Transition Systems

(Guest) **Lecture 17**

James Wilcox

CERTORA

5 March 2020

About me

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

PLSE

CERTORA

Transition Systems

Concurrency and more

What are T.S. good for?

So far, we've been verifying sequential programs

What about multithreaded or networked programs?

Thread 1	Thread 2
`x = 0`	**`x = 1`**
`assert x == 0`	**`assert x == 1`**

What are T.S. good for?

So far, we've been verifying sequential programs

What about multithreaded or networked programs?

```
Thread 1              Thread 2
x = 0
assert x == 0

                      x = 1
                      assert x == 1
```

What are T.S. good for?

So far, we've been verifying sequential programs

What about multithreaded or networked programs?

```
Thread 1                 Thread 2
x = 0
                         x = 1
assert x == 0
                         assert x == 1
```

What are T.S. good for?

So far, we've been verifying sequential programs

What about multithreaded or networked programs?

```
Node 1                    Node 2
x = 0
send “x is 0”
…
x = 1
```

oh I guess x is 0 now

What are T.S. good for?

So far, we've been verifying sequential programs

- What about multithreaded or networked programs?

Hoare logic can be made to work here, but it's complicated

- Active area of research!

Transition systems are an alternative formalism

- Natural model of concurrency and distribution

A Transition System is…

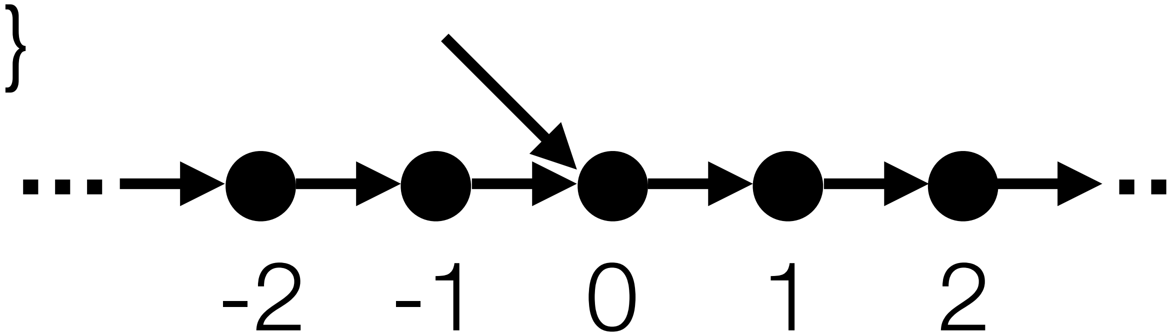
Can think of as a graph or state machine

Three things:

- State space S
- Initial states $S_0 \subseteq S$
- Transition relation $\rightarrow \subseteq S \times S$

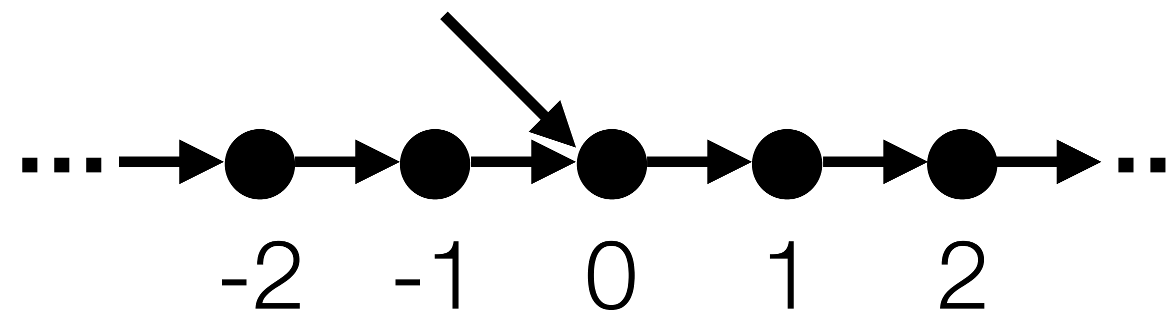
Example: Incrementing counter

- $S = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $S_0 = \{0\}$
- $\rightarrow = \{(n, n+1) \mid n \in S\}$



Executions and Invariants

An **execution** of a T.S. is:



- a sequence of states
- starting with an initial state
- where adjacent states are related by →

What are all executions in the example?

An **invariant** of a T.S. is:

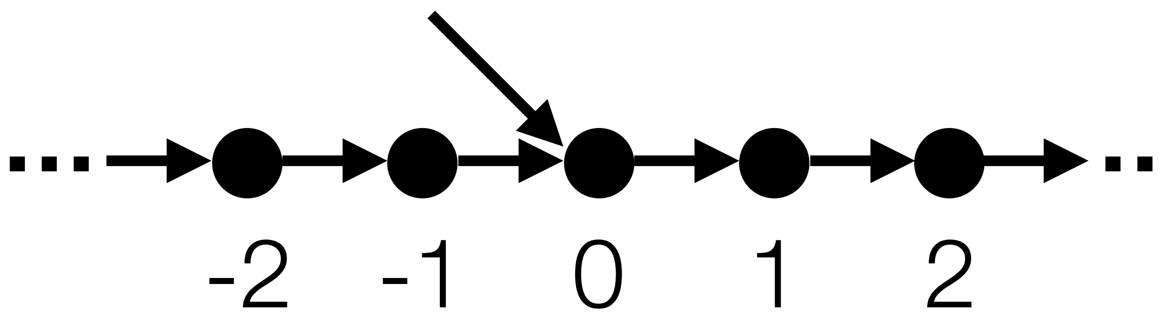
- a property of states
- that is true in every state of every execution

What are some invariants of the example?

Proving Invariants

To **prove** property P is invariant by induction, show:

- P is true in all initial states
- if P(s) and s → s' then P(s')



In the example, let $P(n) = n \geq 0$

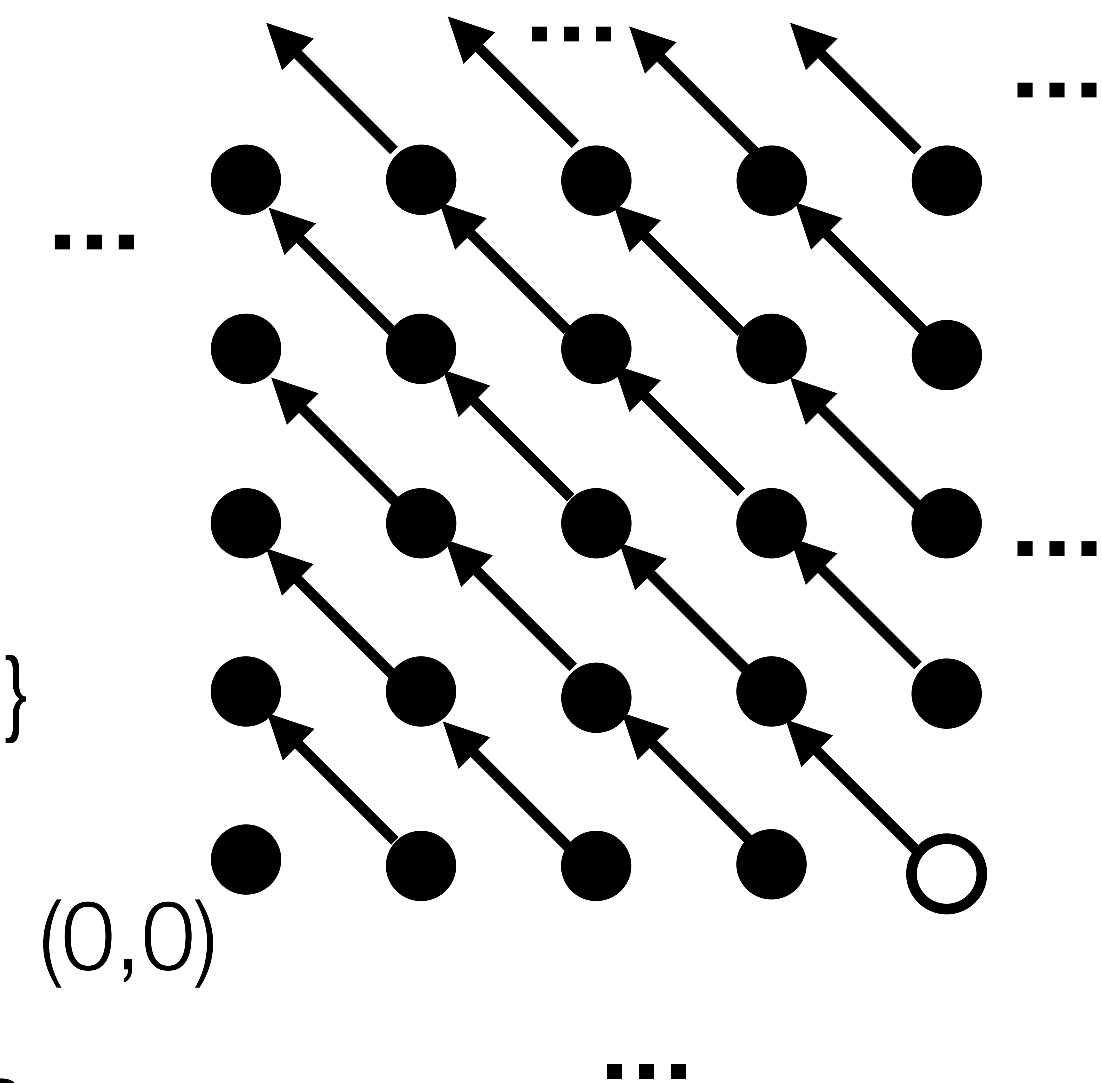
- 0 is only initial state, and P(0) is true
- suppose P(n) and n → n'
  - then $n' = n + 1$
  - so $n' \geq n \geq 0$

What if we change S to {0, 1, 2, …}?

When P is not enough

Example: Given fixed n.

- $S = \{(x,y) \mid x, y \in \mathbb{Z}\}$
- $S_0 = \{(n,0)\}$
- $\rightarrow = \{((x,y), (x-1,y+1)) \mid x > 0\}$



Let's prove $P(x,y) = x = 0 \implies y = n$

Need to strengthen.

Dafny demo

Symbolic Transition Systems

Model system in first-order logic

vocabulary σ — “variables”

initial condition φ_{init} — 1-state formula

transition relation $\rightsquigarrow$ — 2-state formula

safety property φ_{safe} — 1-state formula

Symbolic Transition Systems

init
state

$\mathbf{s}_0$

$\varphi_{\text{init}}(\mathbf{s}_0)$

Symbolic Transition Systems

system
step

$$\mathbf{s}_0 \rightsquigarrow \mathbf{s}$$

Symbolic Transition Systems

$\mathbf{s}_0$

need to show all
reachable states ok

Symbolic Transition Systems

zero or more steps

$$\varphi_{\text{init}}(\mathbf{s}_0) \wedge \mathbf{s}_0 \rightsquigarrow^* \mathbf{s} \quad \Rightarrow \quad \varphi_{\text{safe}}(\mathbf{s})$$

need to show all reachable states ok

Symbolic Transition Systems

Can dispatch to solver

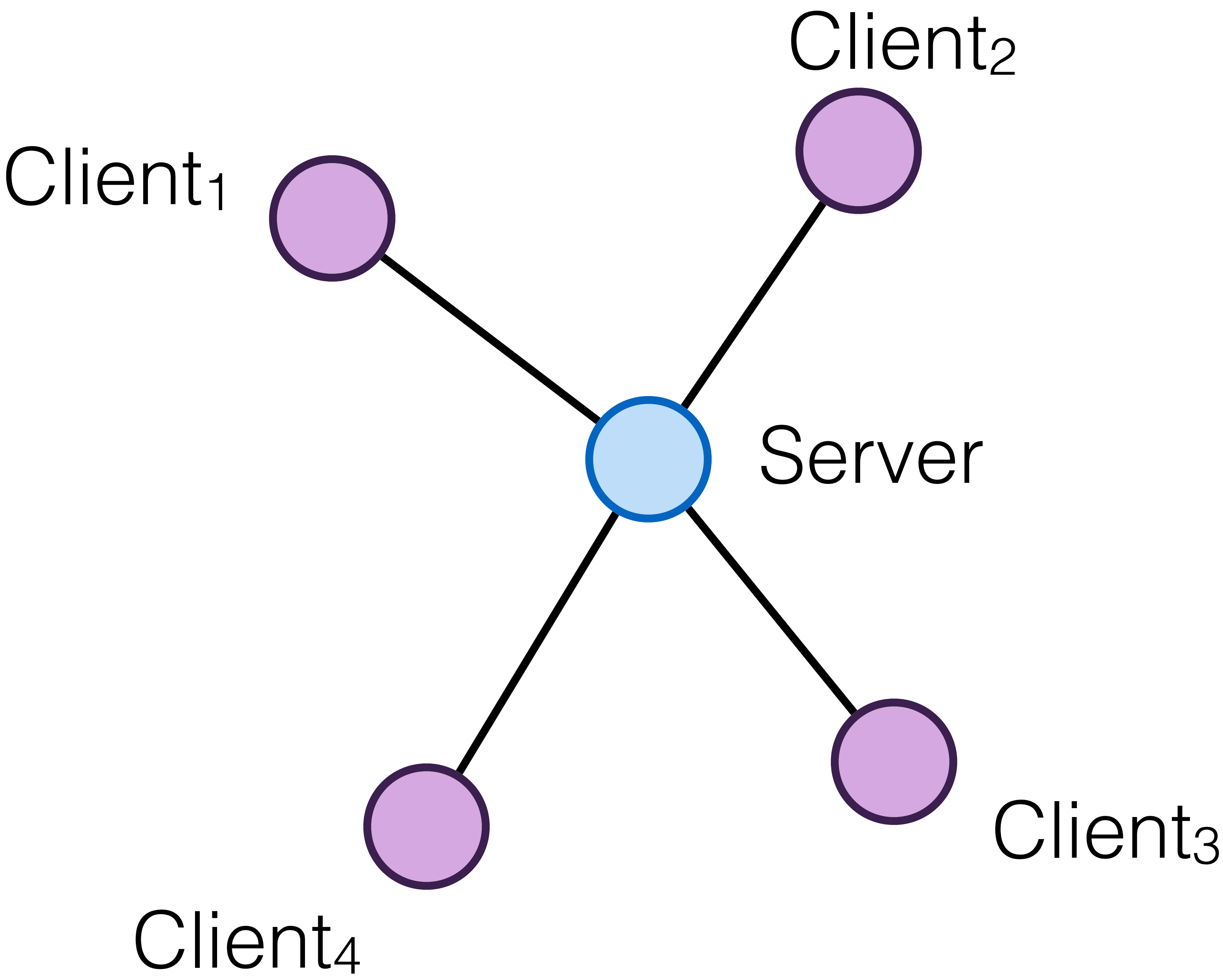
$$\frac{\varphi_{\text{init}} \Rightarrow \texttt{I} \qquad \texttt{I} \Rightarrow \varphi_{\text{safe}} \qquad \texttt{I(s)} \wedge \texttt{s} \rightsquigarrow \texttt{s'} \Rightarrow \texttt{I(s')}}{\varphi_{\text{init}}\texttt{(s}_0\texttt{)} \wedge \texttt{s}_0 \rightsquigarrow^* \texttt{s} \quad \Rightarrow \quad \varphi_{\text{safe}}\texttt{(s)}}$$

Symbolic Transition Systems

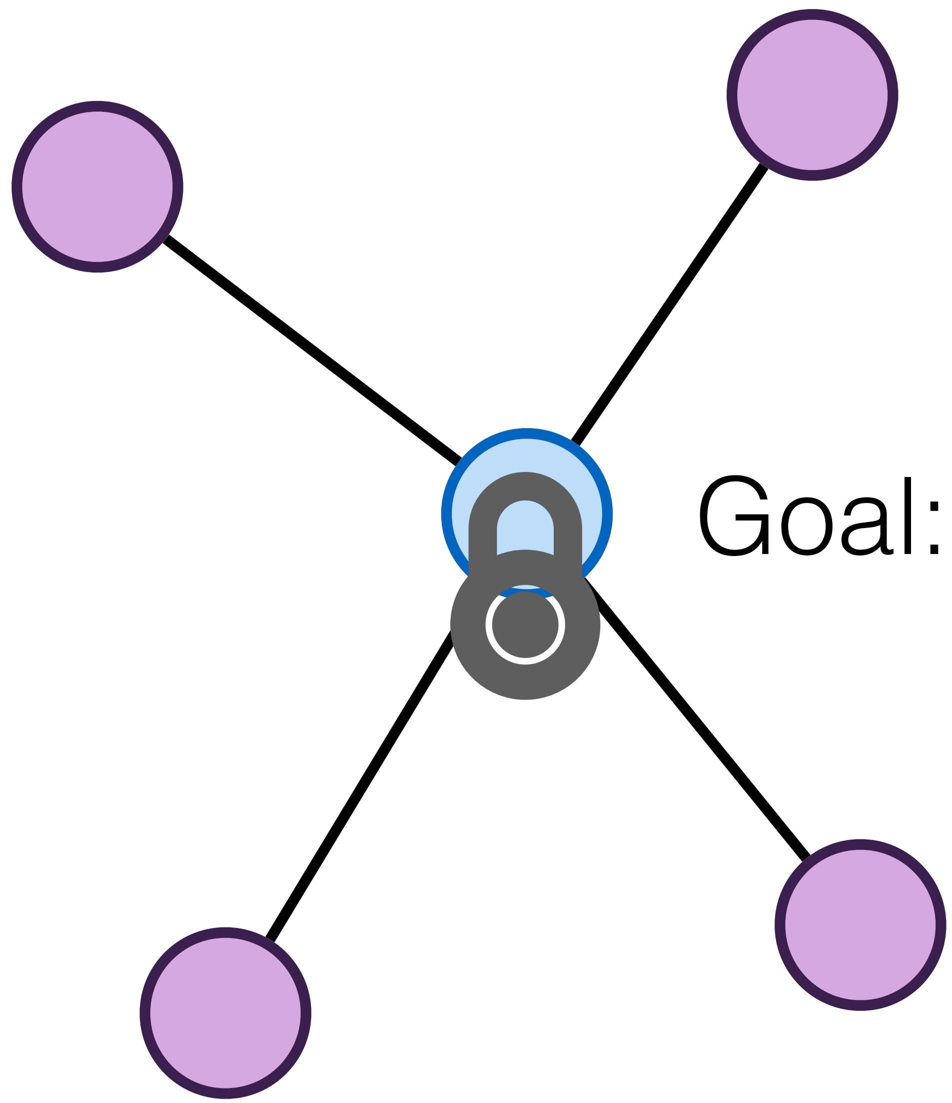
Need to find $\mathtt{I}$ (hard)

$$\frac{\varphi_{\text{init}} \Rightarrow \mathtt{I} \qquad \mathtt{I} \Rightarrow \varphi_{\text{safe}} \qquad \mathtt{I(s)} \wedge \mathtt{s} \rightsquigarrow \mathtt{s'} \Rightarrow \mathtt{I(s')}}{\varphi_{\text{init}}\mathtt{(s_0)} \wedge \mathtt{s_0} \rightsquigarrow^{*} \mathtt{s} \quad \Rightarrow \quad \varphi_{\text{safe}}\mathtt{(s)}}$$

Lock Service

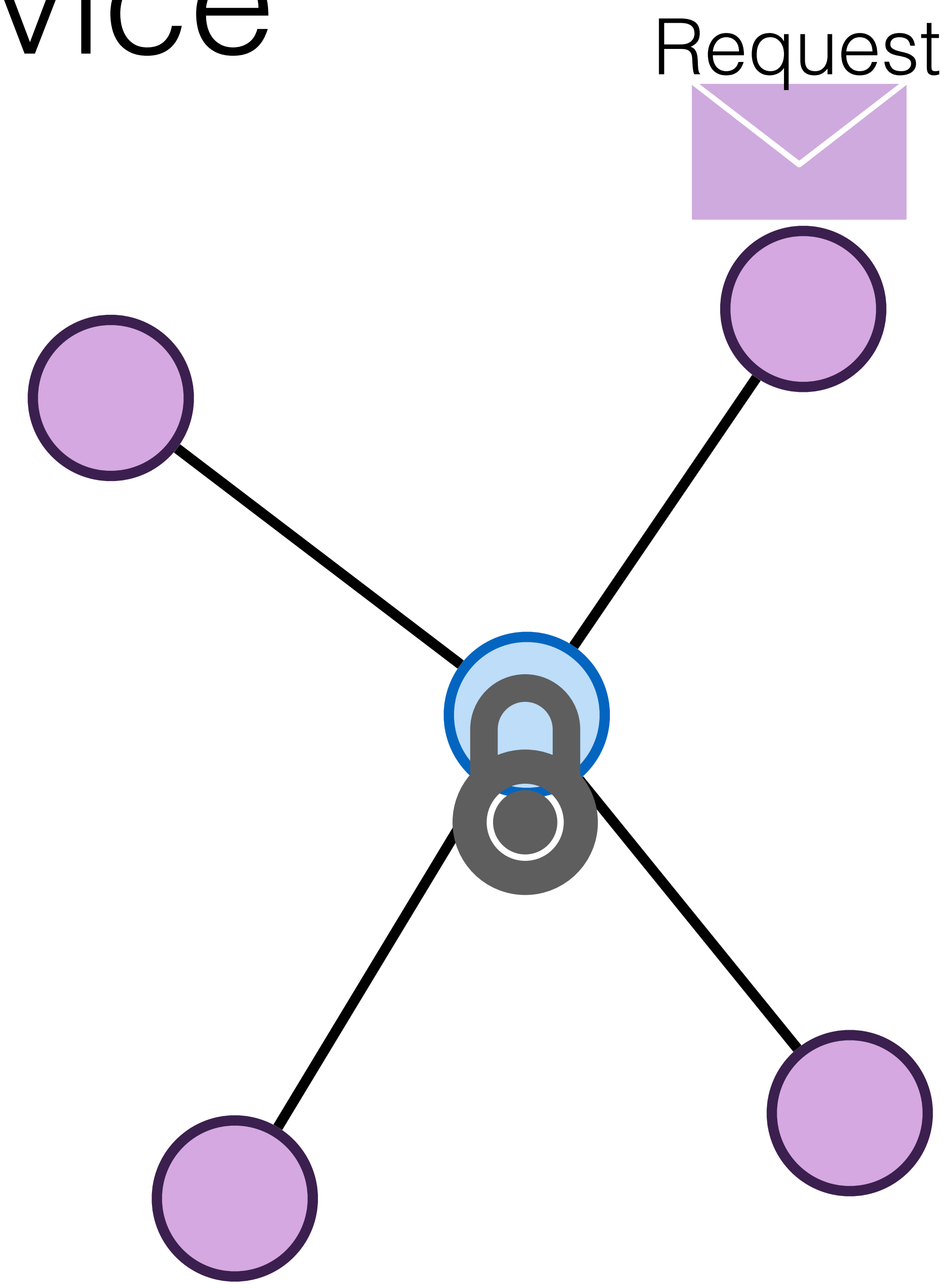


Lock Service

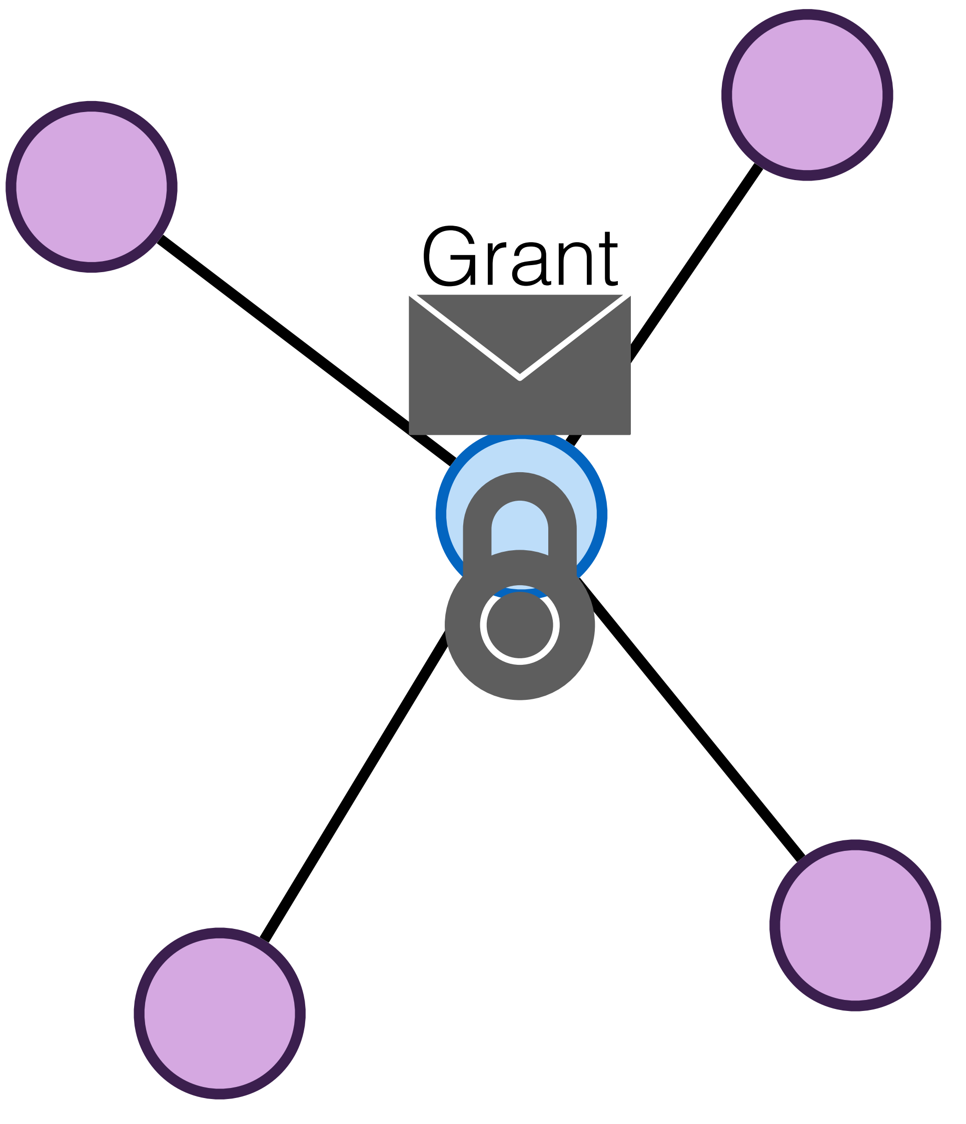


Goal: Mutual Exclusion

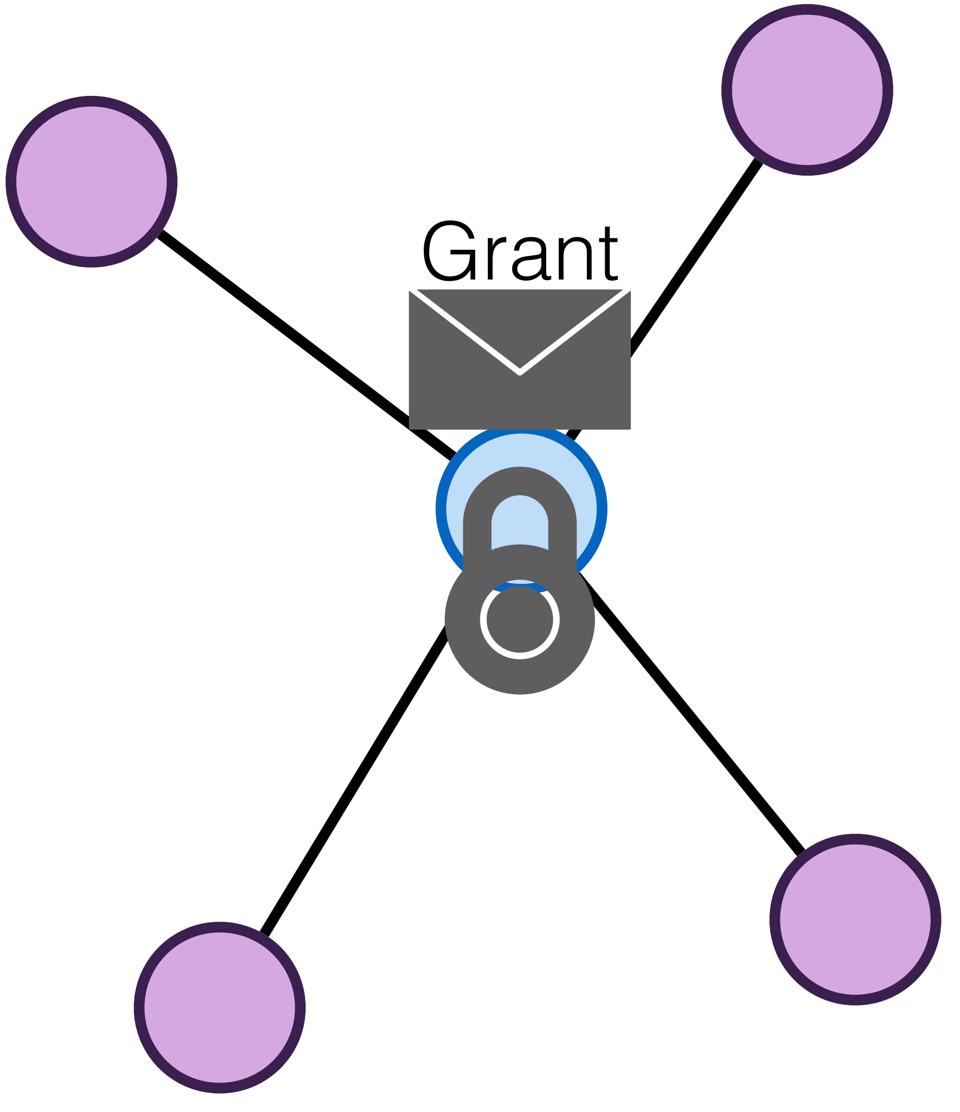
Lock Service



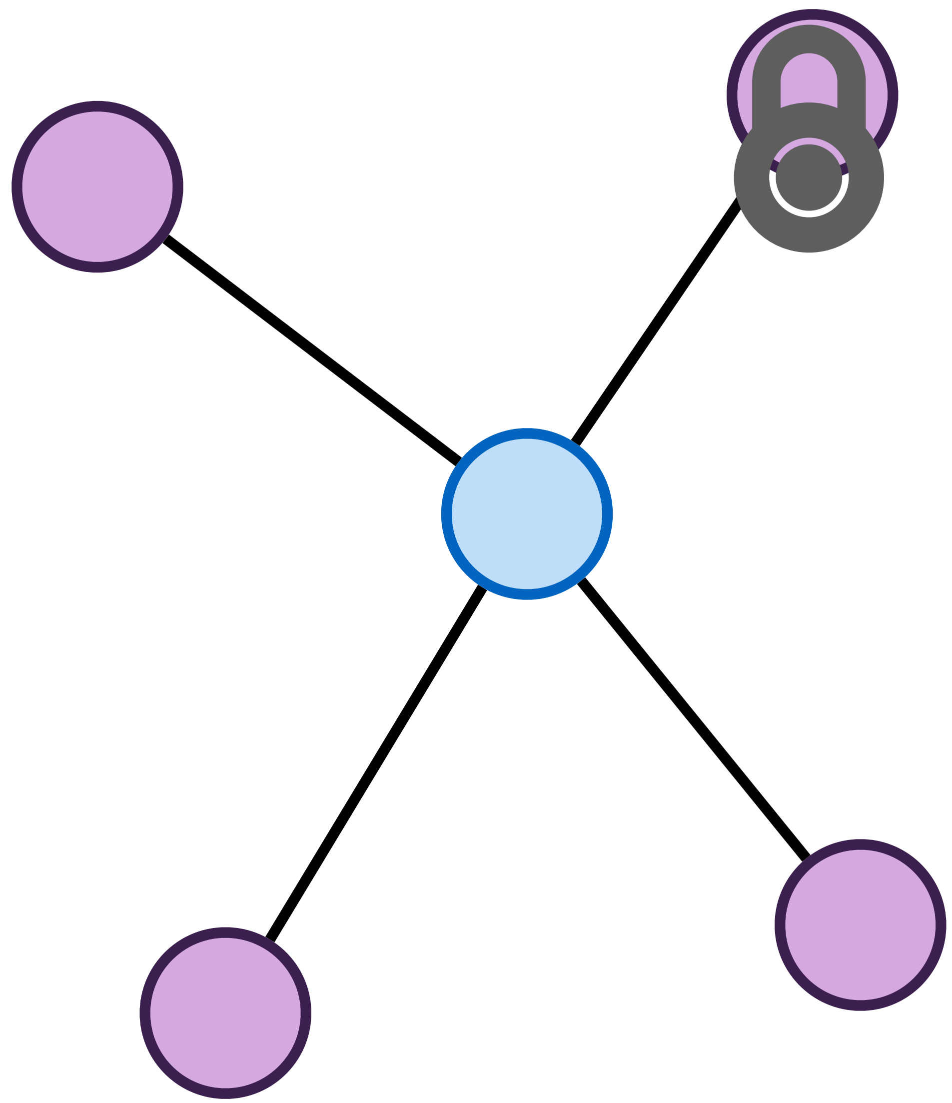
Lock Service



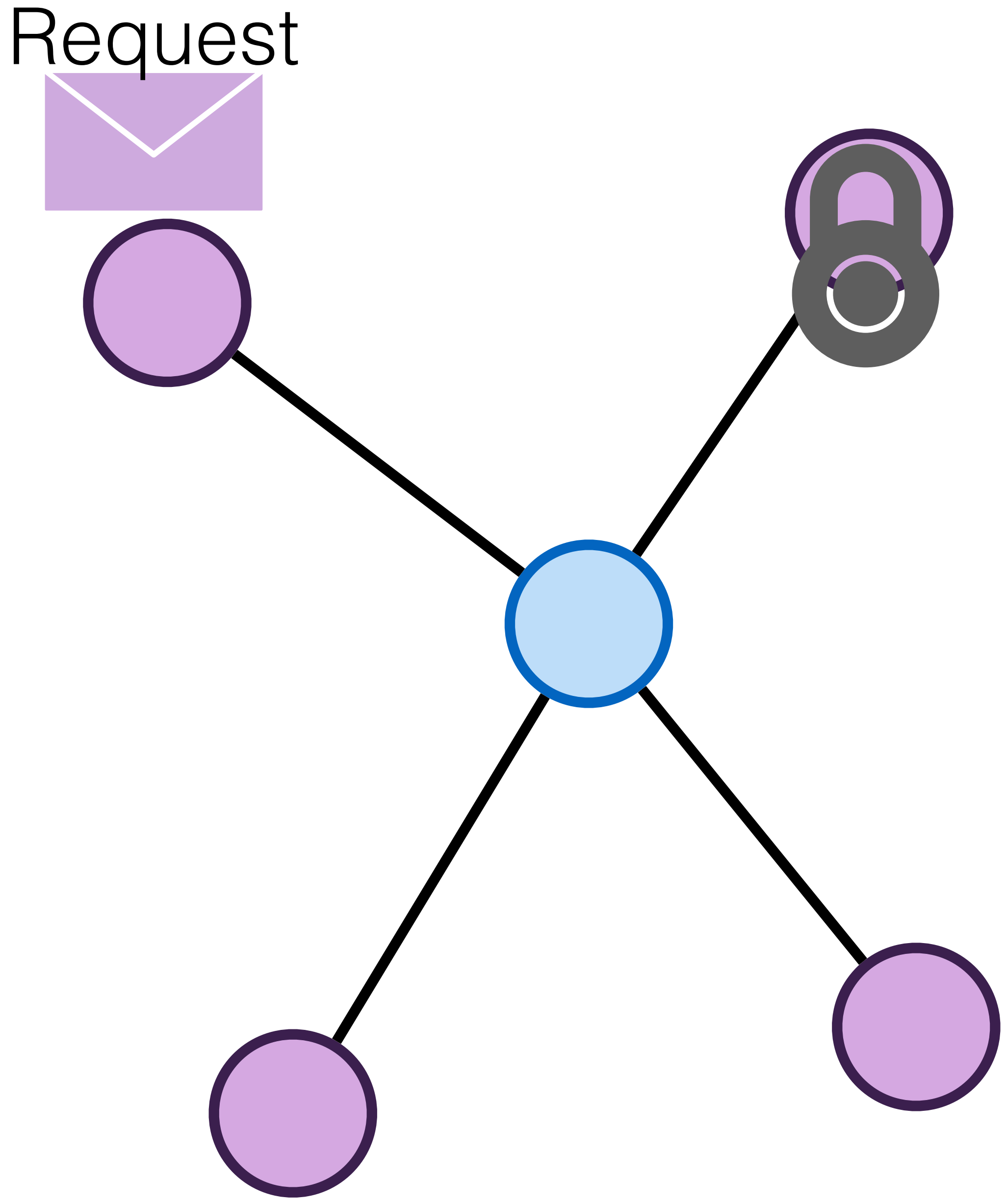
Lock Service



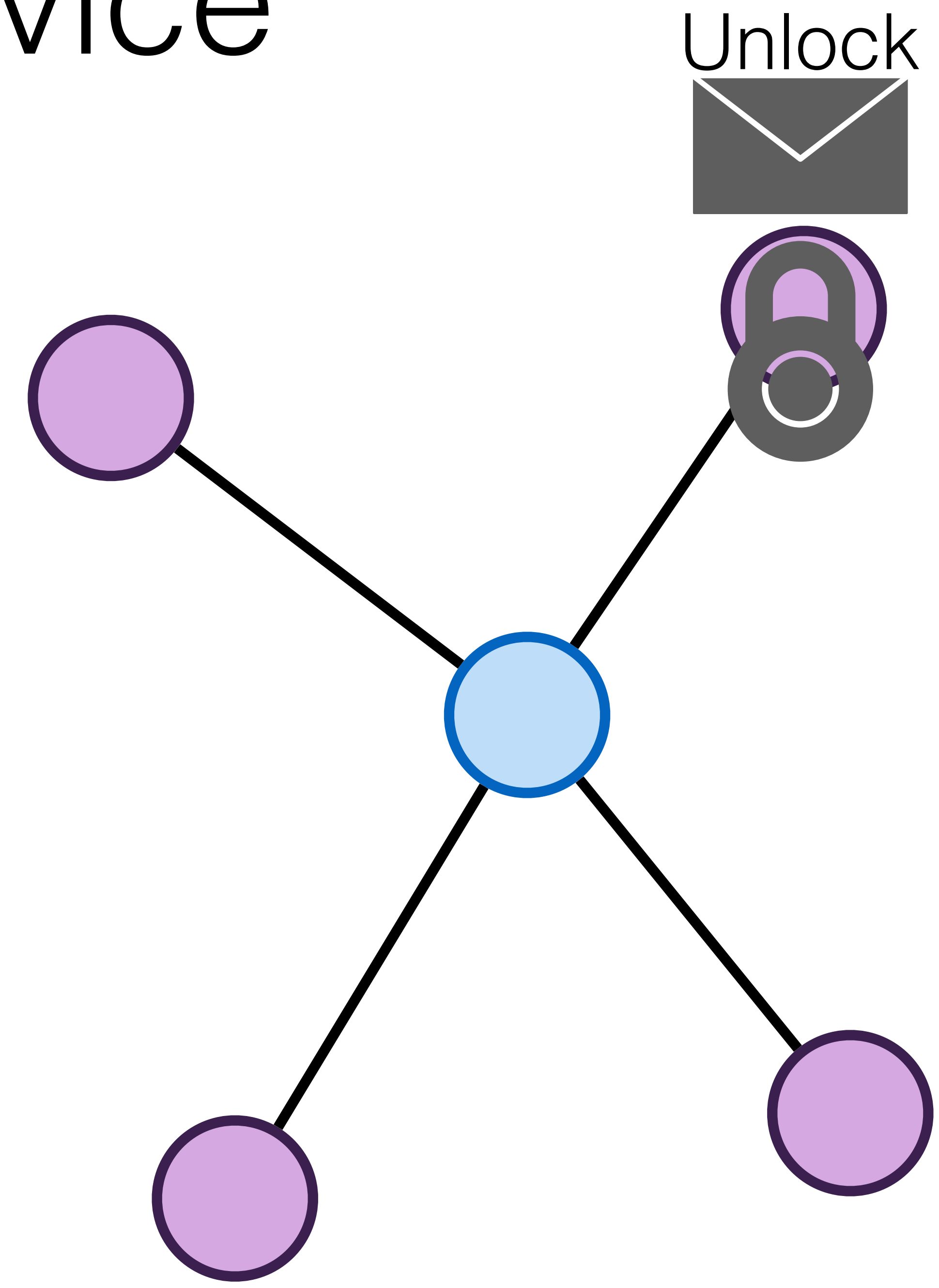
Lock Service



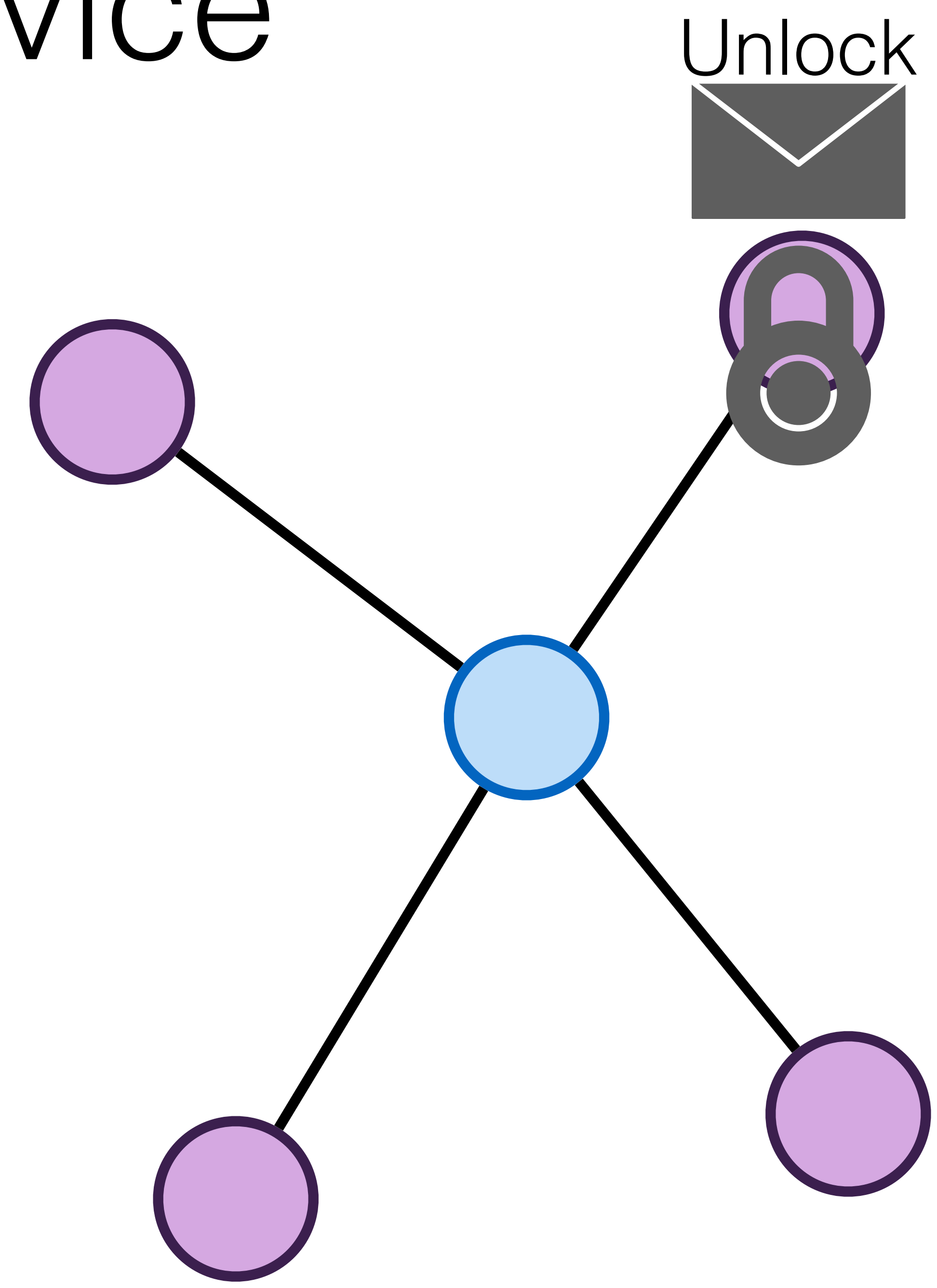
Lock Service



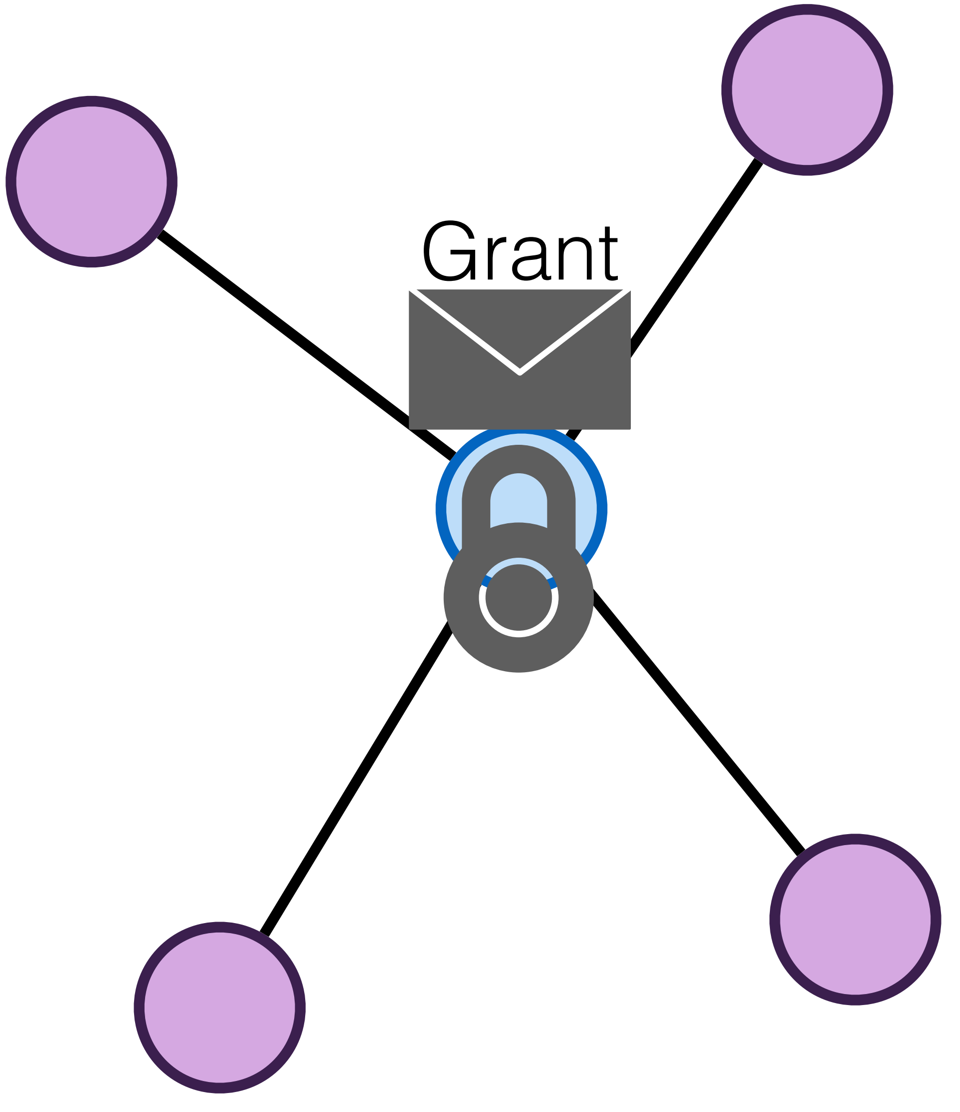
Lock Service



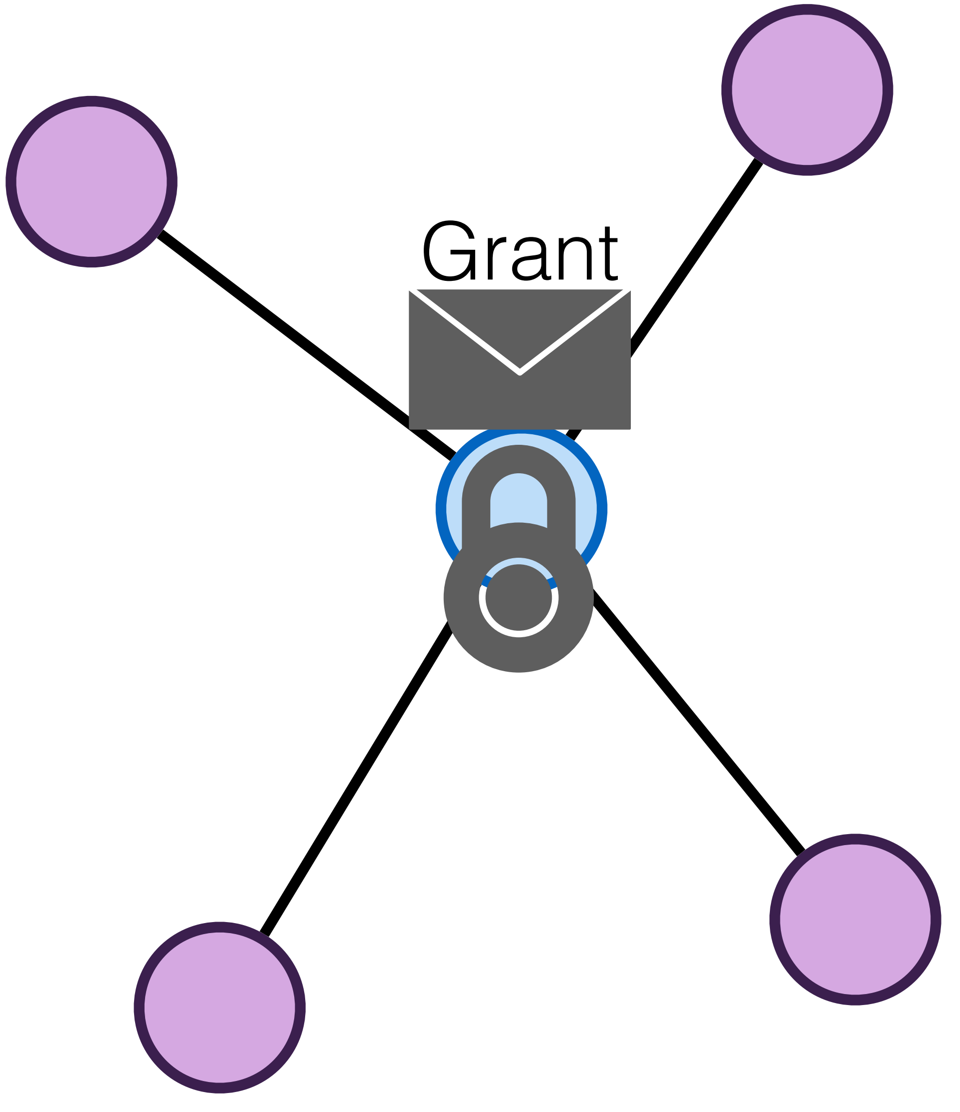
Lock Service



Lock Service



Lock Service



Lock Service