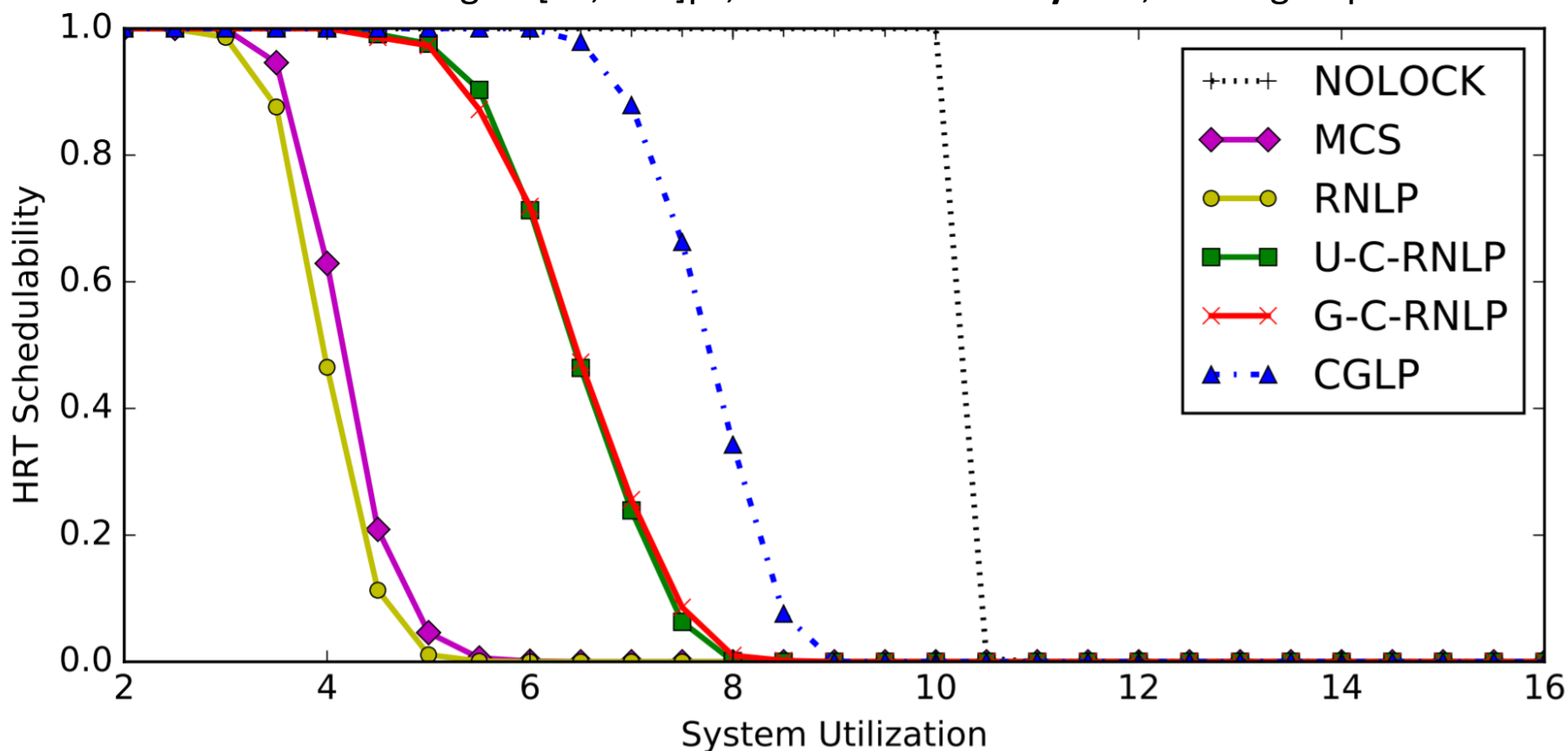
# Evaluation: Schedulability

---

- On basis of schedulability, using SchedCAT
- Incorporated blocking and overhead
- Generated 1,000 task sets per evaluation point
- 384 scenarios of varying:
  - Task Utilization
  - Period
  - Percentage of tasks issuing requests
  - Critical-section lengths
  - Nested probability
  - Nesting depth

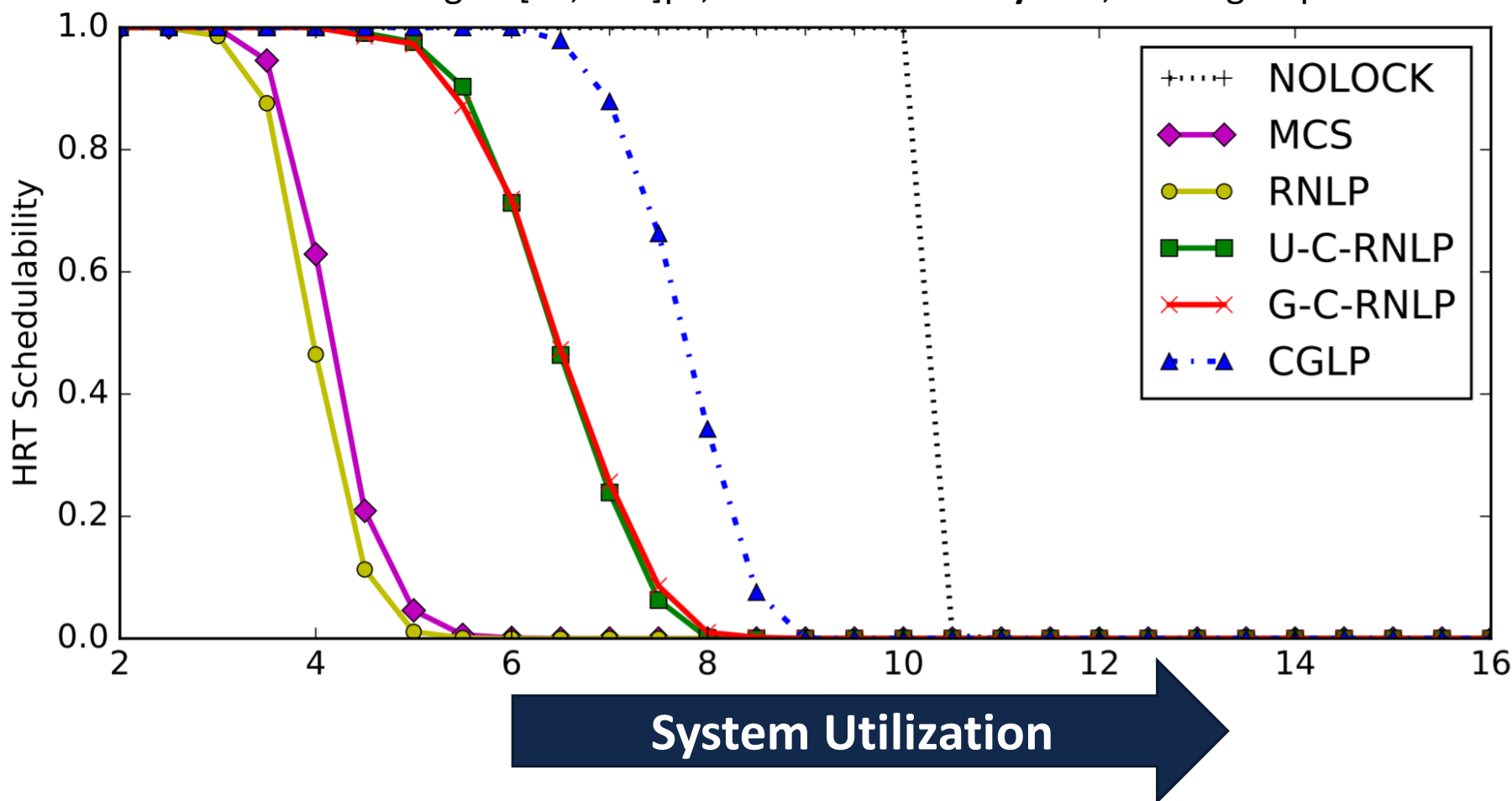
# Evaluation: Schedulability

Task utilization: [0.1, 0.4], Period: [3, 33]ms, **Percentage Issuing Requests: 100%**,  
 Critical-section Length: [15, 100] $\mu$ s, **Nested Probability: 0.5**, Nesting Depth: 4



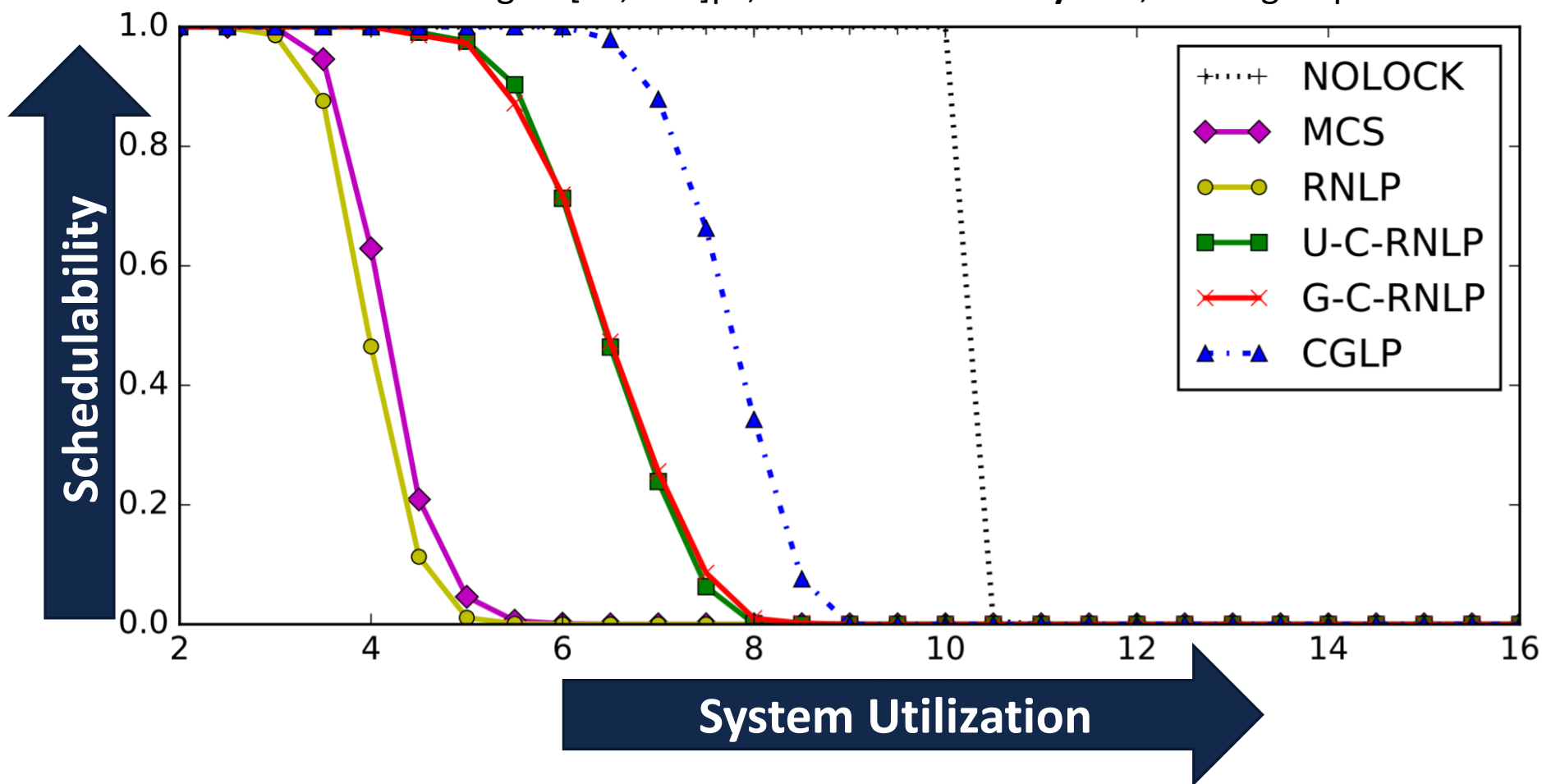
# Evaluation: Schedulability

Task utilization: [0.1, 0.4], Period: [3, 33]ms, **Percentage Issuing Requests: 100%**,  
 Critical-section Length: [15, 100] $\mu$ s, **Nested Probability: 0.5**, Nesting Depth: 4



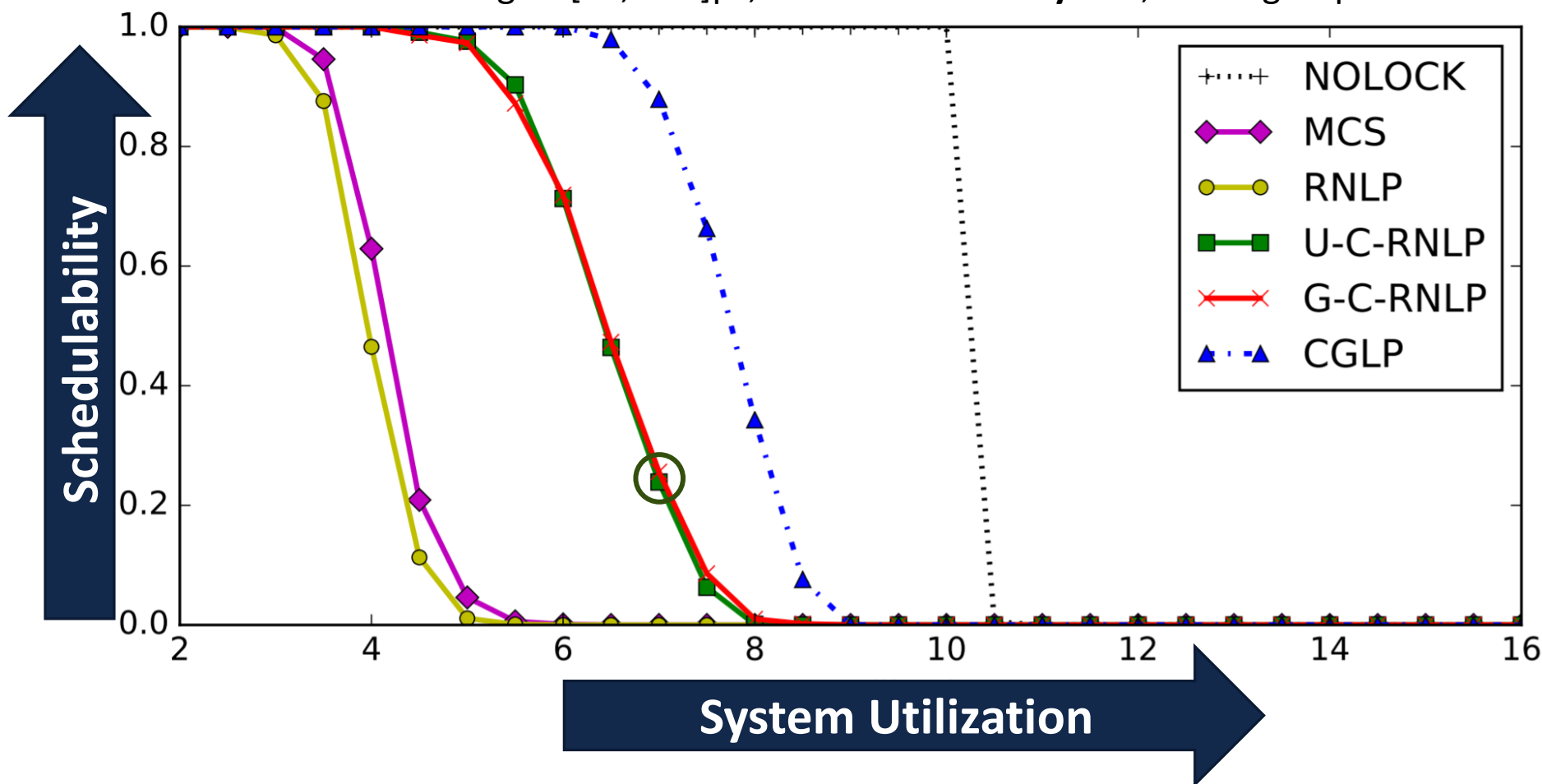
# Evaluation: Schedulability

Task utilization: [0.1, 0.4], Period: [3, 33]ms, **Percentage Issuing Requests: 100%**,  
 Critical-section Length: [15, 100] $\mu$ s, **Nested Probability: 0.5**, Nesting Depth: 4



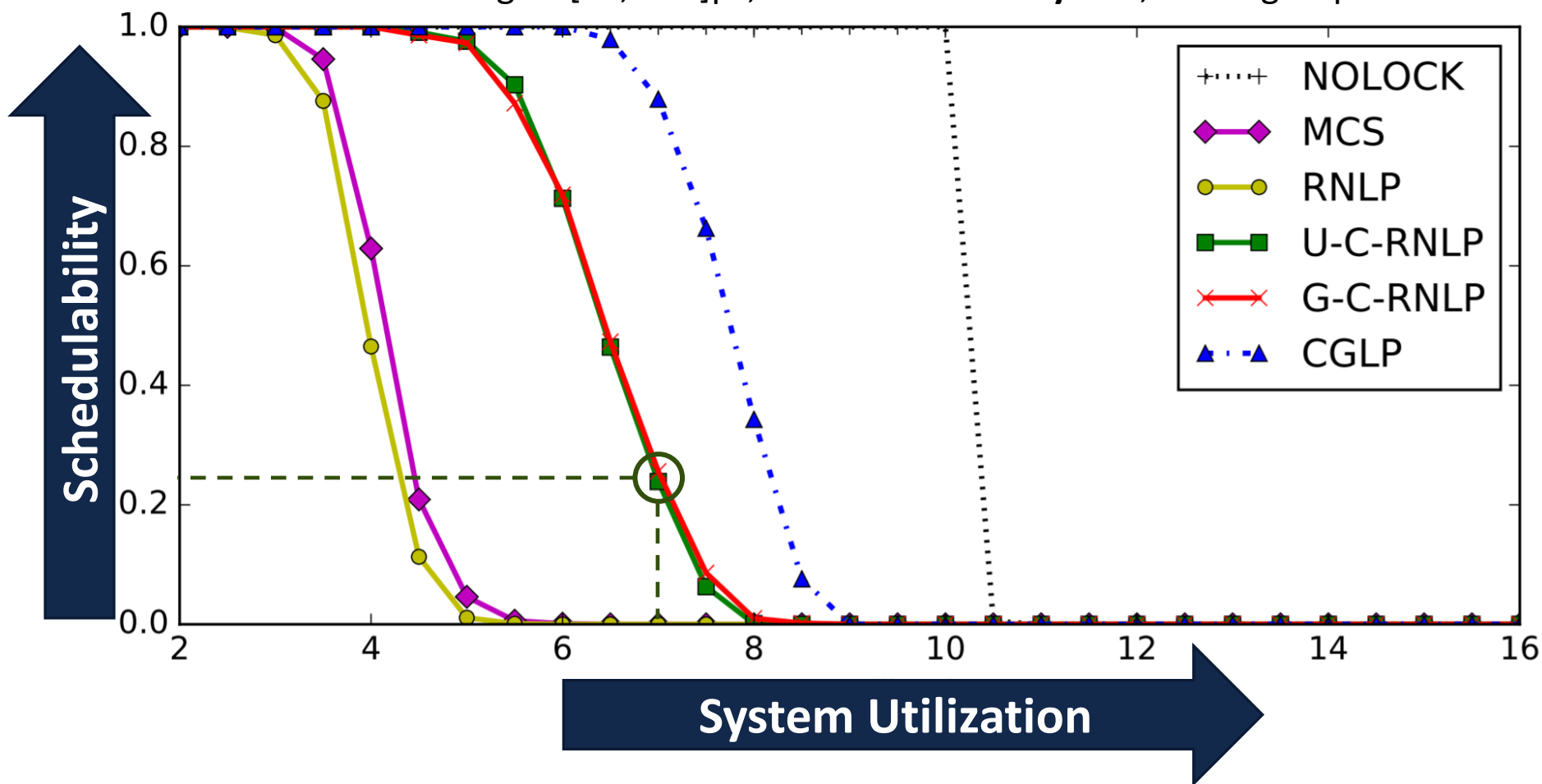
# Evaluation: Schedulability

Task utilization: [0.1, 0.4], Period: [3, 33]ms, **Percentage Issuing Requests: 100%**,  
 Critical-section Length: [15, 100] $\mu$ s, **Nested Probability: 0.5**, Nesting Depth: 4



# Evaluation: Schedulability

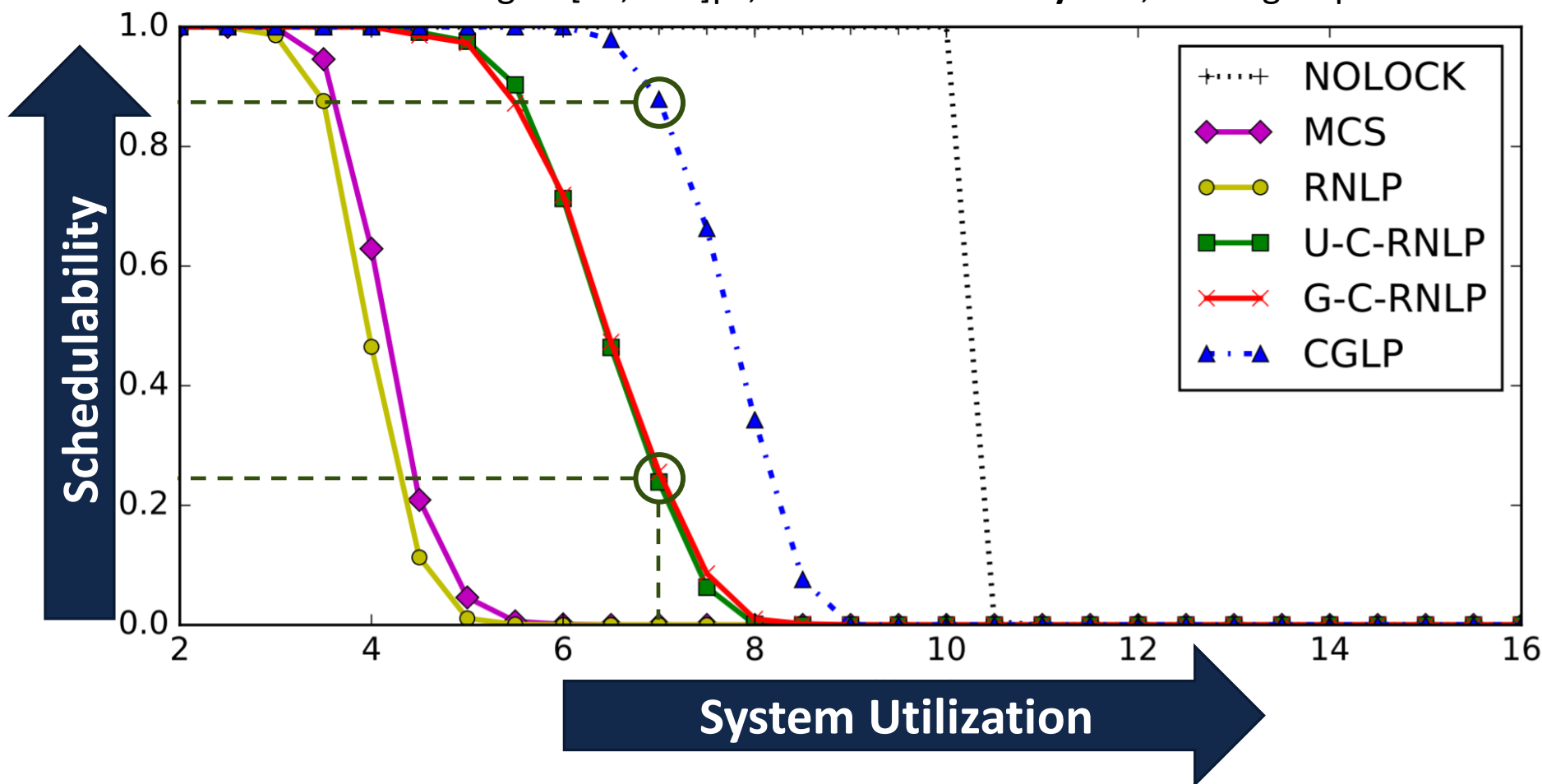
Task utilization: [0.1, 0.4], Period: [3, 33]ms, **Percentage Issuing Requests: 100%**,  
 Critical-section Length: [15, 100] $\mu$ s, **Nested Probability: 0.5**, Nesting Depth: 4





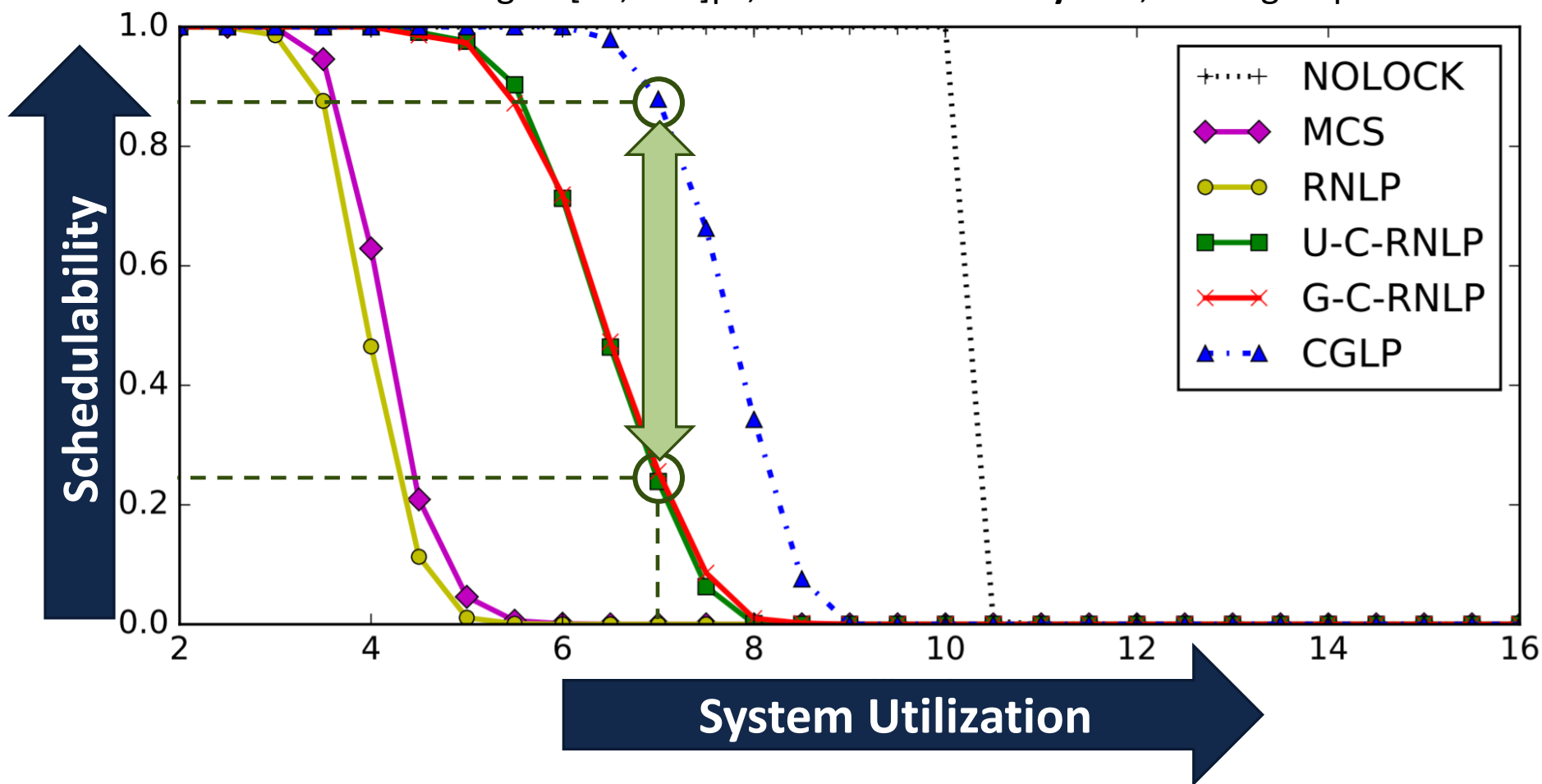
# Evaluation: Schedulability

Task utilization: [0.1, 0.4], Period: [3, 33]ms, **Percentage Issuing Requests: 100%**,  
 Critical-section Length: [15, 100] $\mu$ s, **Nested Probability: 0.5**, Nesting Depth: 4



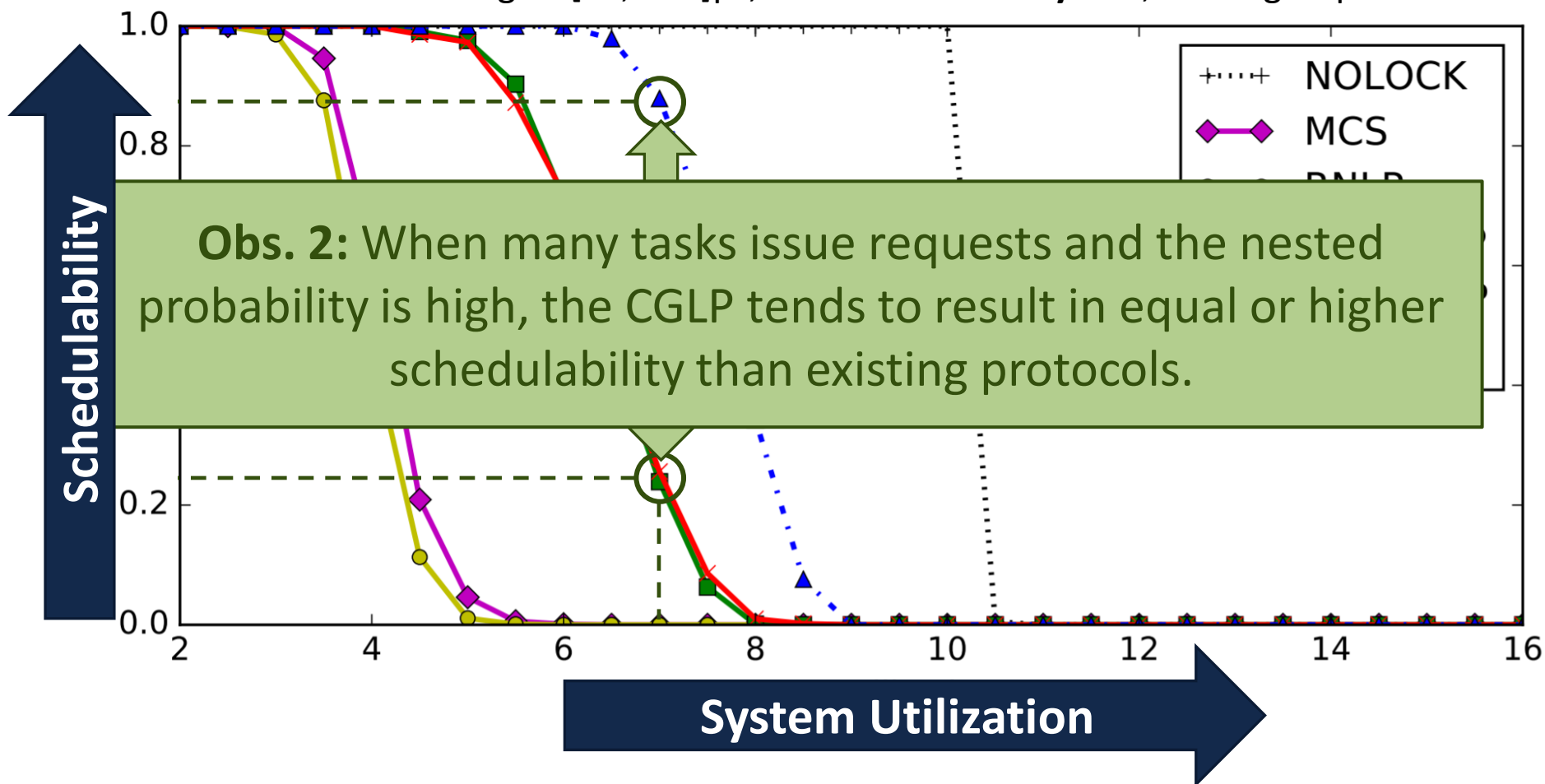
# Evaluation: Schedulability

Task utilization: [0.1, 0.4], Period: [3, 33]ms, **Percentage Issuing Requests: 100%**,  
 Critical-section Length: [15, 100] $\mu$ s, **Nested Probability: 0.5**, Nesting Depth: 4



# Evaluation: Schedulability

Task utilization: [0.1, 0.4], Period: [3, 33]ms, **Percentage Issuing Requests: 100%**,  
 Critical-section Length: [15, 100] $\mu$ s, **Nested Probability: 0.5**, Nesting Depth: 4



# Evaluation: Schedulability

---

**Obs. 3:** When the tasks are grouped by critical-section length, the schedulability of the CGLP increases.

# Evaluation: Schedulability

---

**Obs. 3:** When the tasks are grouped by critical-section length, the schedulability of the CGLP increases.

**Obs. 4:** Schedulability under the CGLP decreases as the number of groups increases.

# Future Work

---

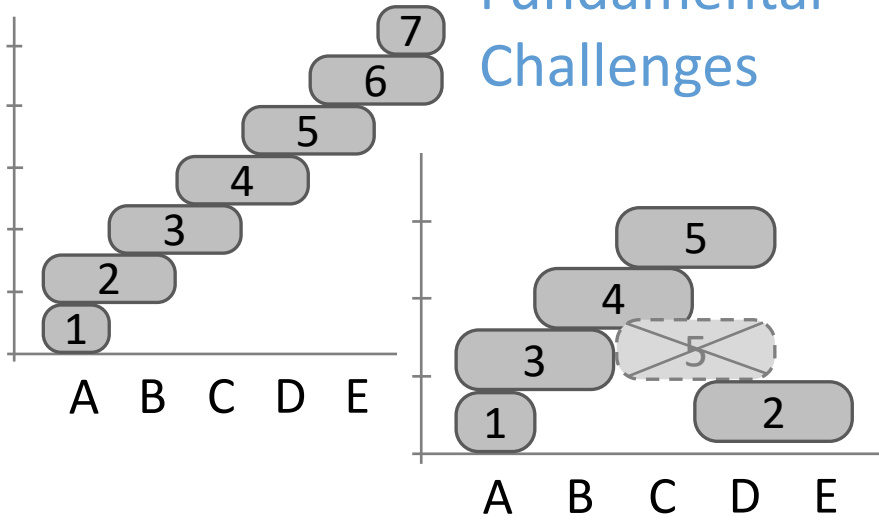
- Determine groups based on critical-section length
- Construct the optimal hierarchical structure
- Explore the benefits of handling mixed-type requests

---

# Concurrency Groups

---

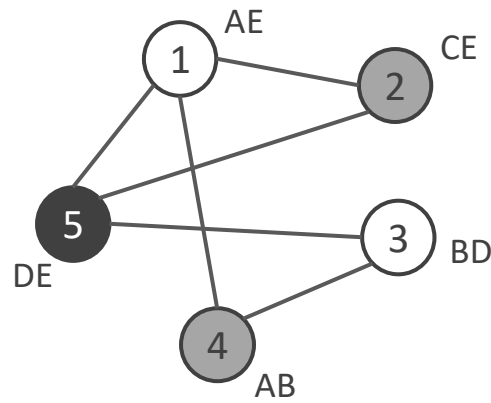
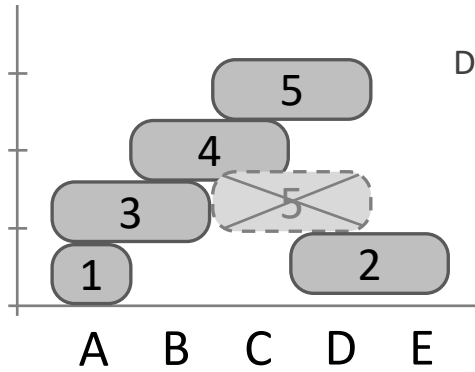
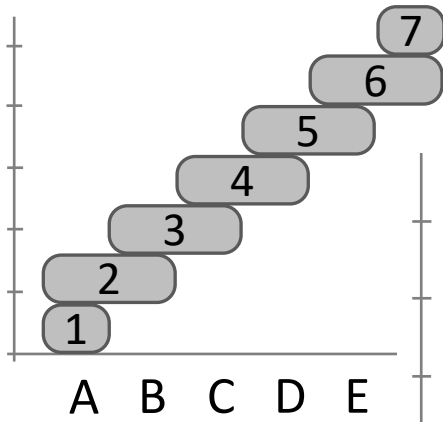
## Fundamental Challenges



# Concurrency Groups



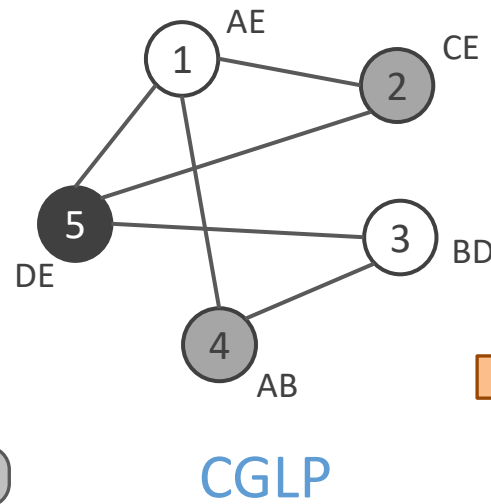
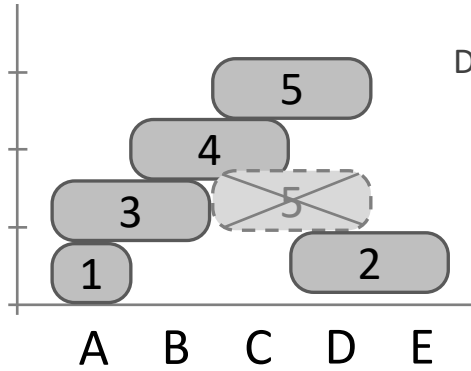
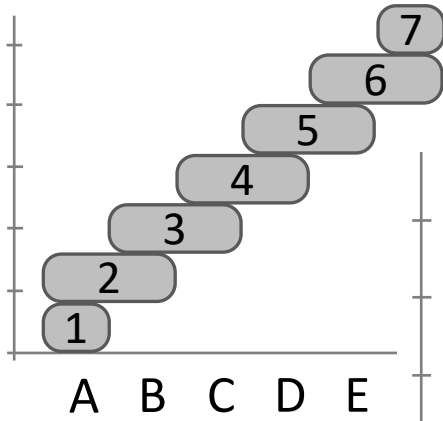
## Fundamental Challenges



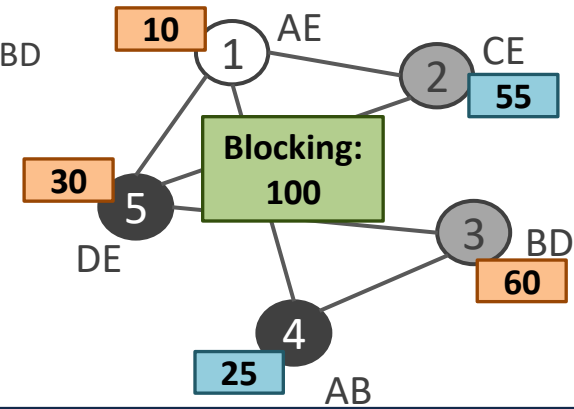
CGLP

# Concurrency Groups

## Fundamental Challenges

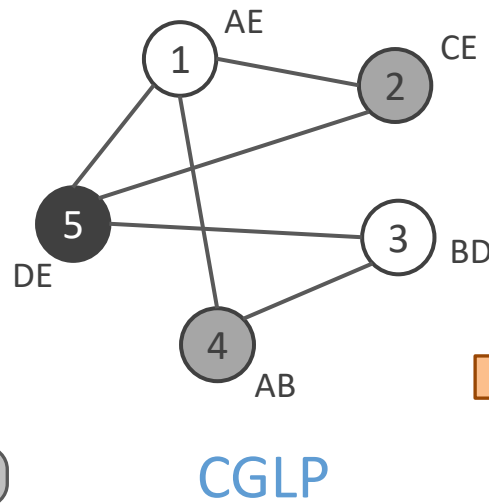
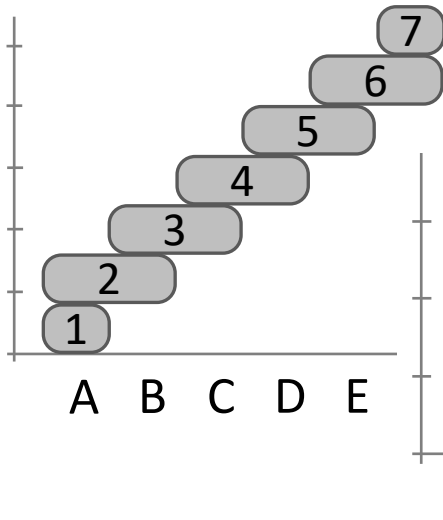


## Refinements

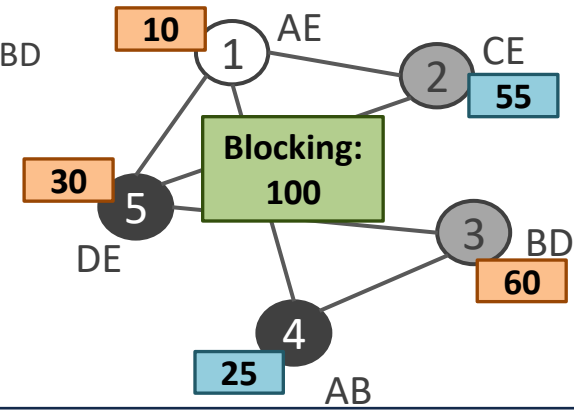


# Concurrency Groups

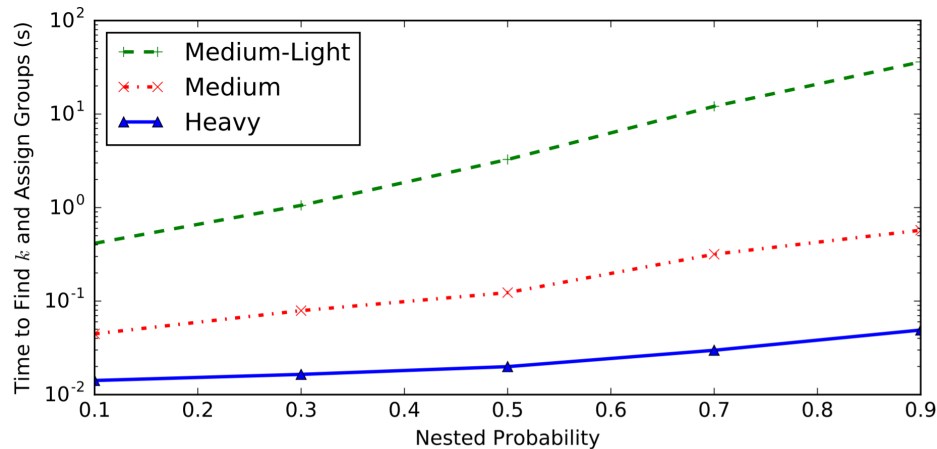
## Fundamental Challenges



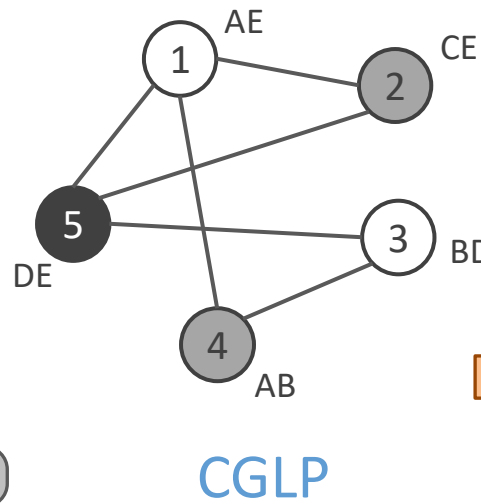
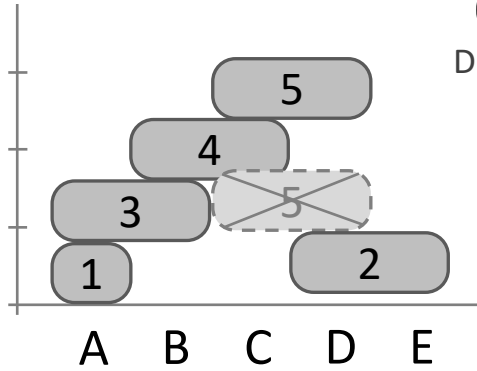
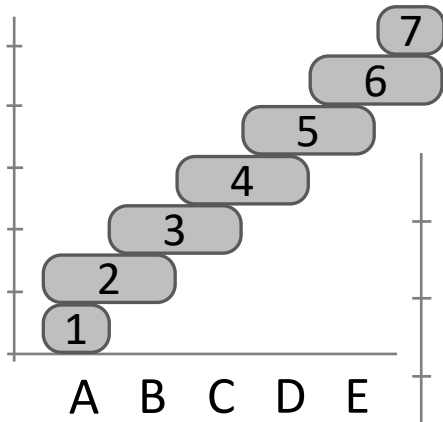
## Refinements



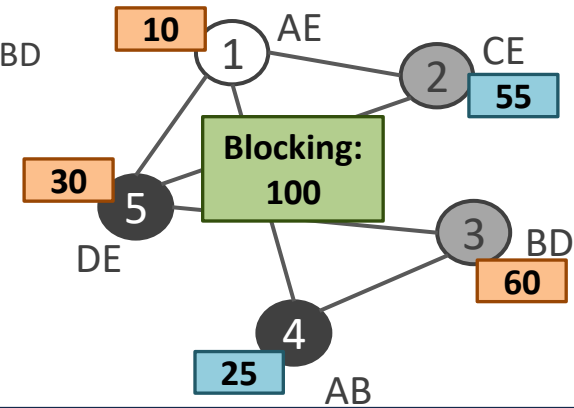
# Concurrency Groups



## Fundamental Challenges



## Refinements



# Concurrency Groups

