

# Rejection-Based Simulation of Stochastic Spreading Processes on Complex Networks

**Gerrit Großmann**  
Verena Wolf  
Saarland University



# Complex Networks

Networks are everywhere

- Friendship networks
- Online social networks
- Telecommunication networks
- Infrastructure networks
- Biological and Ecological Networks
- ...



# Motivation

Understand spreading phenomena of

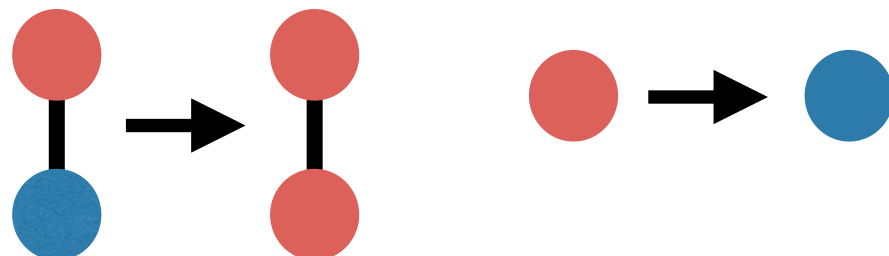
- Infectious diseases
- Computer viruses
- Rumours/opinions/emotions
- Blackouts
- ...

# Spreading Process

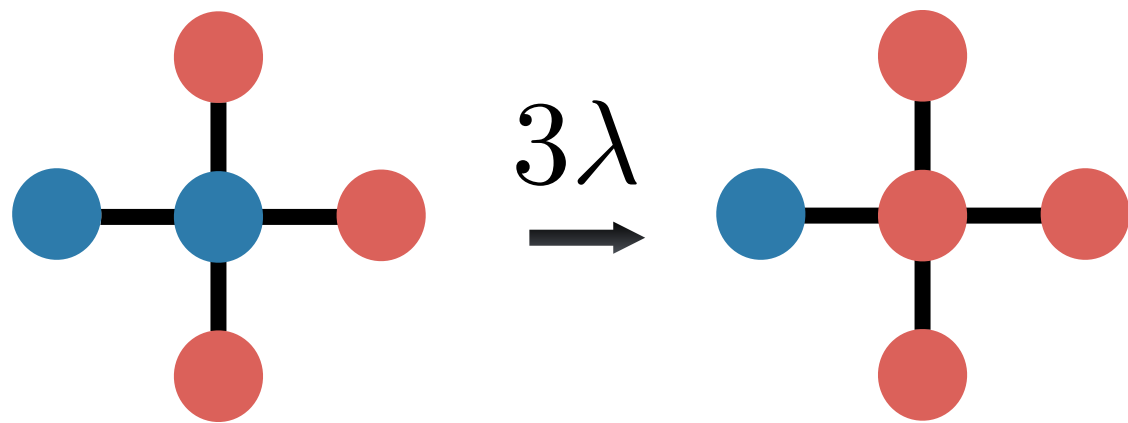
- Fixed graph topology
- Continuous time dynamics
- Nodes have local states
- Nodes' states change randomly w.r.t. rules

Classical example: SIS model

- 2 local states (**infected**, **susceptible**)
- 2 rules (infection, recovery)

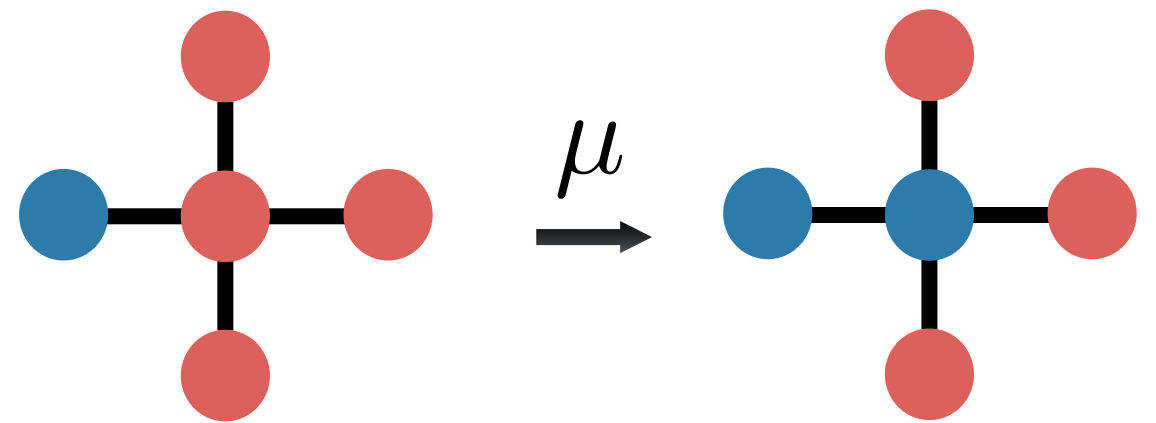


# SIS Model



Infection (edge-based)

(rate depends on neighborhood)

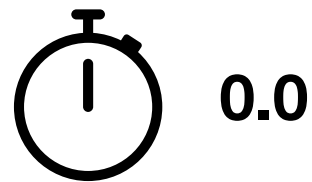
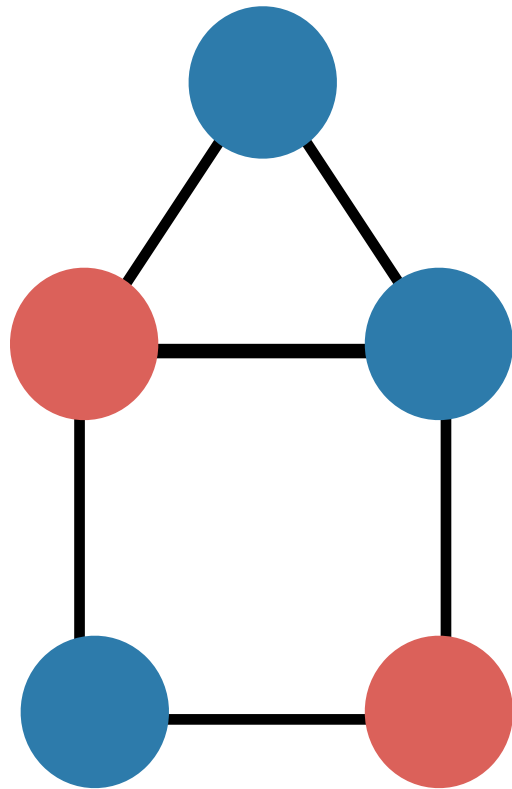


Recovery (node-based)

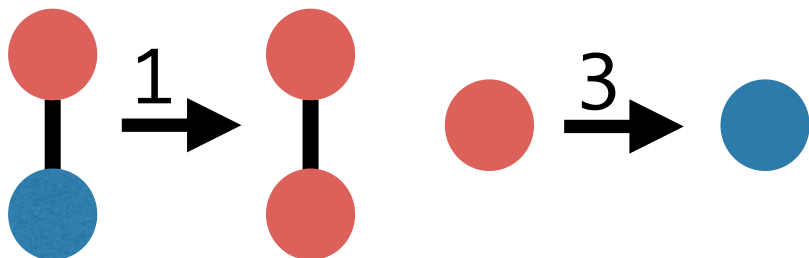
(rate does not depend on neighborhood)

- 2 local states (**infected**, **susceptible**)
- 2 rules (infection, recovery)

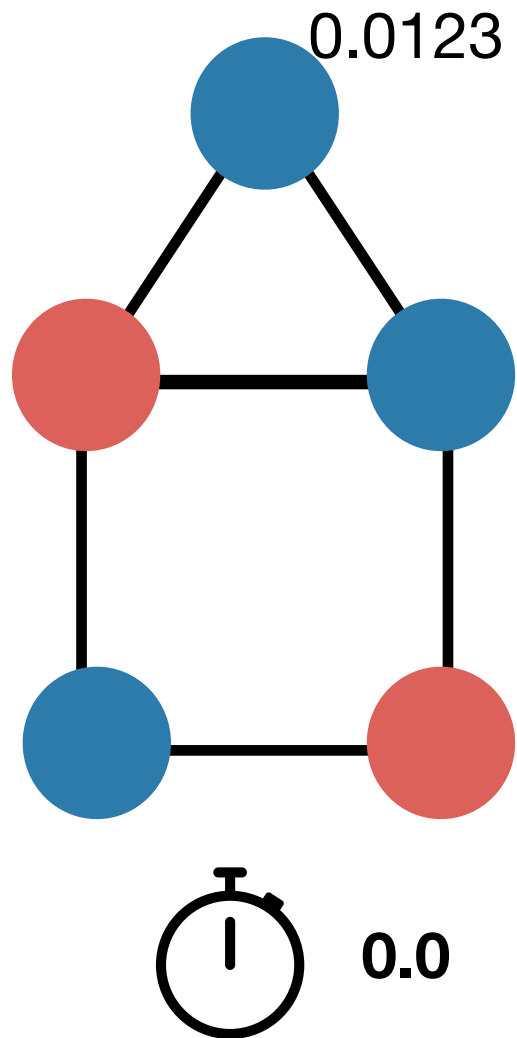
# SIS Naïve Simulation



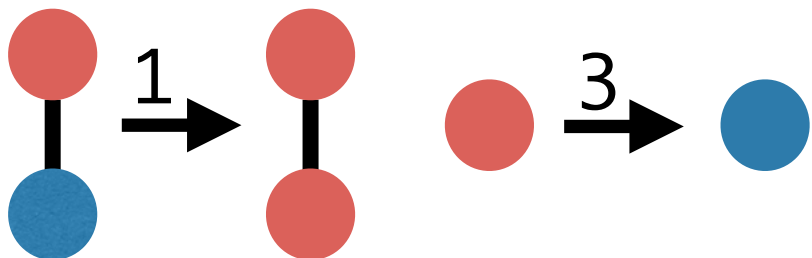
**Rules:**



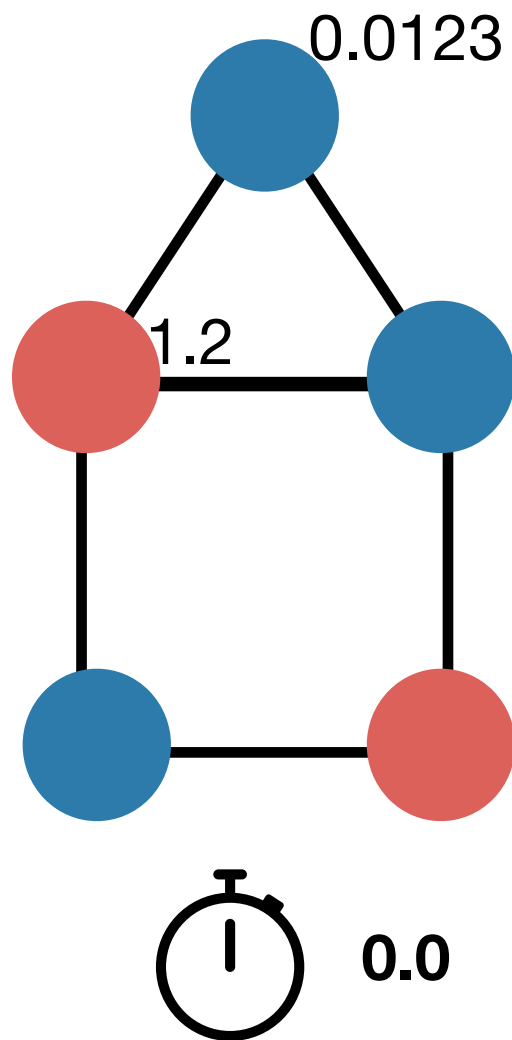
# SIS Naïve Simulation



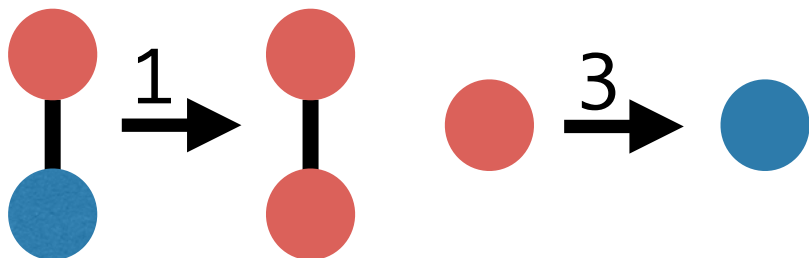
**Rules:**



# SIS Naïve Simulation

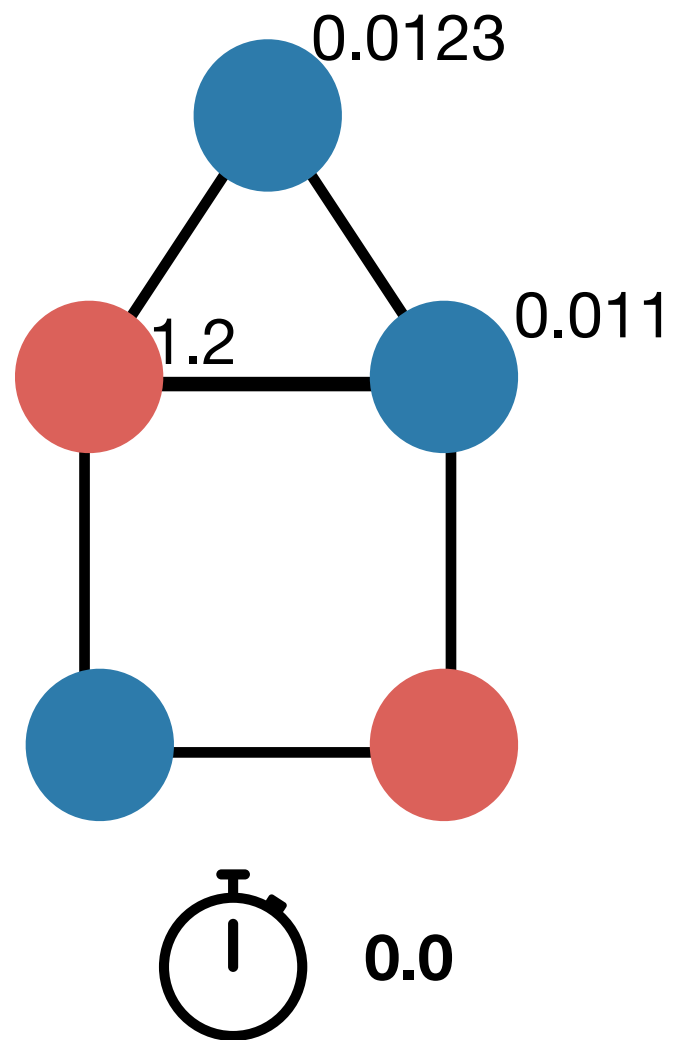


**Rules:**

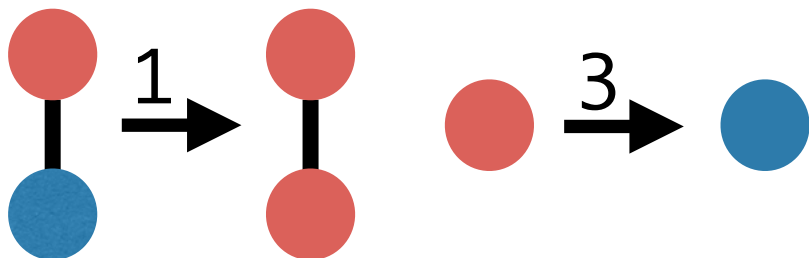




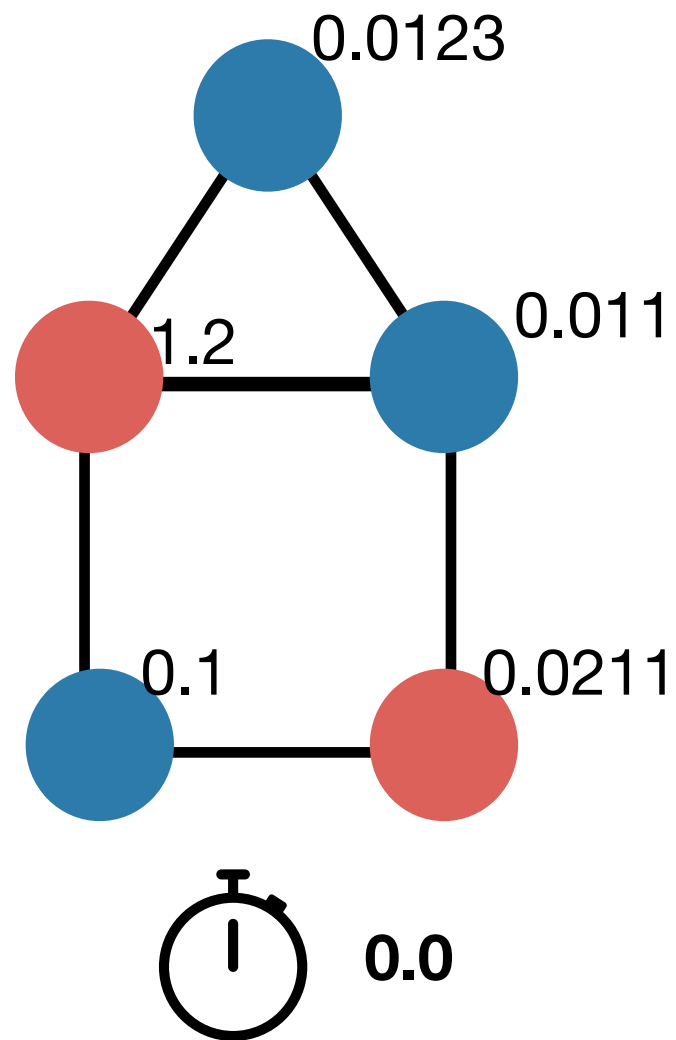
# SIS Naïve Simulation



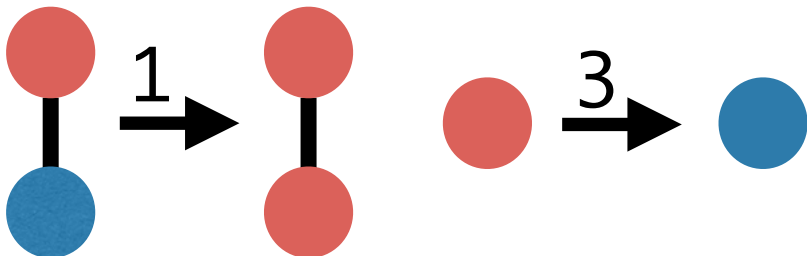
**Rules:**



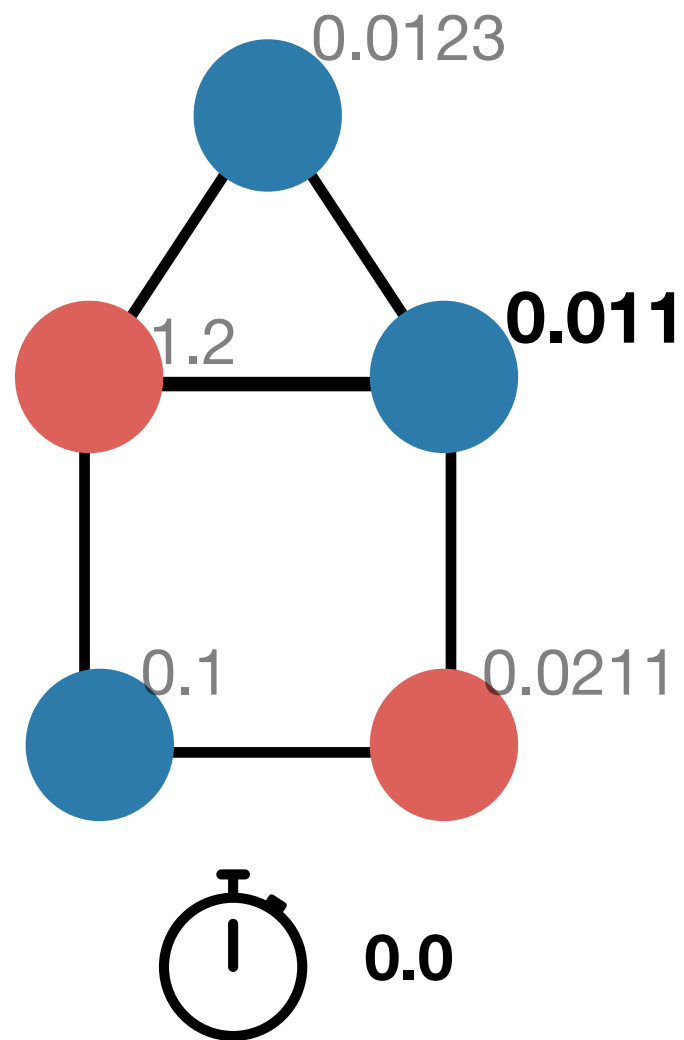
# SIS Naïve Simulation



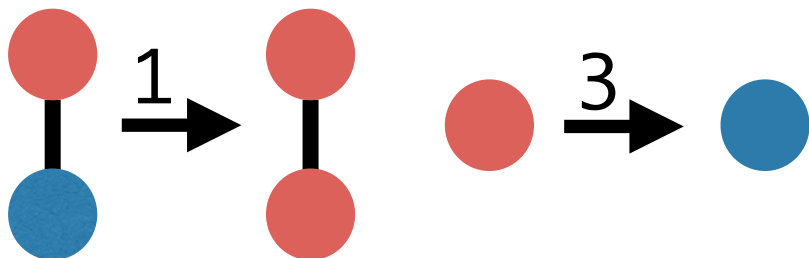
**Rules:**



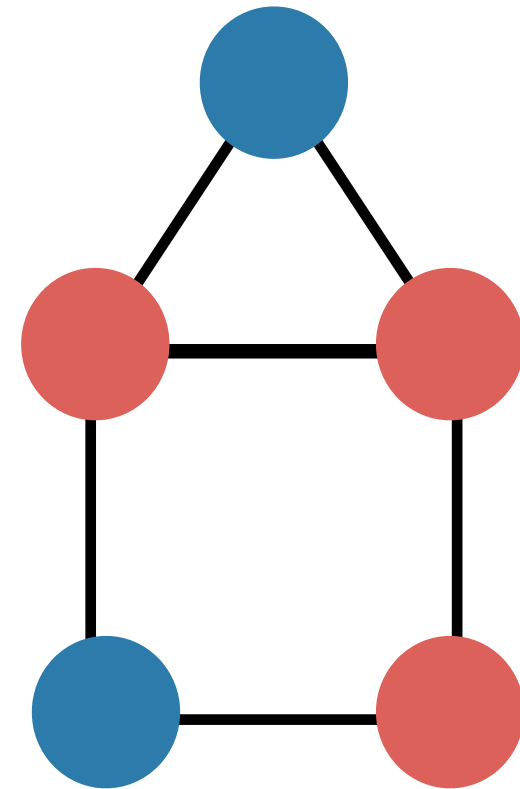
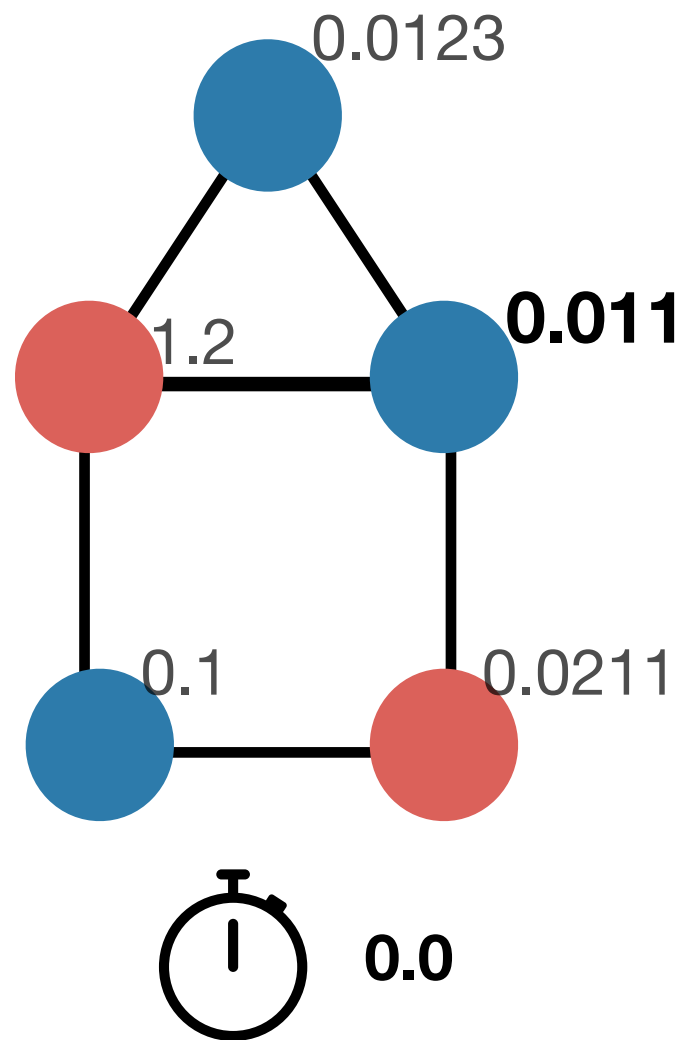
# SIS Naïve Simulation



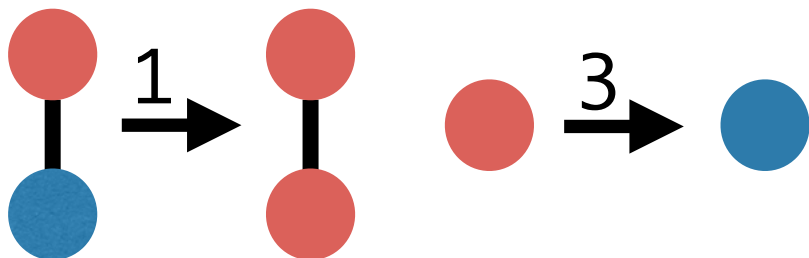
**Rules:**



# SIS Naïve Simulation

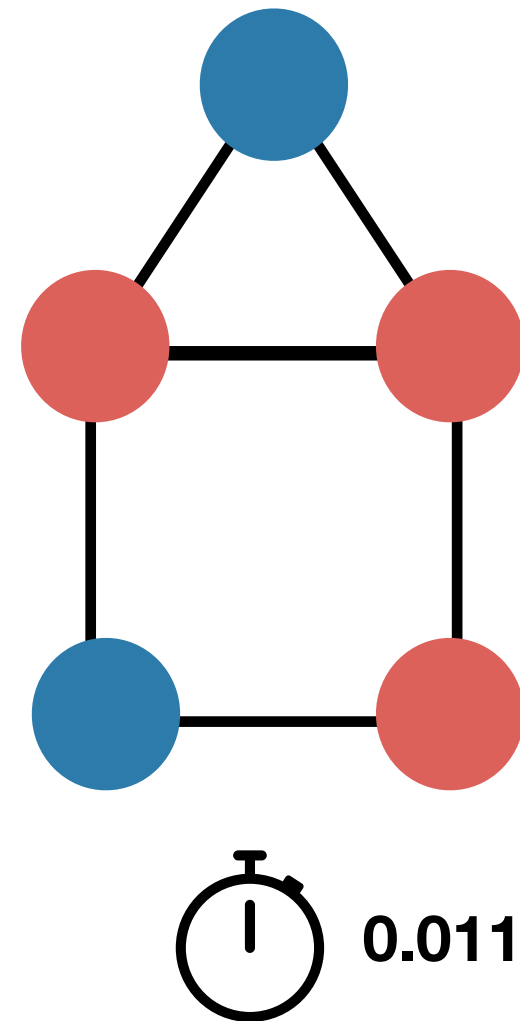
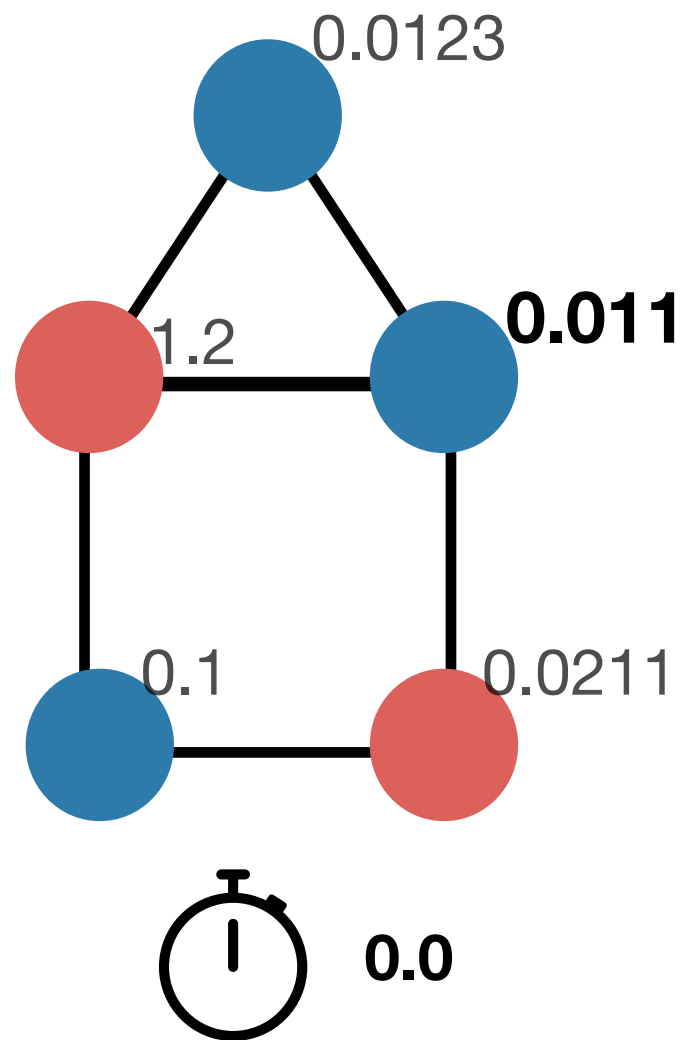


Rules:

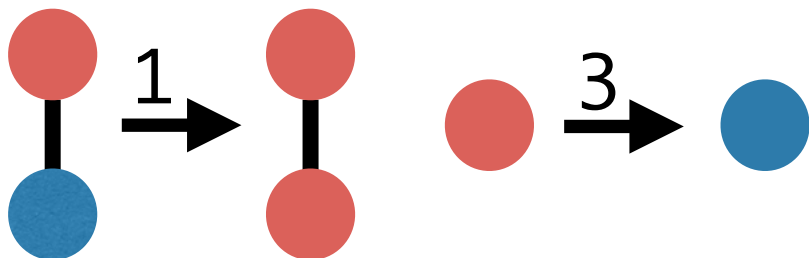




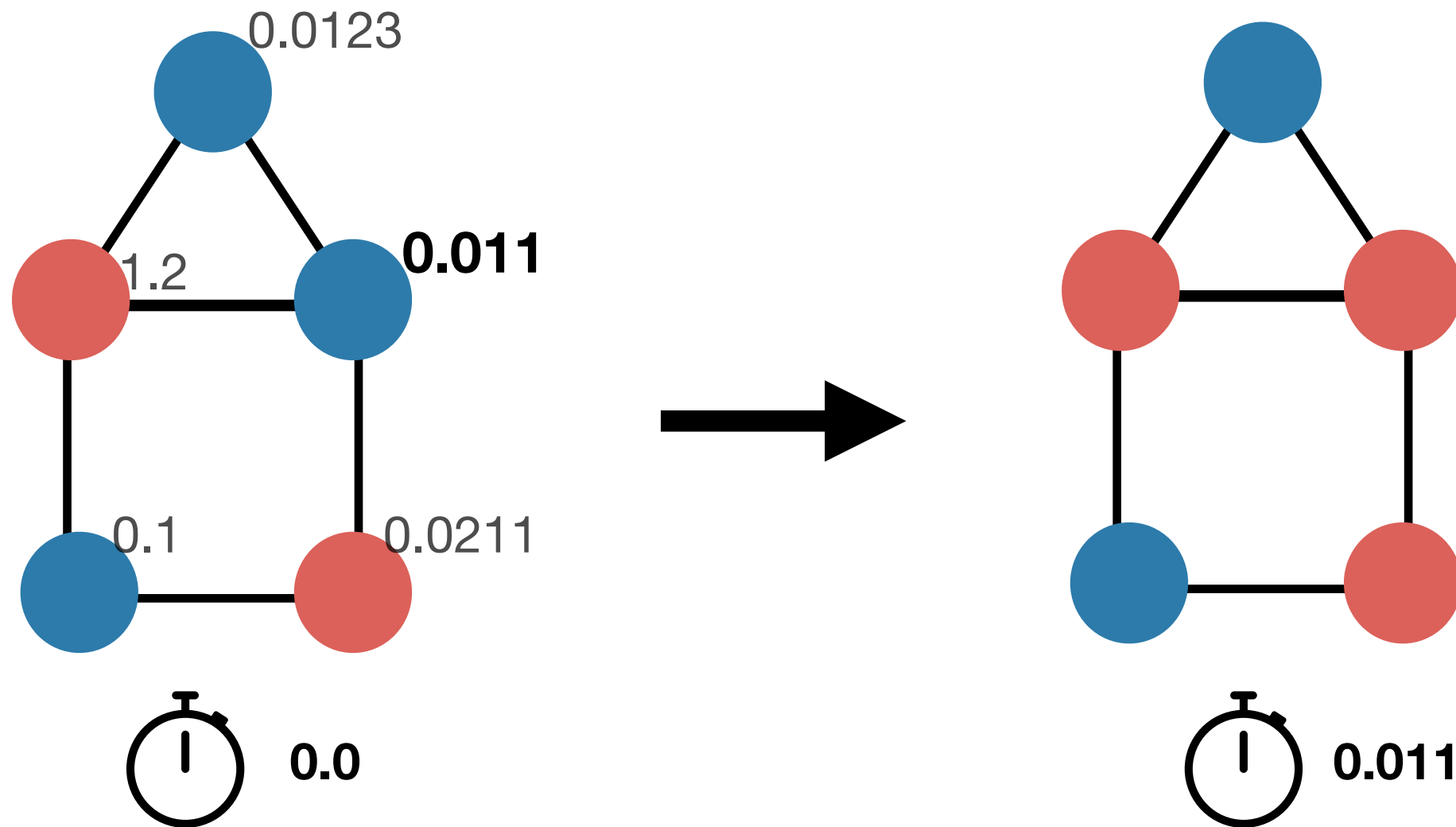
# SIS Naïve Simulation



Rules:



# SIS Naïve Simulation



:( Iteration over all nodes is very slow.

# Stochastic Simulation Approaches

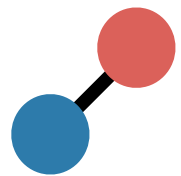
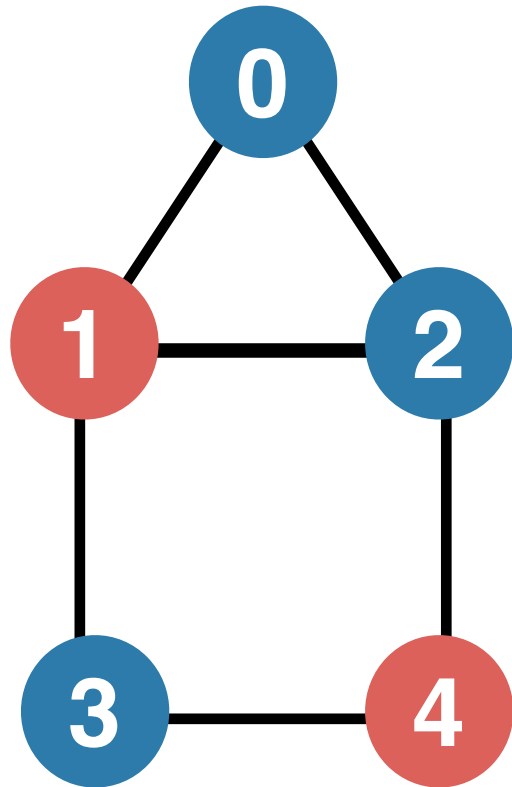
- Standard Gillespie Algorithm
- Optimized Gillespie Algorithm
- Event-Based Rejection Simulation (Our Method)

# Stochastic Simulation Approaches

- **Standard Gillespie Algorithm**
- Optimized Gillespie Algorithm
- Event-Based Rejection Simulation (Our Method)



# Standard Gillespie

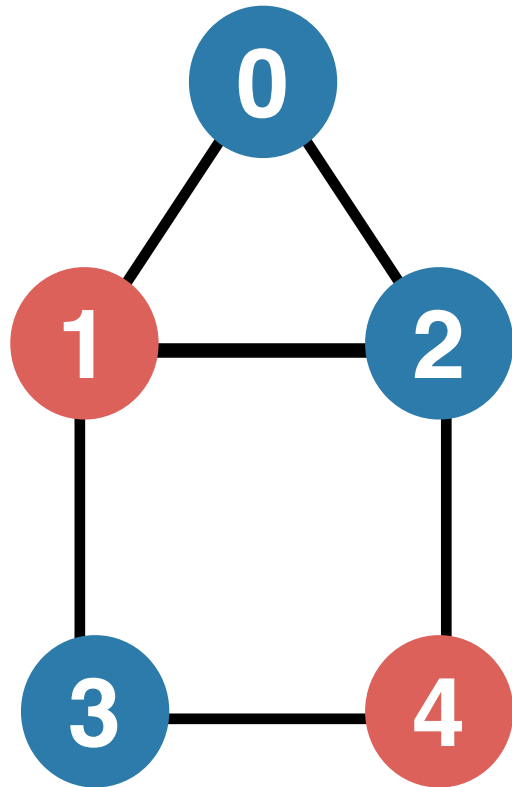


$[0-1, 1-3, 1-2, 2-4, 3-4]$



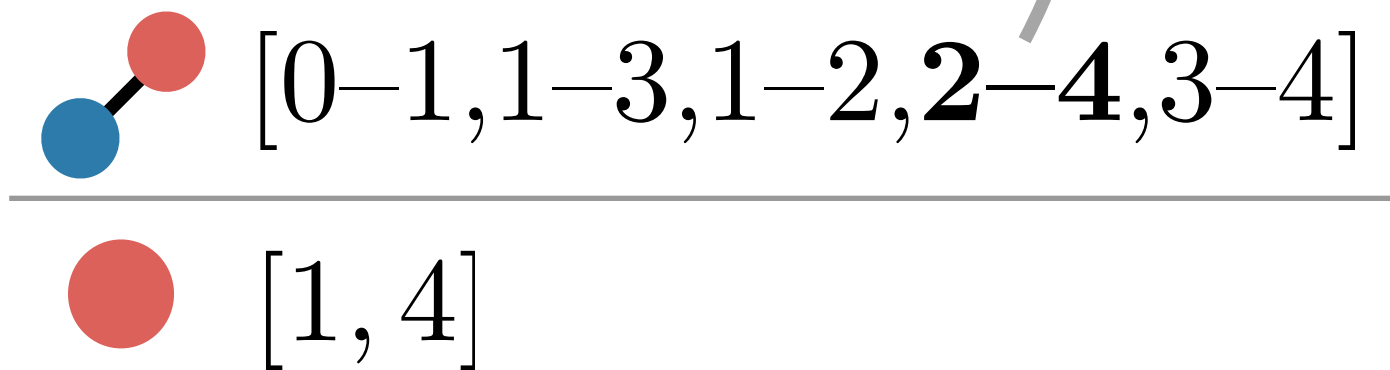
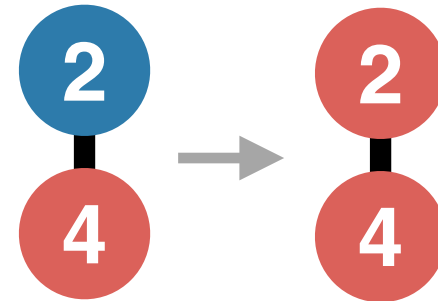
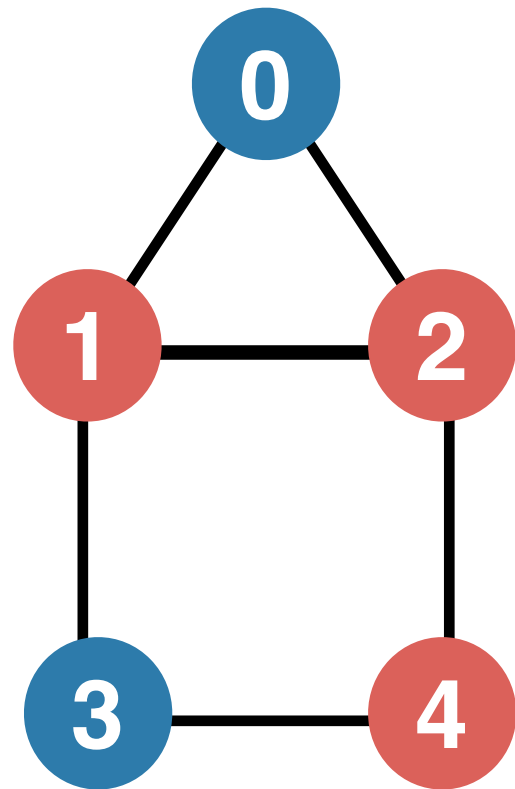
$[1, 4]$

# Standard Gillespie



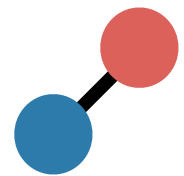
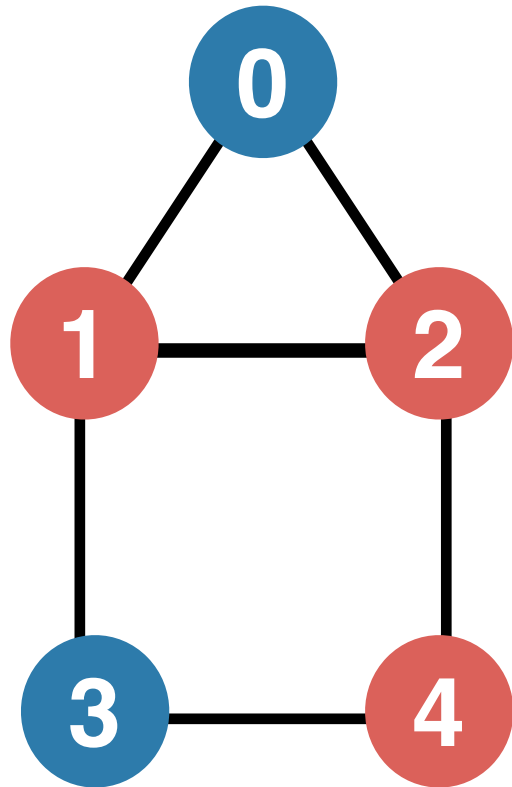
$$\begin{array}{c}
 \text{blue} \text{---} \text{red} \\
 \hline
 \text{red}
 \end{array}
 \frac{[0-1, 1-3, 1-2, 2-4, 3-4]}{[1, 4]} \rightarrow 2\mu + 5\lambda$$

# Standard Gillespie



$$\rightarrow 2\mu + 5\lambda$$

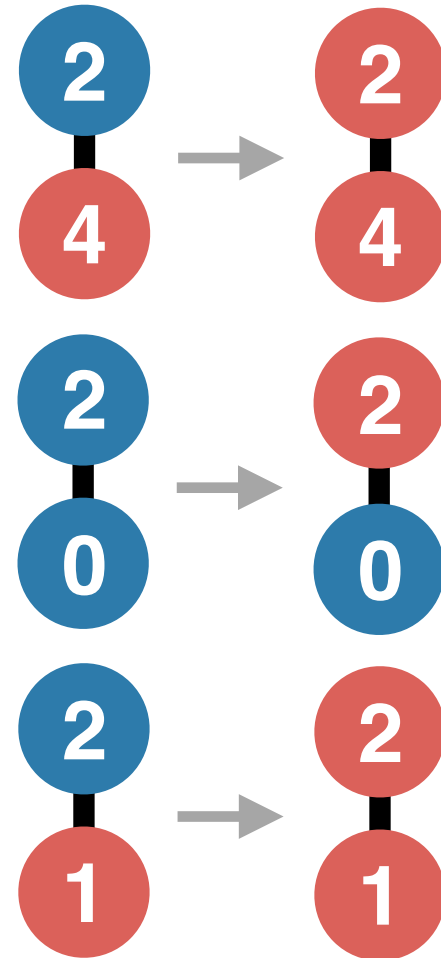
# Standard Gillespie



$[0-1, 1-3, 1-2, \mathbf{2-4}, 3-4]$



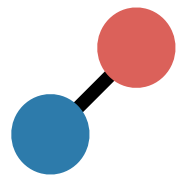
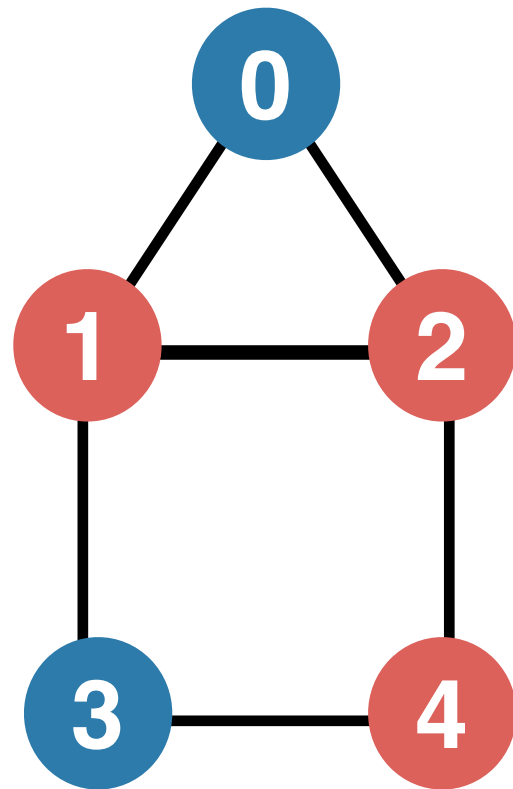
$[1, 4]$



$$\rightarrow 2\mu + 5\lambda$$



# Standard Gillespie

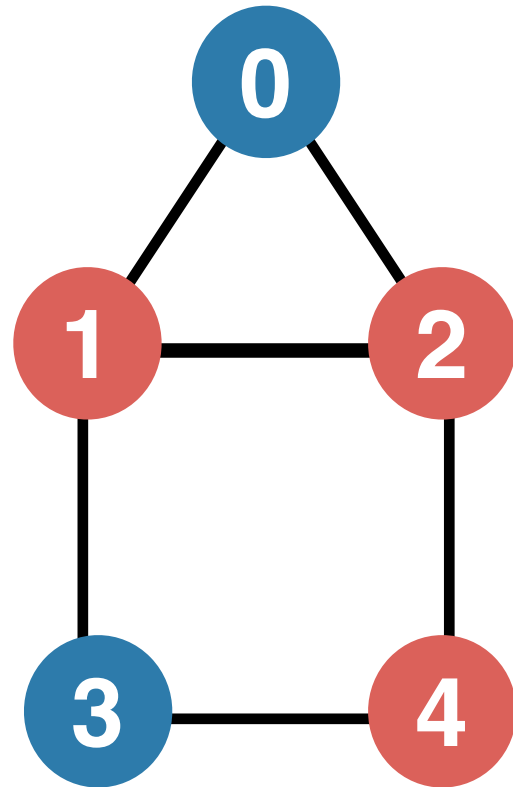


$[0-1, 0-2, 1-3, 3-4]$



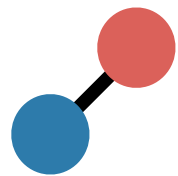
$[1, 2, 4]$

# Standard Gillespie



:(

Iteration over all neighbours of the updated node is needed.



$[0-1, 0-2, 1-3, 3-4]$

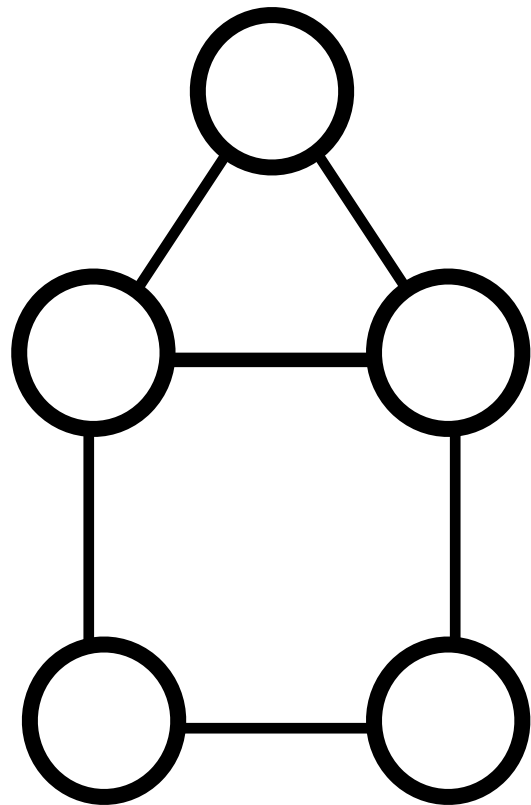


$[1, 2, 4]$

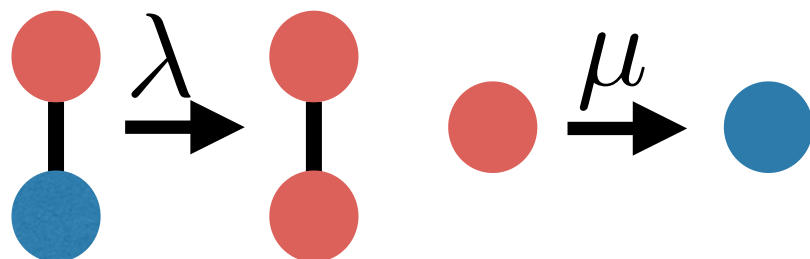
# Stochastic Simulation Approaches

- Standard Gillespie Algorithm
- **Optimized Gillespie Algorithm**
- Event-Based Rejection Simulation (Our Method)

# Optimized Gillespie

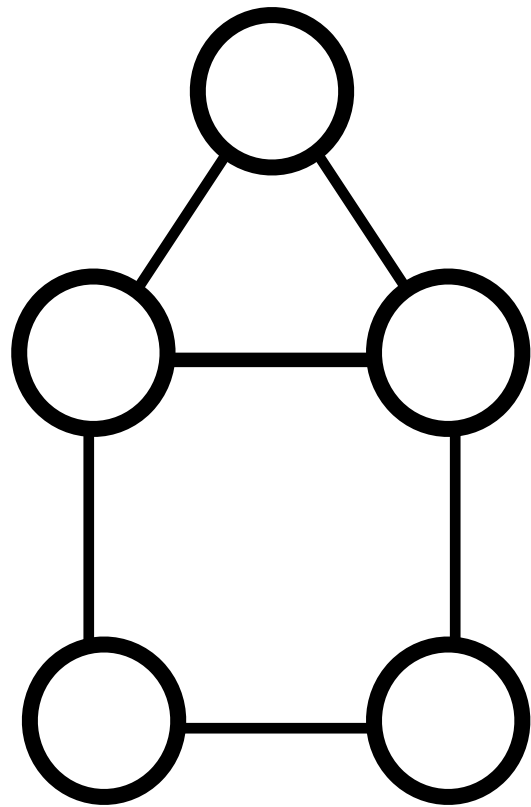


What is the maximal rate at which an **infected** node attacks its neighbours?  
(under all network configurations)



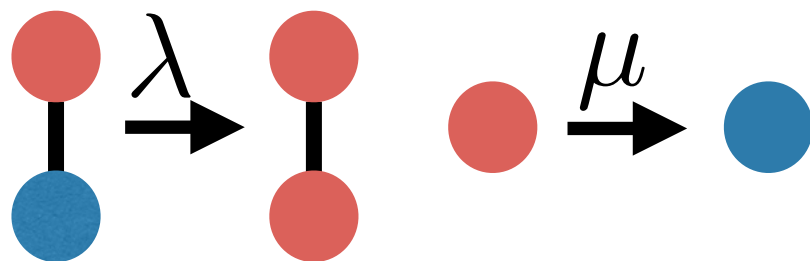
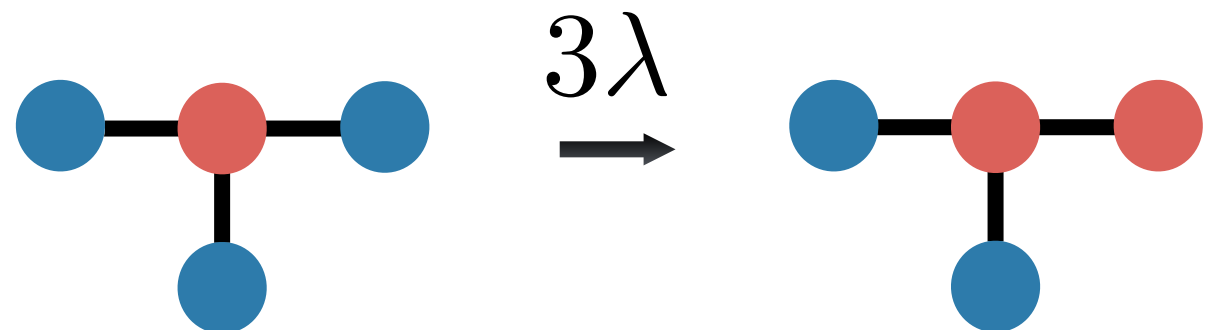


# Optimized Gillespie

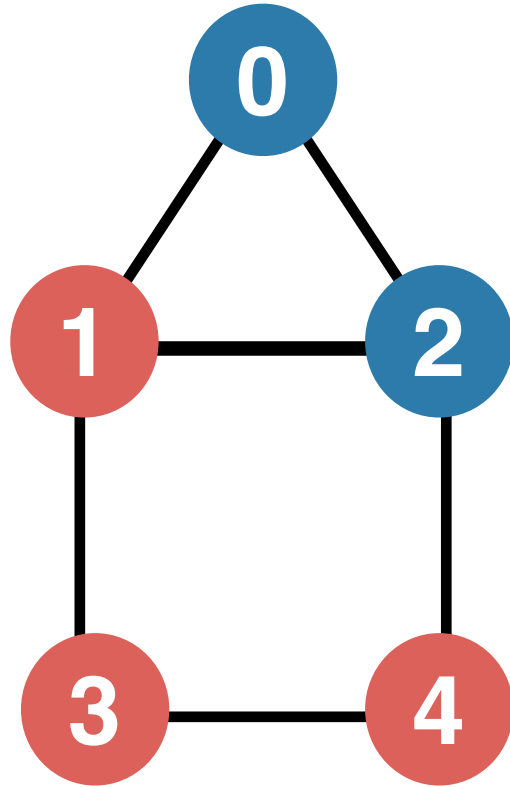


What is the maximal rate at which an **infected** node attacks its neighbours?  
(under all network configurations)

Maximal degree: 3

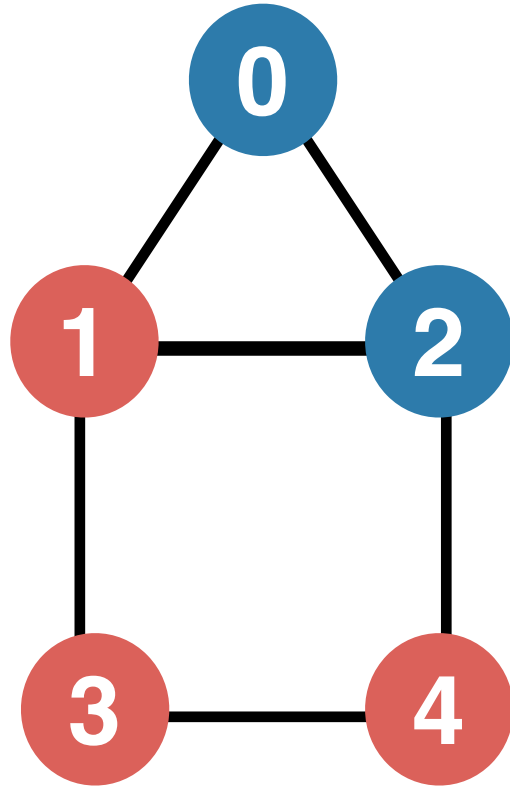


# Optimized Gillespie



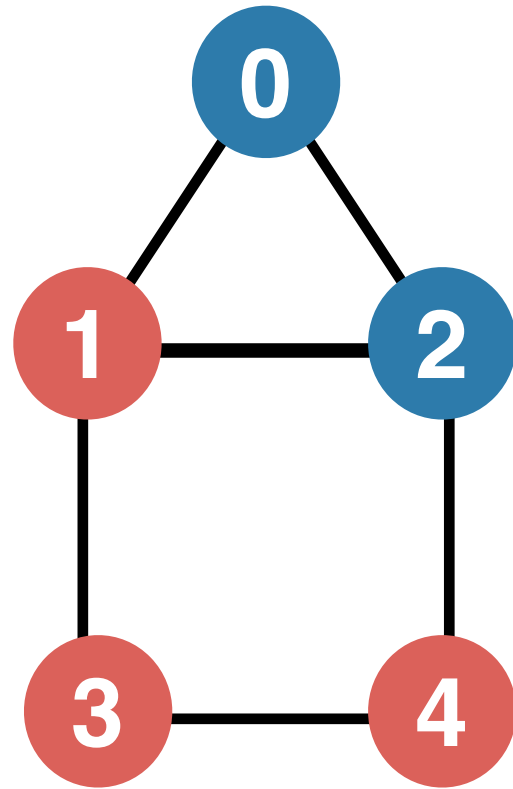
●  $[1, 3, 4]$

# Optimized Gillespie



●  $[1, 3, 4] \rightarrow 3\mu + 3 \cdot (3\lambda)$  (upper bound)

# Optimized Gillespie

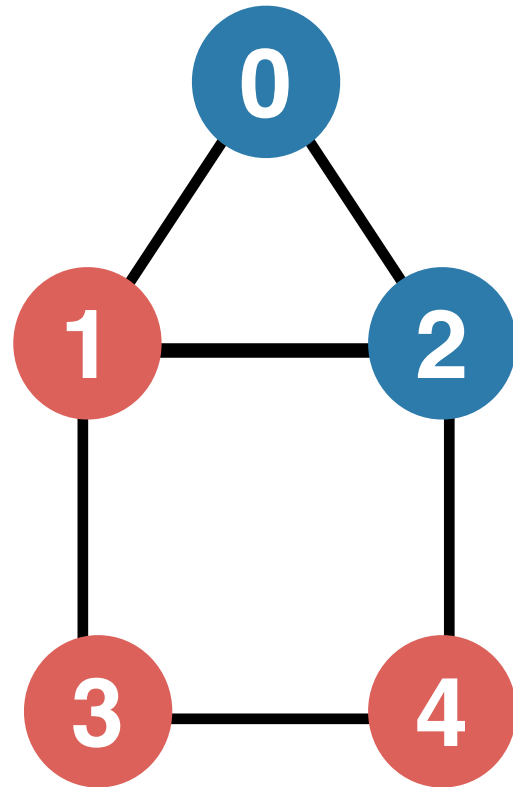


Actual (effective) rate:  $\lambda$

Upper bound:  $3\lambda$

$$\bullet [1, 3, 4] \rightarrow 3\mu + 3 \cdot (3\lambda) \quad (\text{upper bound})$$

# Optimized Gillespie



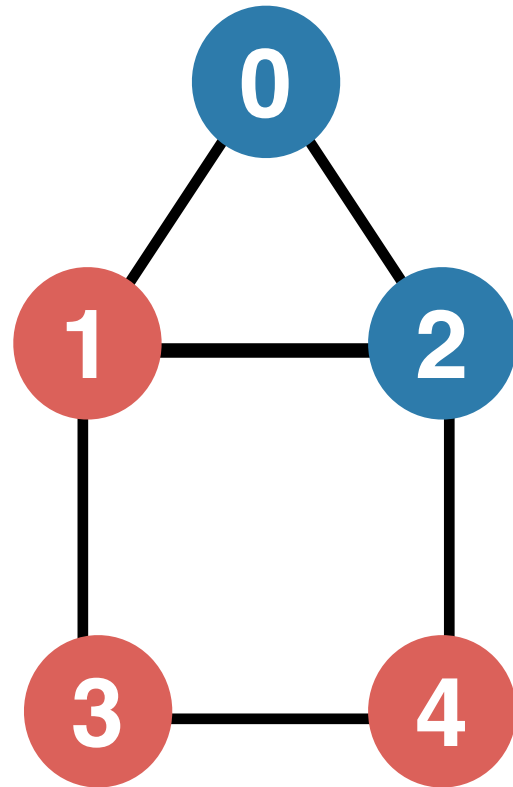
Actual (effective) rate:  $\lambda$

Upper bound:  $3\lambda$

Rejection step with probability:  $\frac{2}{3}$

$$\bullet [1, 3, 4] \rightarrow 3\mu + 3 \cdot (3\lambda) \quad (\text{upper bound})$$

# Optimized Gillespie



Actual (effective) rate:  $\lambda$

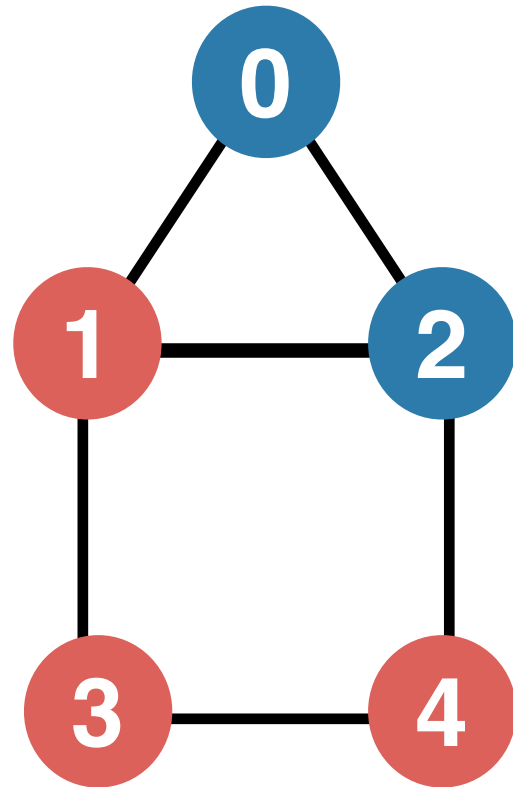
Upper bound:  $3\lambda$

Rejection step with probability:  $\frac{2}{3}$

$$\text{Rejection probability} = \frac{\text{Maximal rate} - \text{Effective rate}}{\text{Maximal rate}}$$

$$\bullet [1, 3, 4] \rightarrow 3\mu + 3 \cdot (3\lambda) \quad (\text{upper bound})$$

# Optimized Gillespie



Actual (effective) rate:  $\lambda$

Upper bound:  $3\lambda$

Rejection step with probability:  $\frac{2}{3}$

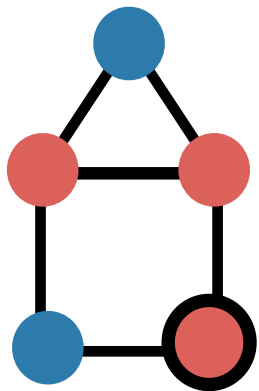
$$\text{Rejection probability} = \frac{\text{Maximal rate} - \text{Effective rate}}{\text{Maximal rate}}$$

:) Not necessary to update whole neighbourhood of node 4

:( Potentially large number of rejections



# Rejection Probabilities



Upper bound  
 $3\lambda$

Maximal degree: 3  
Effective degree: 2



Degree: 2  
Susceptible neighbors: 1

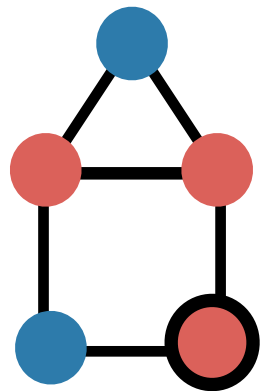


Reject with probability  $\frac{1}{3}$



Reject with probability  $\frac{1}{2}$

# Rejection Probabilities



Upper bound  
 $3\lambda$

Maximal degree: 3  
Effective degree: 2



Degree: 2  
Susceptible neighbors: 1



Reject with probability  $\frac{1}{3}$



Reject with probability  $\frac{1}{2}$



Rejection step with probability:  $\frac{2}{3}$

# Rejection Probabilities

Maximal degree: 3  
Effective degree: 2

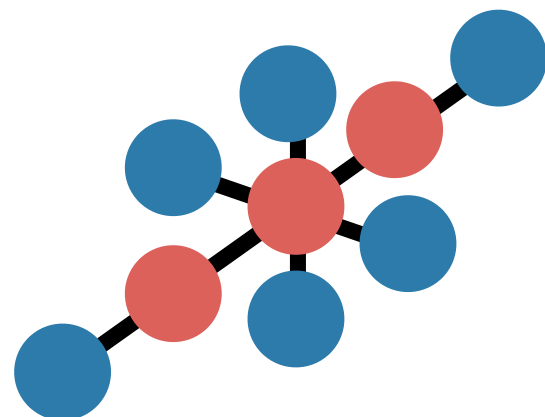


Degree: 2  
Susceptible neighbors: 1



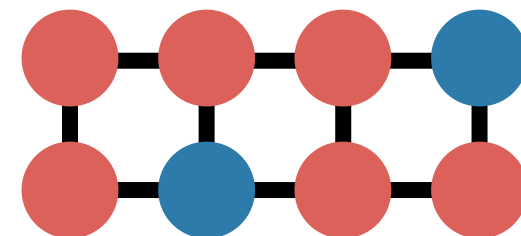
Reject with probability  $\frac{1}{3}$

(high rejection probability when  
degree differences are large)



Reject with probability  $\frac{1}{2}$

(high rejection probability for  
many infected nodes)



# Stochastic Simulation Approaches

- Standard Gillespie Algorithm
- Optimized Gillespie Algorithm
- **Event-Based Rejection Simulation (Our Method)**

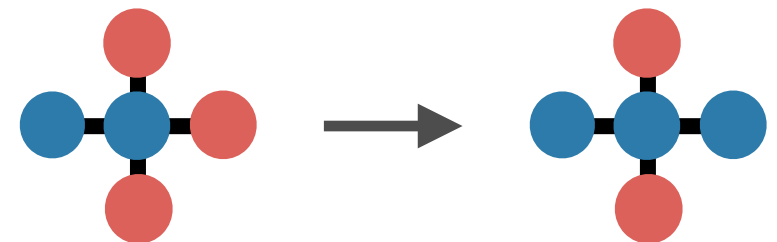
# Event-Driven Simulation

In each step

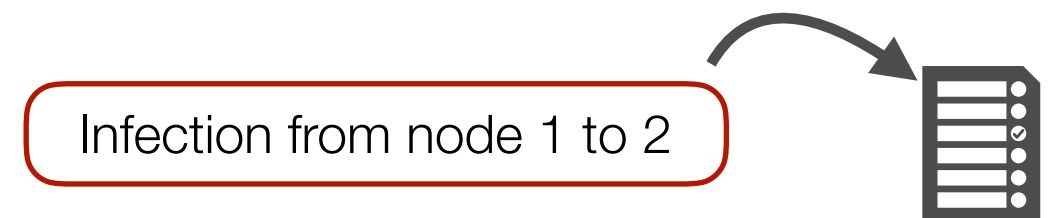
1. Take first event from event queue.



2. Apply event to the network.



3. Generate new event(s).



# Event-Driven Simulation

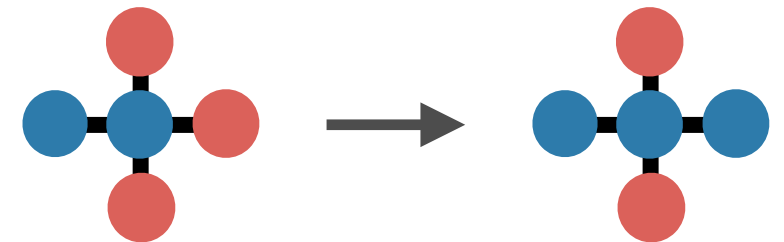
## In each Step

1. Take first event from event queue.

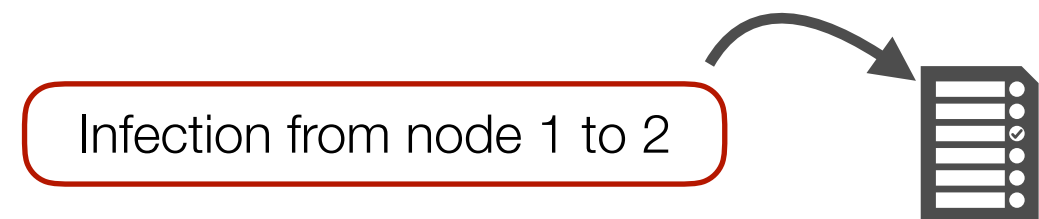


2. Check if event is applicable to network:

A. If Yes: apply event - Else: ignore



3. Generate new event(s).



# Event-Driven Simulation

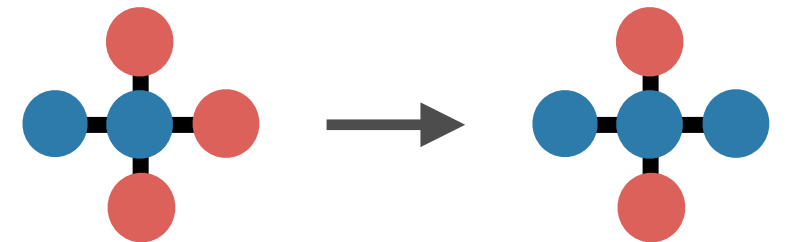
## In each Step

1. Take first event from event queue.



2. Check if event is applicable to network:

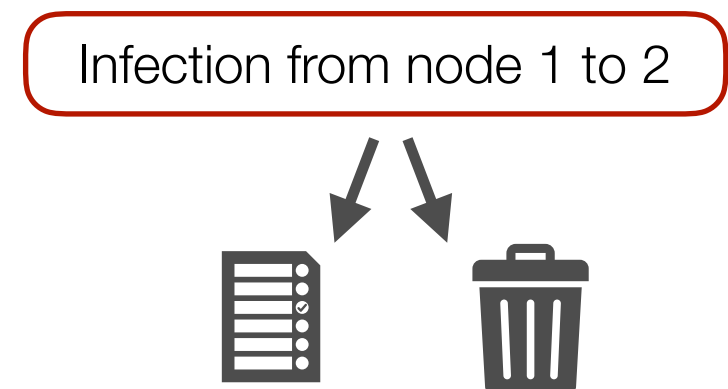
A. If Yes: apply event - Else: ignore



3. Generate new event(s) and check the events will be rejected later.

A. If Yes: go back to 3

B. Else: add event(s) to queue



# Event-Driven Simulation

## In each Step

1. Take first event from event queue.
2. Check if event is applicable to network:
  - A. If Yes: apply event - **Else: ignore**
3. Generate new event(s) and check the events will be rejected later.
  - A. If Yes: **go back to 3**
  - B. Else: add events(s) to queue

Late  
Reject

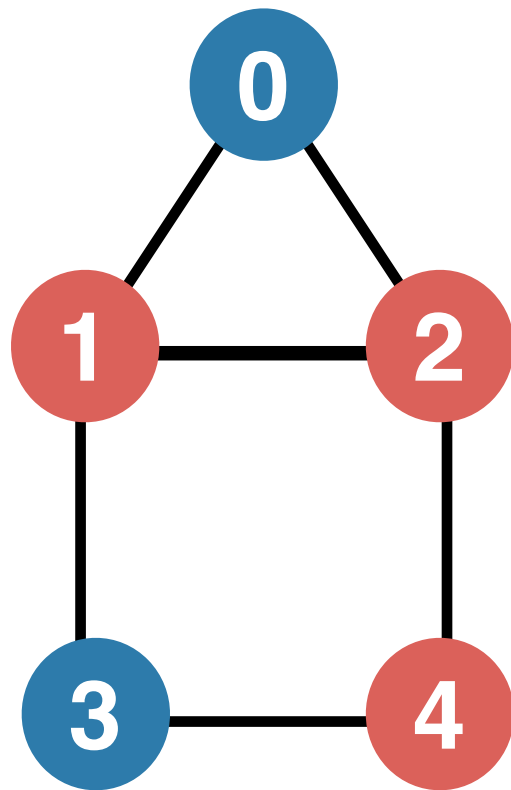
Early  
Reject



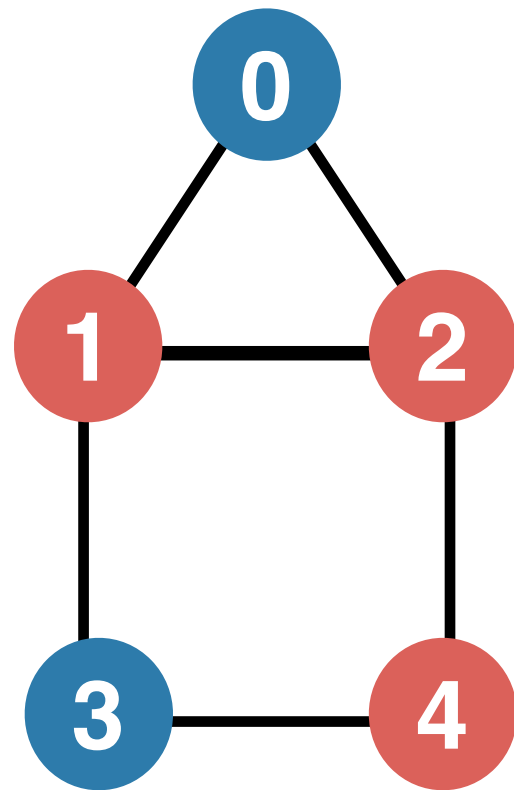
# Algorithm Outline

## Initialization

1. Generate a curing event for each infected node + annotate node with its curing time
2. Generate an infection attempt for each infected node



# Algorithm Outline



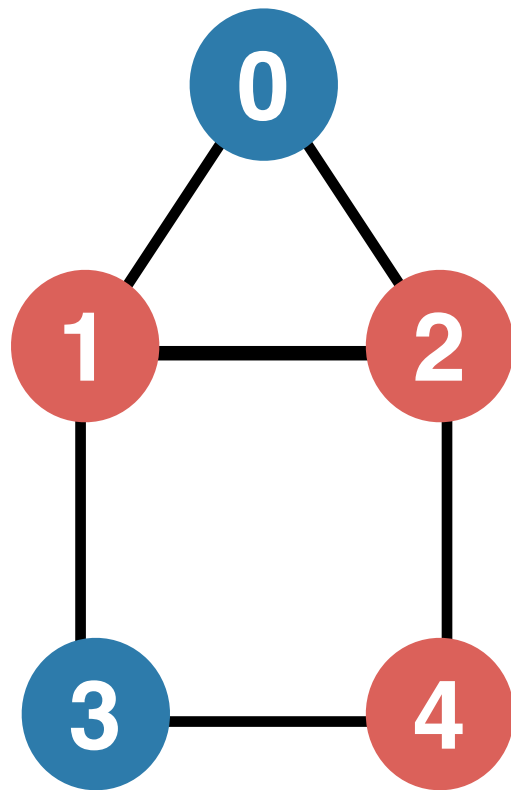
## Initialization

1. Generate a curing event for each infected node + annotate node with its curing time
2. Generate an infection attempt for each infected node

## In each step

1. Take first event from queue
2. If event is applicable:
  1. Change node states acc. to event
  2. Generate two new events in case of infection
3. Else (Late Rejection):
  1. Generate new infection attempt

# Algorithm Outline



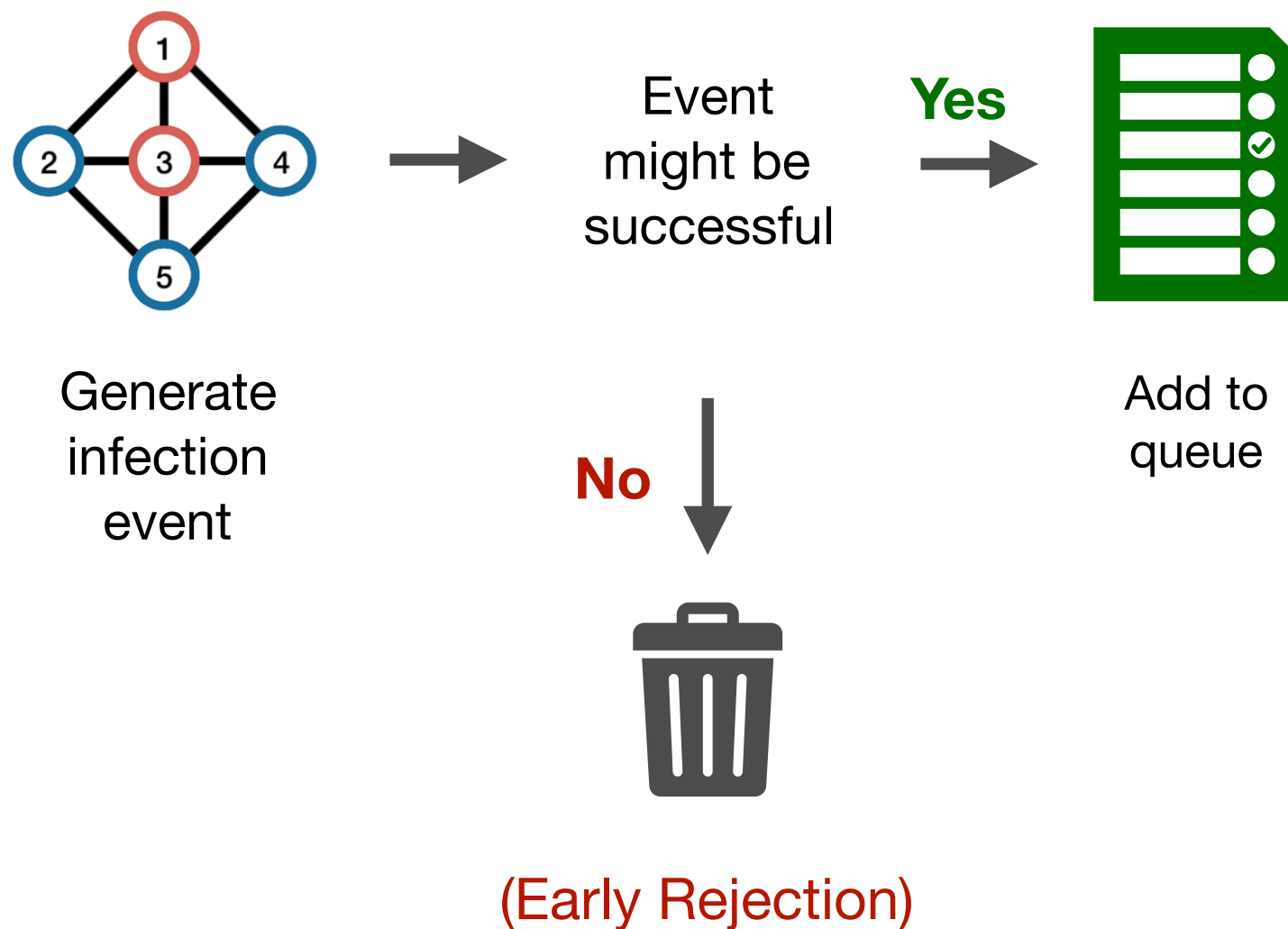
## Initialization

1. Generate a curing event for each infected node + annotate node with its curing time
2. **Generate an infection attempt** for each infected node

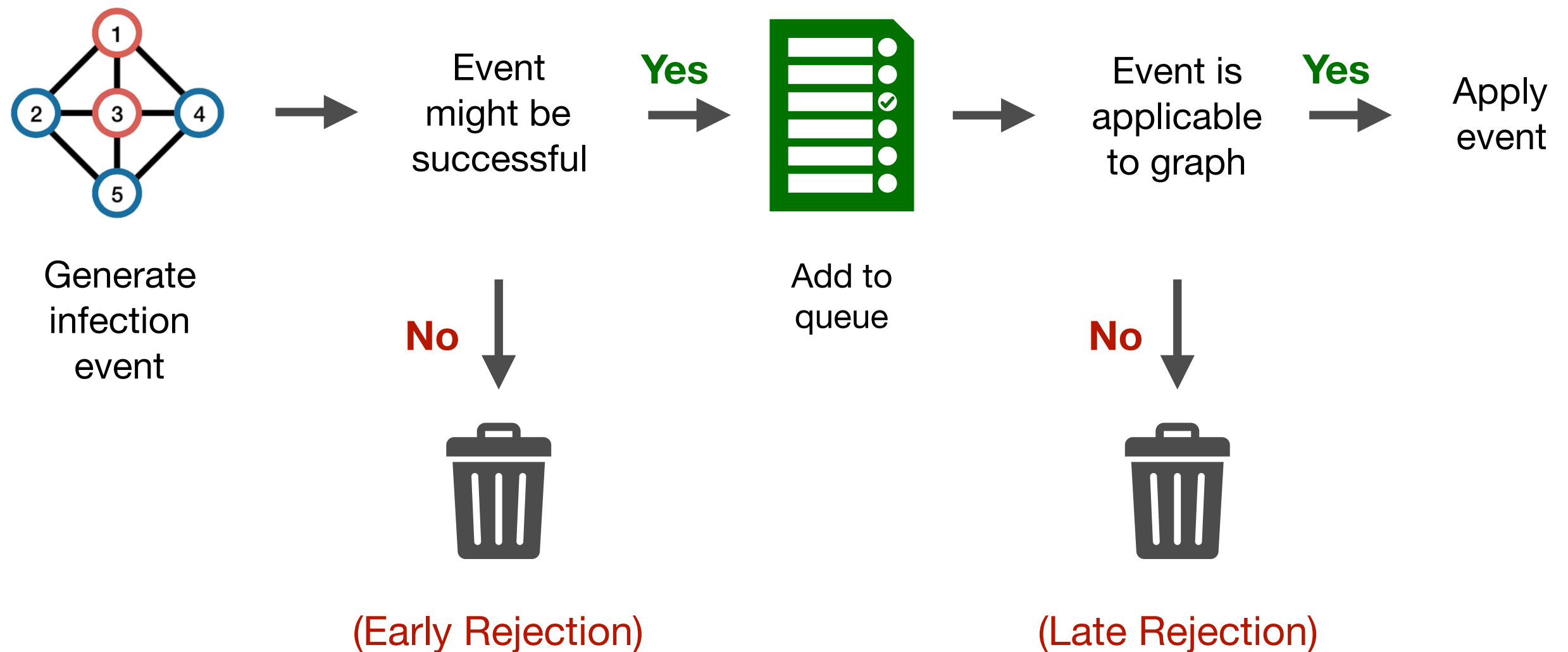
## In each step

1. Take first event from queue
2. If event is applicable:
  1. Change node states acc. to event
  2. **Generate two new events** in case of infection
3. Else (Late Rejection):
  1. **Generate new infection attempt**

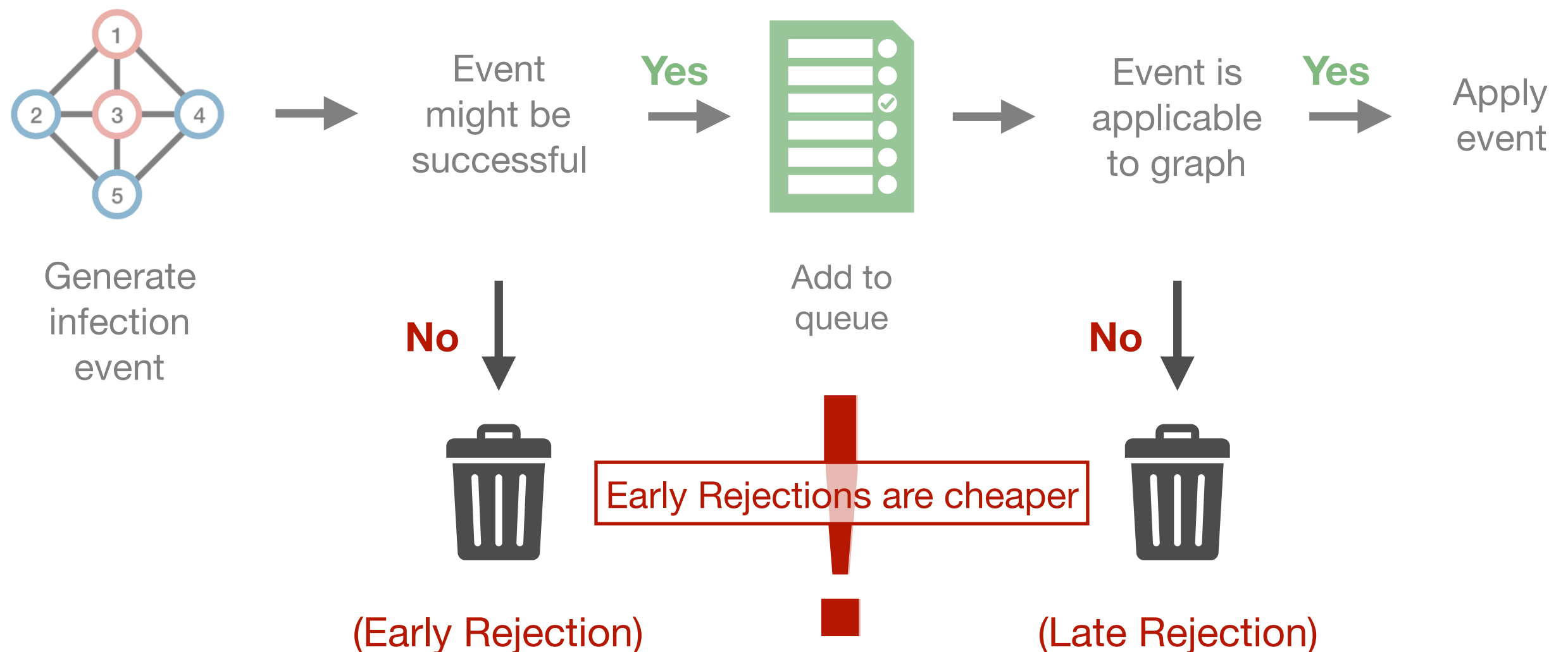
# Early and Late Rejections



# Early and Late Rejections

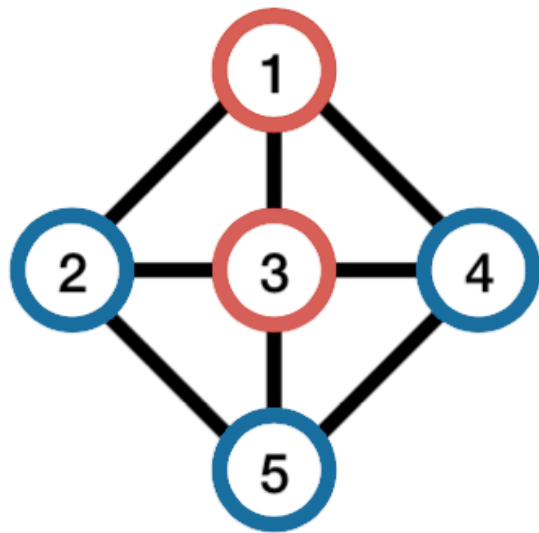


# Early and Late Rejections



# Sample Run

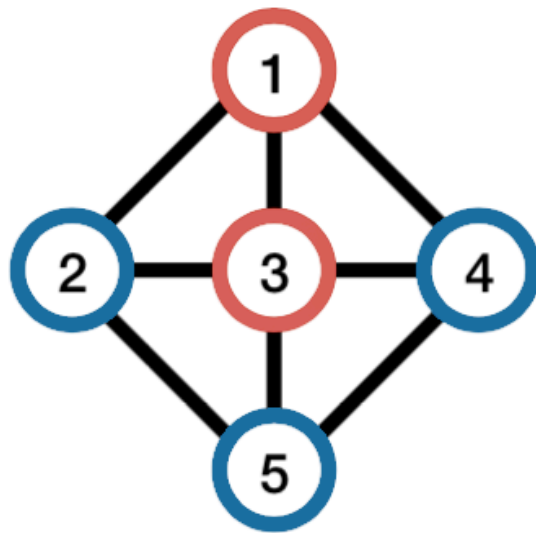
(Initialization)



t	Event
1.6	Recovery Node: 1
1.7	Recovery Node: 3


# Sample Run

(Initialization)



t	Event
1.6	Recovery Node: 1
1.7	Recovery Node: 3

t	Event
0.1	Infection Edge: 1 → 3

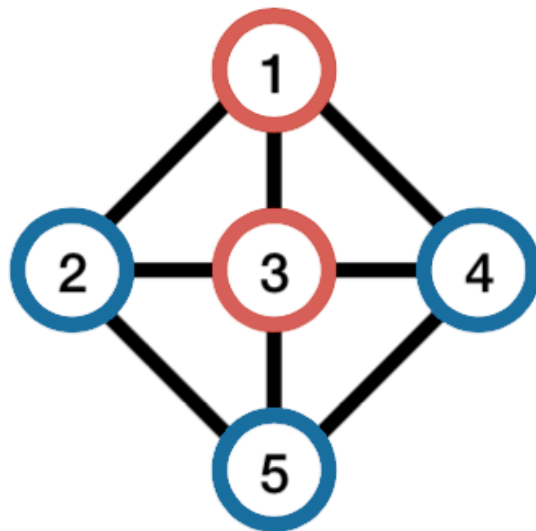


t	Event
0.3	Infection Edge: 1 → 4
0.4	Infection Edge: 3 → 4
1.6	Recovery Node: 1
1.7	Recovery Node: 3



# Sample Run

(Initialization)



t	Event
1.6	Recovery Node: 1
1.7	Recovery Node: 3

3 will still be  
infected

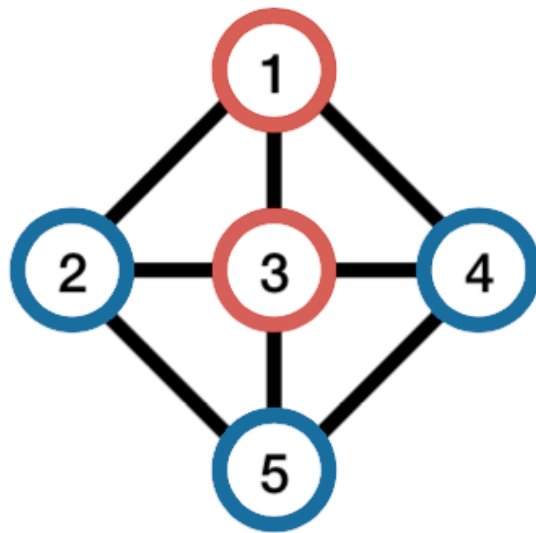
t	Event
0.1	Infection Edge: 1 → 3



t	Event
0.3	Infection Edge: 1 → 4
0.4	Infection Edge: 3 → 4
1.6	Recovery Node: 1
1.7	Recovery Node: 3

# Sample Run

(Initialization)



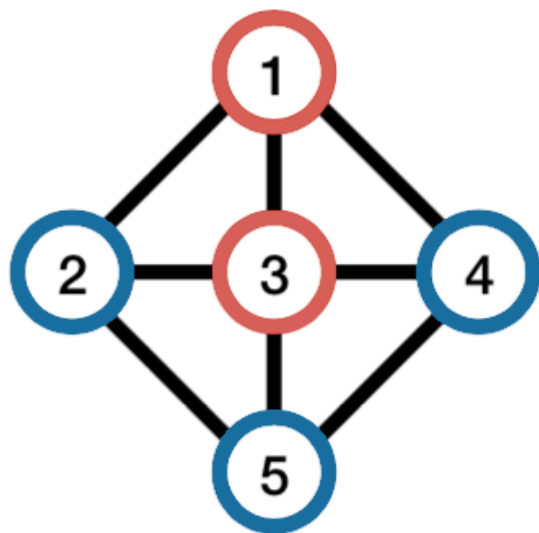
t	Event
1.6	Recovery Node: 1
1.7	Recovery Node: 3

t	Event
0.1	Infection Edge: 1 → 3
0.3	Infection Edge: 1 → 4
0.4	Infection Edge: 3 → 4
1.6	Recovery Node: 1
1.7	Recovery Node: 3

2nd event

# Sample Run

(Initialization)



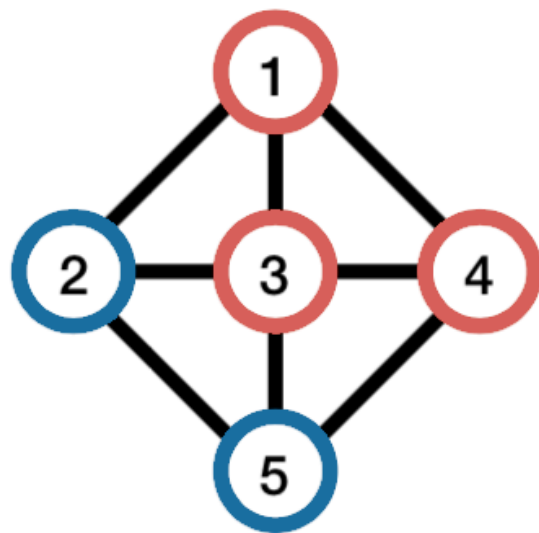
t	Event
1.6	Recovery Node: 1
1.7	Recovery Node: 3

t	Event
0.1	Infection Edge: 1 → 3
0.3	Infection Edge: 1 → 4
0.4	Infection Edge: 3 → 4
1.6	Recovery Node: 1
1.7	Recovery Node: 3

Apply this event

# Sample Run

(Iteration)

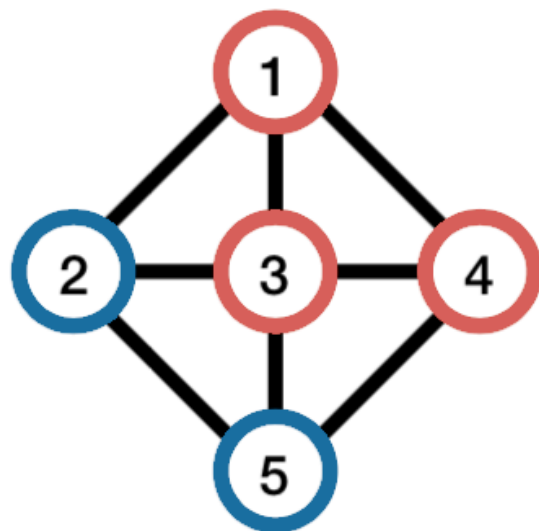


Recovery  
of 4

t	Event
0.4	Infection Edge: 3→4
0.5	Recovery Node: 4
1.6	Recovery Node: 1
1.7	Recovery Node: 3

# Sample Run

(Iteration)



t	Event
0.4	Infection Edge: 3→4
0.5	Recovery Node: 4
1.6	Recovery Node: 1
1.7	Recovery Node: 3

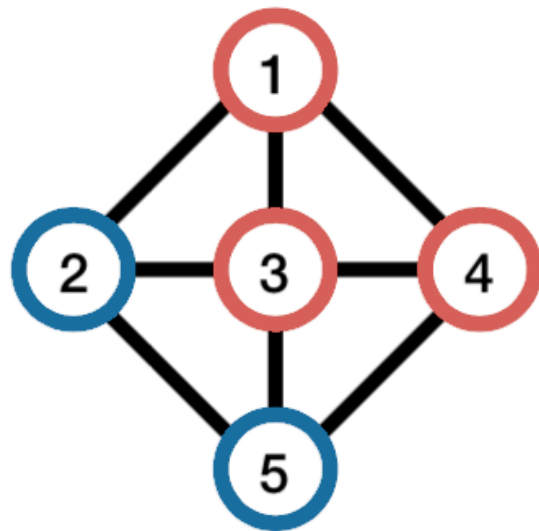
t	Event
0.9	Infection Edge: 4→5



t	Event
0.4	Infection Edge: 3→4
0.5	Recovery Node: 4
0.7	Infection Edge: 1→2
1.6	Recovery Node: 1
1.7	Recovery Node: 3

# Sample Run

(Iteration)



t	Event
0.4	Infection Edge: 3→4
0.5	Recovery Node: 4
1.6	Recovery Node: 1
1.7	Recovery Node: 3

4 will already be  
recovered already

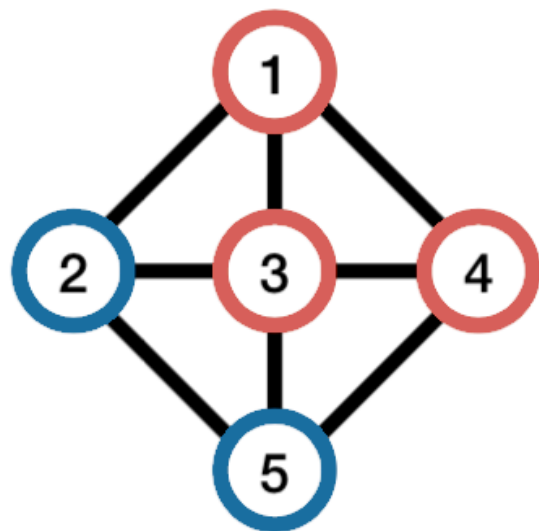
t	Event
0.9	Infection Edge: 4→5




t	Event
0.4	Infection Edge: 3→4
0.5	Recovery Node: 4
0.7	Infection Edge: 1→2
1.6	Recovery Node: 1
1.7	Recovery Node: 3

# Sample Run

(Iteration)

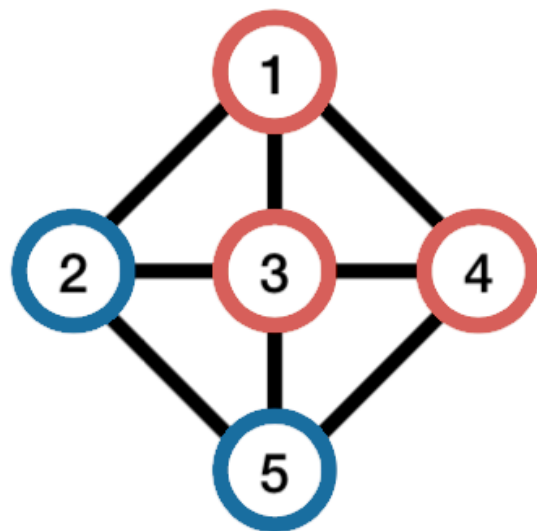


t	Event
0.4	Infection Edge: 3→4 
0.5	Recovery Node: 4
0.7	Infection Edge: 1→2
1.6	Recovery Node: 1
1.7	Recovery Node: 3

Event not applicable (late rejection)

# Sample Run

(Iteration)



t	Event
0.4	Infection Edge: 3→4 🗑️
0.5	Recovery Node: 4
0.7	Infection Edge: 1→2
1.6	Recovery Node: 1
1.7	Recovery Node: 3

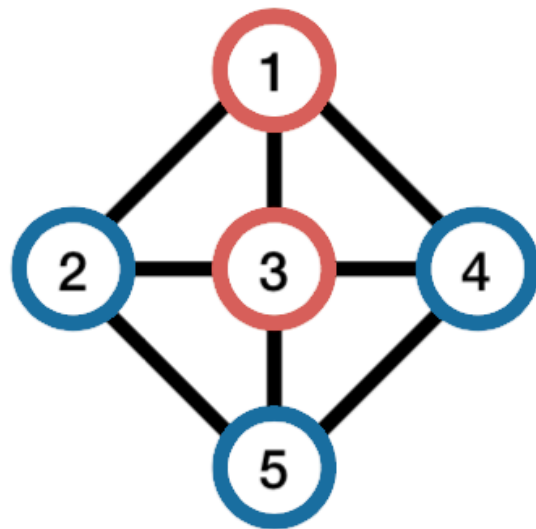
Generate new infection attempt

t	Event
0.5	Recovery Node: 4
0.6	Infection Edge: 3→4
0.7	Infection Edge: 1→2
1.6	Recovery Node: 1
1.7	Recovery Node: 3



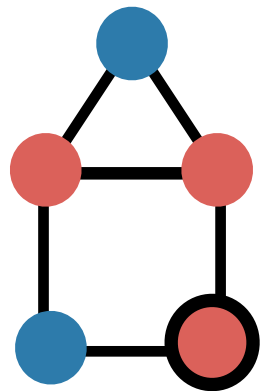
# Sample Run

(Iteration)



t	Event
0.6	Infection Edge: 3→4
0.7	Infection Edge: 1→2
1.6	Recovery Node: 1
1.7	Recovery Node: 3

# Rejection Probabilities



Upper bound  
 $3\lambda$

Maximal degree: 3  
Effective degree: 2



Degree: 2  
Susceptible neighbors: 1



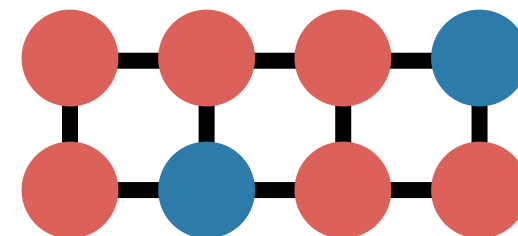
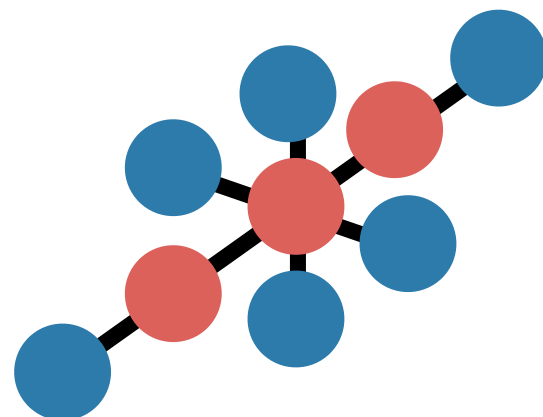
Reject with probability  $\frac{1}{3}$



Reject with probability  $\frac{1}{2}$

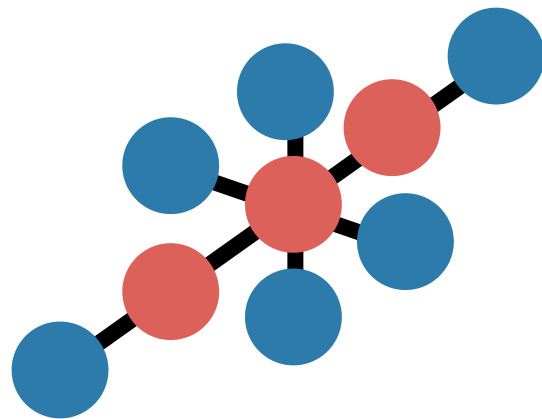
(high rejection probability when  
degree differences are large)

(high rejection probability for  
many infected nodes)



# Rejection Probabilities

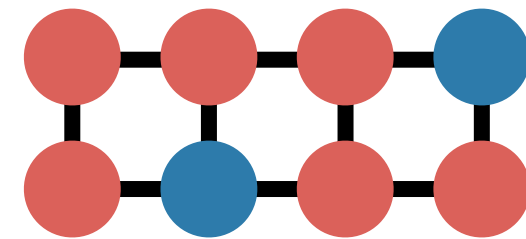
The degree differences are large



Event queue

(Getting the next event is constant regardless of degree)

There are many infected nodes



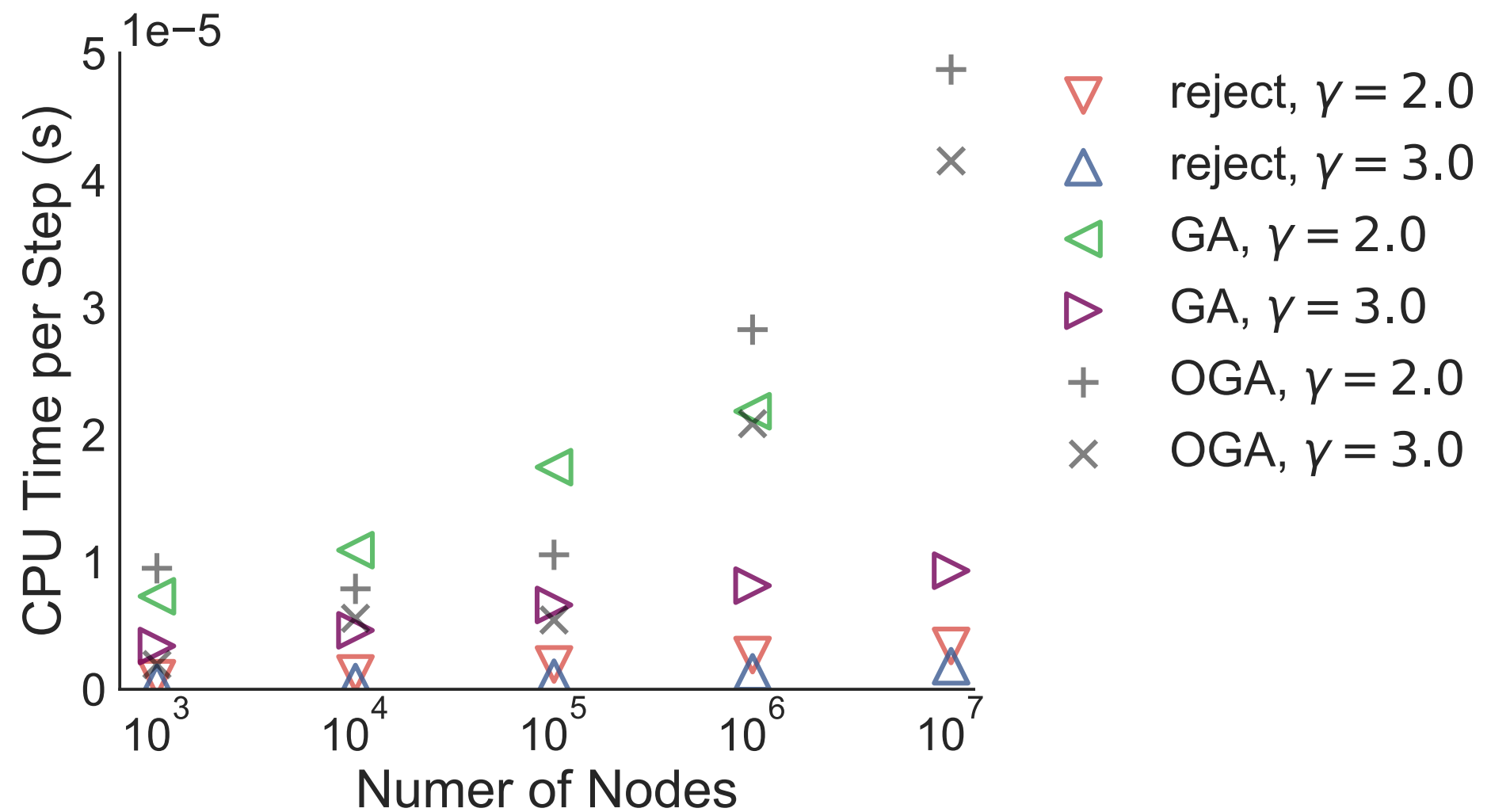
Early Rejections

(Infections towards infected neighbours can be discarded early on)

# Generalizations

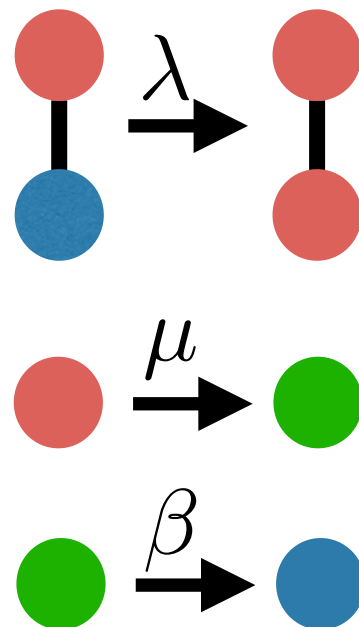
- Possible for most epidemic models (but not for all)
- Easy for weighted networks
- Easy for temporal networks (for external process which alters network)
- Possible for non-Markovian dynamics (depends on formalism)

# Results (SIS)

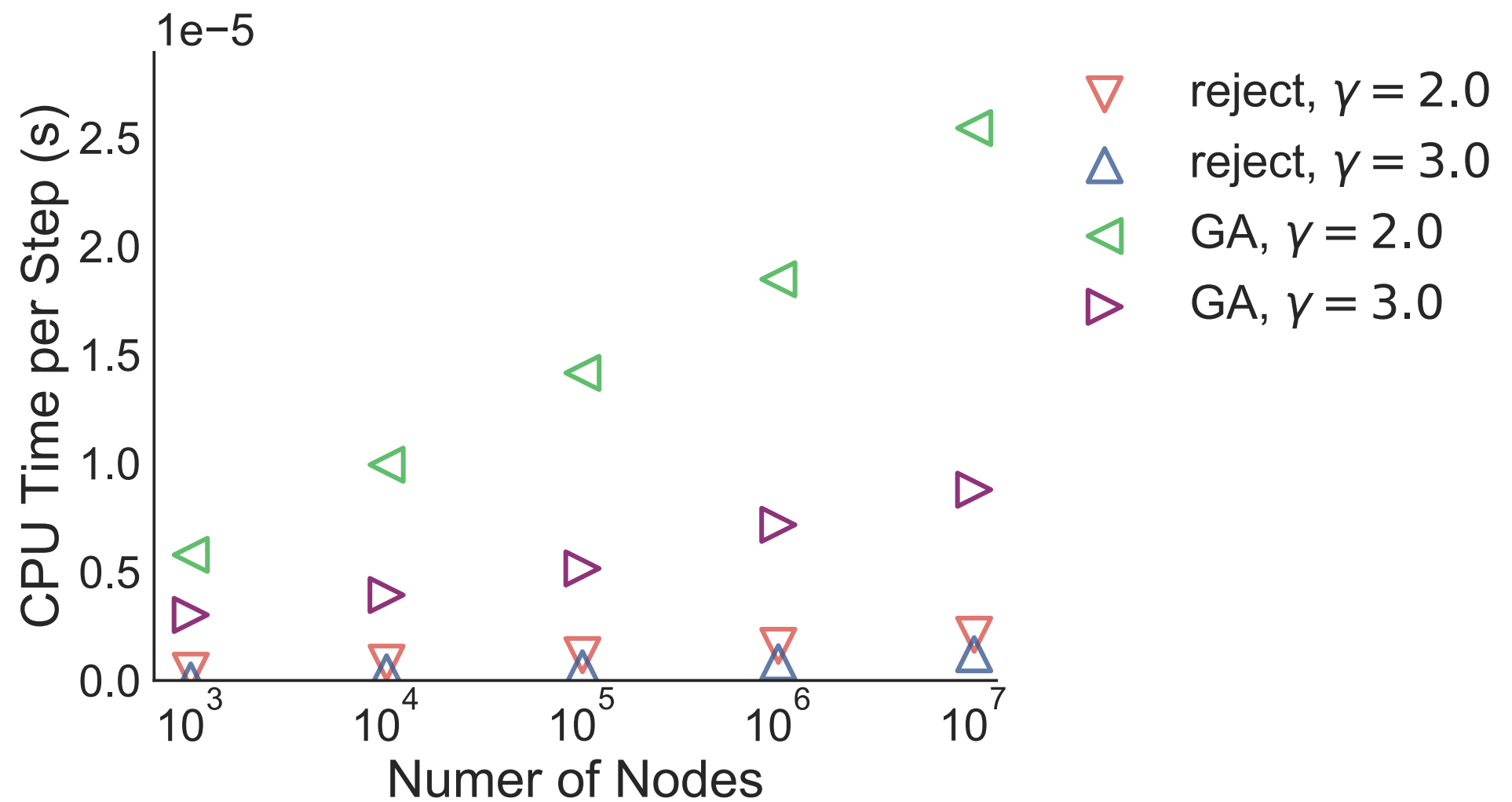


# Results (SIR)

**Infected** nodes become **recovered** before becoming **susceptible** again.

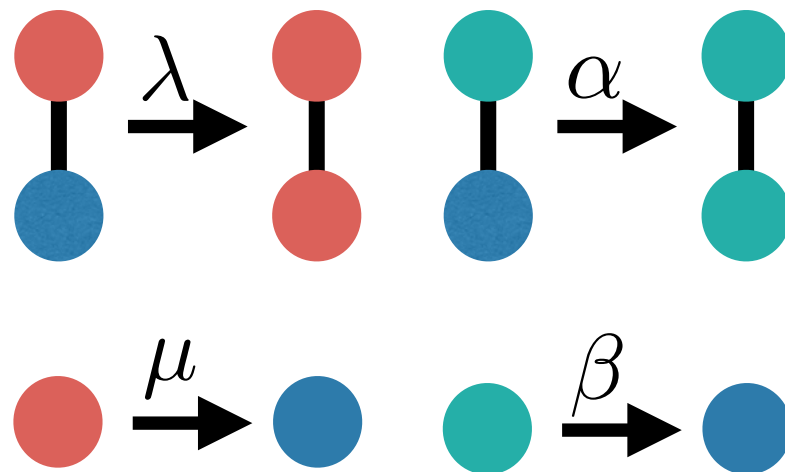


# Results (SIR)



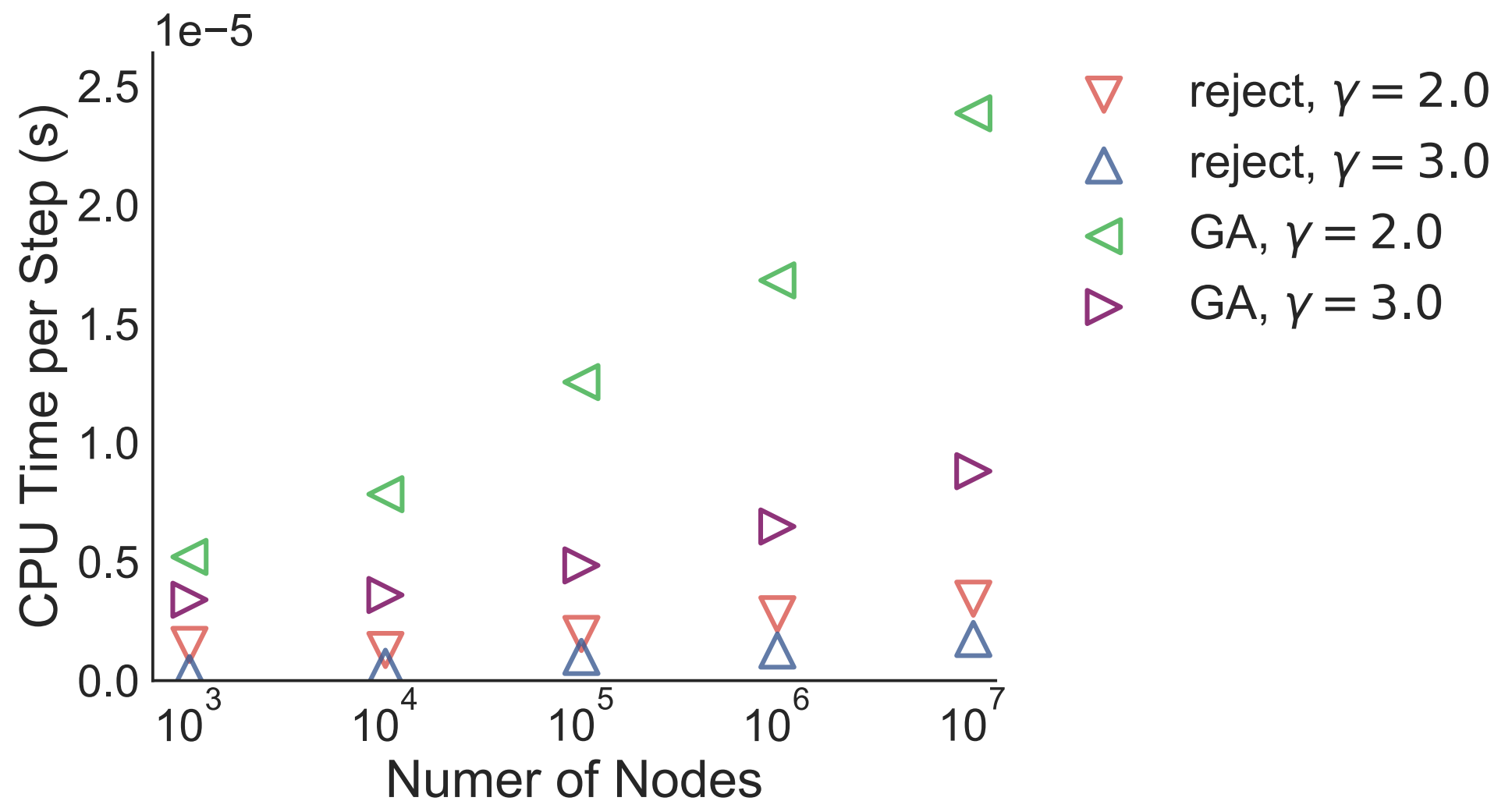
# Results (competing pathogens)

**Pathogen I** and **Pathogen II** compete over **susceptible** nodes.





# Results (competing pathogens)



# Conclusion

- Not Iterating over a node's neighbourhood yields huge performance improvements
- Event-based simulation reduces rejection steps significantly
- Exploit problem structure for early rejections
- Event-based simulation is very flexible and easy to adopt to different formalisms

*Thank you*

# Correctness (Sketch)

- Same as for Optimized Gillespie
- Add shadow rule to system
- Interpret rejections not als rejections but as applications of the shadow rule
- Joint rate attributed to infection and shadow rule is always constant

