Programming with Proofs

Verifying Distributed Protocols

Distributed Protocols

Define how multiple parties (nodes)
collaborate with each other
to achieve a common goal 🤝

Operating Systems — R. Stockton Gaines, Editor

Time, Clocks, and the Ordering of Events in a Distributed System

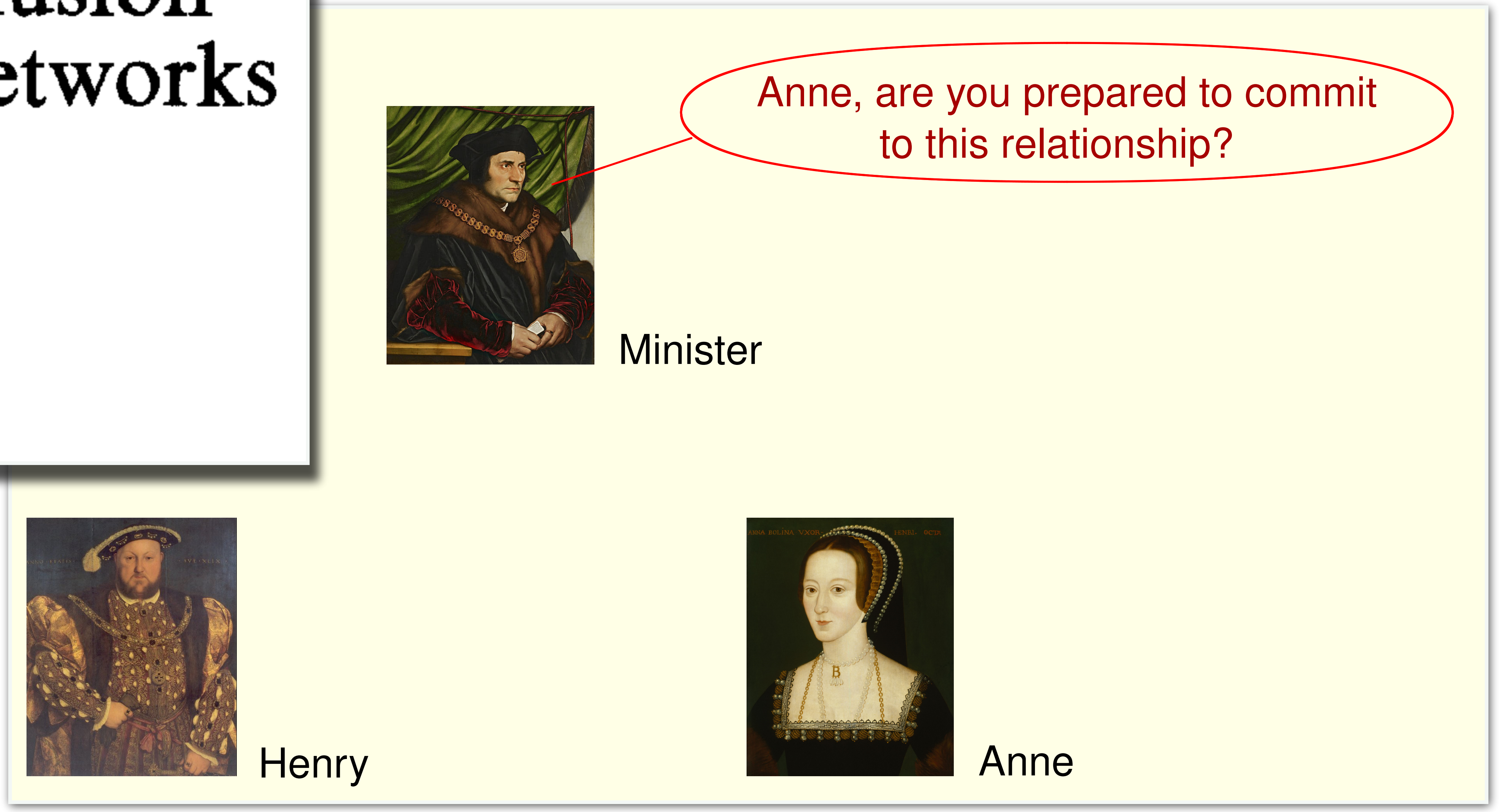
Leslie Lamport
Massachusetts Computer Associates, Inc.

Operating Systems — R. Stockton Gaines, Editor

An Optimal Algorithm for Mutual Exclusion in Computer Networks

Glenn Ricart
National Institutes of Health

Ashok K. Agrawala
University of Maryland



Two-Phase Commit depiction from Leslie Lamport's video course on TLA+

Distributed Protocols

Some nodes might *fail* 💥

Viewstamped Replication: A New Primary Copy Method to Support Highly-Available Distributed Systems

Brian M. Oki
Barbara H. Liskov

Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, MA 02139

The Part-Time Parliament

LESLIE LAMPORT
Digital Equipment Corporation

ned de-
nsistent
forget-
menting

There Is More Consensus in Egalitarian Parliaments

Iulian Moraru, David G. Andersen, Michael Kaminsky
Carnegie Mellon University and Intel Labs

In Search of an Understandable Consensus Algorithm

Diego Ongaro and John Ousterhout, *Stanford U*

https://www.usenix.org/conference/atc14/technical-sessions/pres

Just Say NO to Paxos Overhead: Replacing Consensus with Network Ordering

Jialin Li Ellis Michael Naveen Kr. Sharma Adriana Szekeres Dan R. K. Ports
University of Washington
{lijl, emichael, naveenks, aaasz, drkp}@cs.washington.edu

Distributed Protocols

Some nodes might behave *maliciously* 😈

The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Practical Byzantine Fault Tolerance

Miguel Castro and Barbara Liskov

The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus

DAVID MAZIÈRES, Stellar Development Foundation

HotStuff: BFT Consensus with Linearity and Responsiveness

Maofan Yin, Cornell University, VMware Research
Dahlia Malkhi, VMware Research
Michael K. Reiter, UNC-Chapel Hill
Guy Golan Gueta, VMware Research
Ittai Abraham, VMware Research

All You Need is DAG

Eleftherios Kokoris-Kogias, IST Austria and Novi Research
Alexander Spiegelman

Bullshark: DAG BFT Protocols Made Practical

Alexander Spiegelman
Neil Giridharan, giridhn@berkeley.edu

Jolteon and Ditto: Network-Adaptive Efficient Consensus with Asynchronous Fallback

Rati Gelashvili, *Novi Research*
Lefteris Kokoris-Kogias, *Novi Research & IST Austria*
Alberto Sonnino, *Novi Research*
Alexander Spiegelman, *Novi Research*
Zhuolun Xiang*, *University of Illinois at Urbana-Champaign*

A Secure Sharding Protocol For Open Blockchains

Loi Luu, National University of Singapore, loiluu@comp.nus.edu.sg
Viswesh Narayanan, National University of Singapore, visweshn@comp.nus.edu.sg
Chaodong Zheng, National University of Singapore, chaodong.zheng@comp.nus.edu.sg
Kunal Baweja, National University of Singapore, bawejaku@comp.nus.edu.sg
Seth Gilbert, National University of Singapore, seth.gilbert@comp.nus.edu.sg
Prateek Saxena, National University of Singapore, prateeks@comp.nus.edu.sg

Verifying Distributed Protocols

Google "Errors found in distributed protocols"
github.com/dranov/protocol-bugs-list

- Safety properties: "Bad thing will not happen."
- Liveness properties: "Good thing will eventually happen."

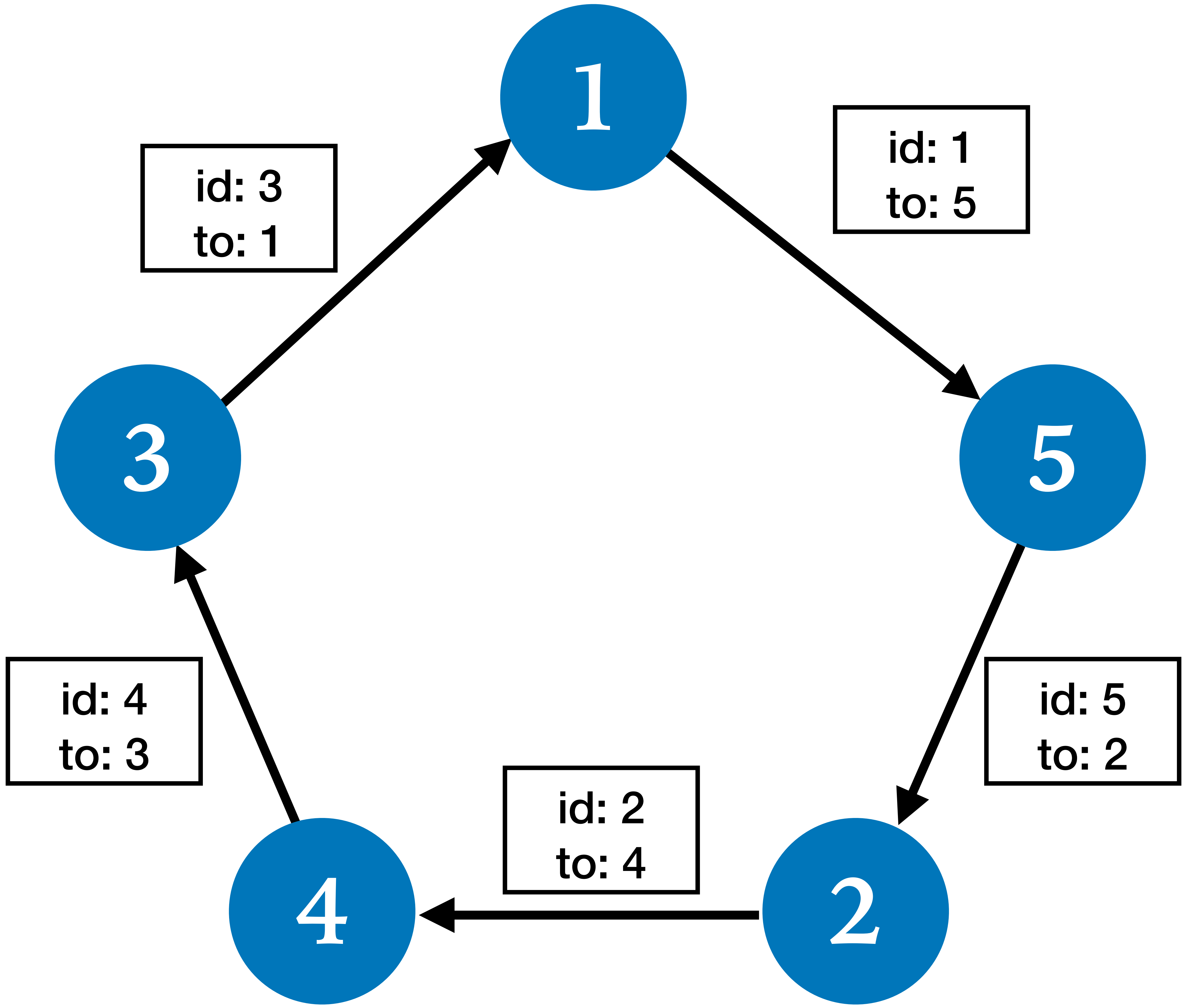
Safety bugs **can** be reliably discovered with conventional black-box **testing**.

Veil

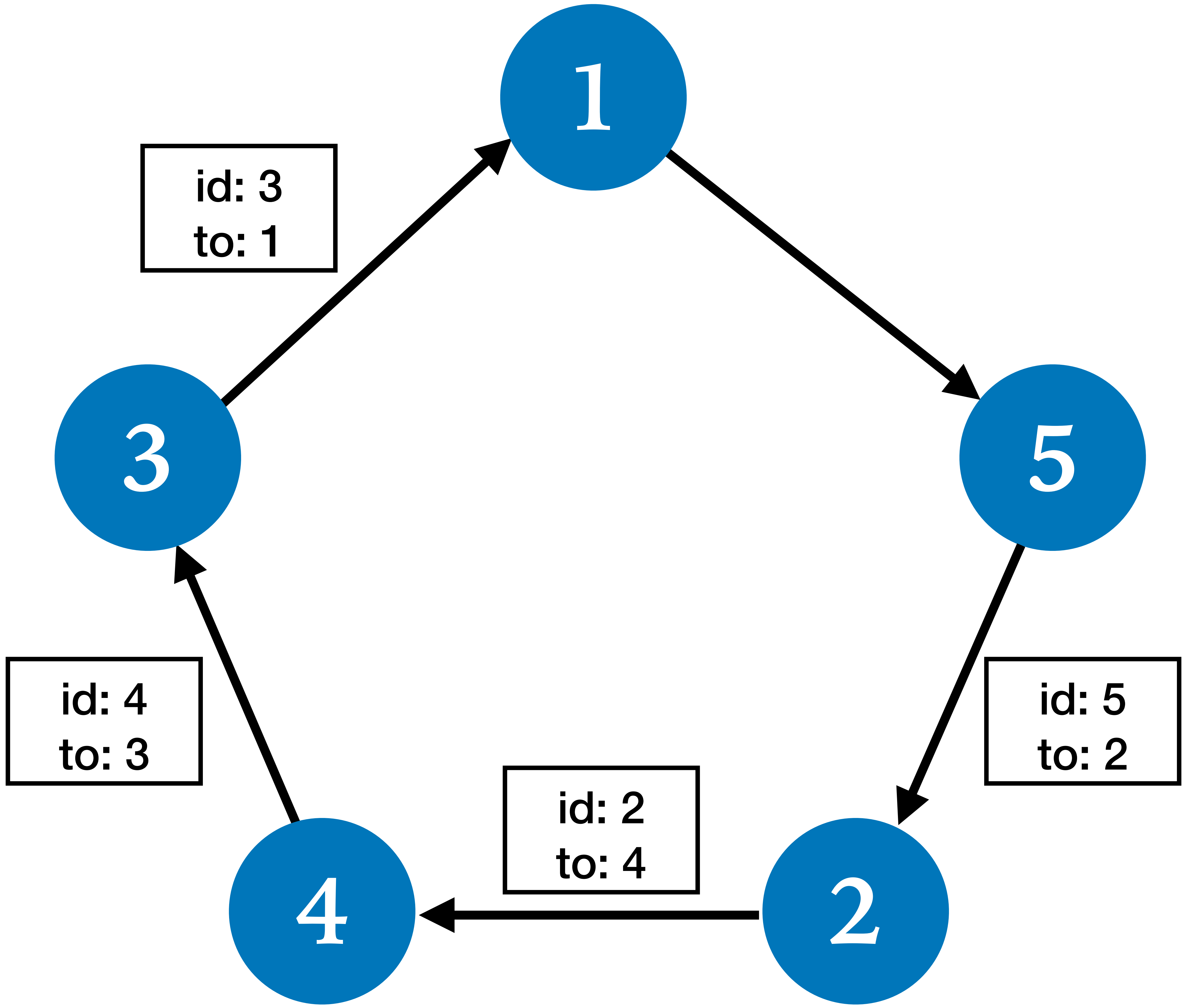
A Lean Library for Verifying Transition Systems

Arbitrary Properties	✅	A shallowly-embedded DSL in Lean
Fast Feedback	✅	Bounded model checking via SMT
Interactive Proofs for Complex (HO) Properties	✅	Out-of-the-box interactive proofs in Lean
FOL Proof Automation	✅	Uses external SMT solvers for proof goals in FOL

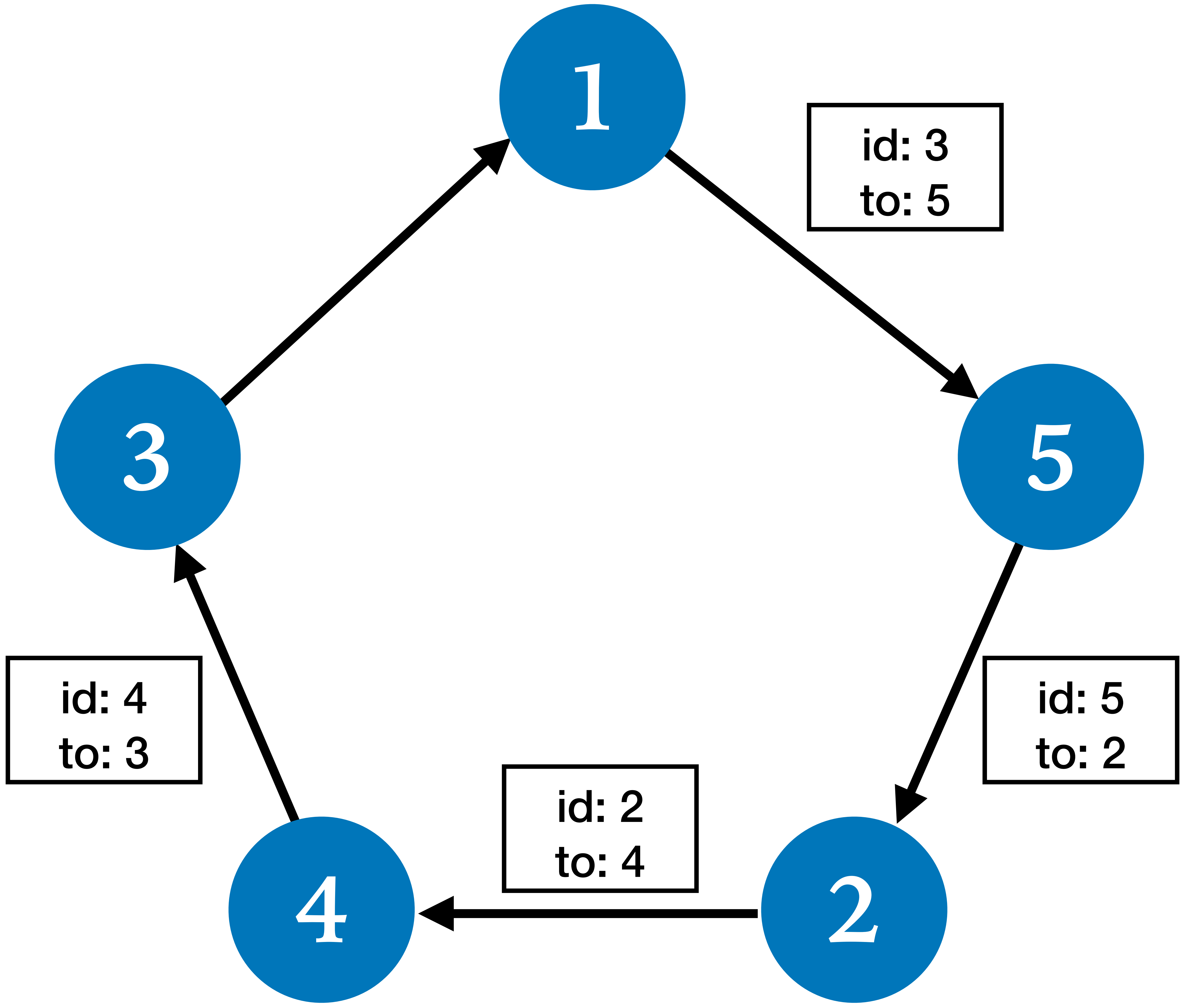
Ring Leader Election



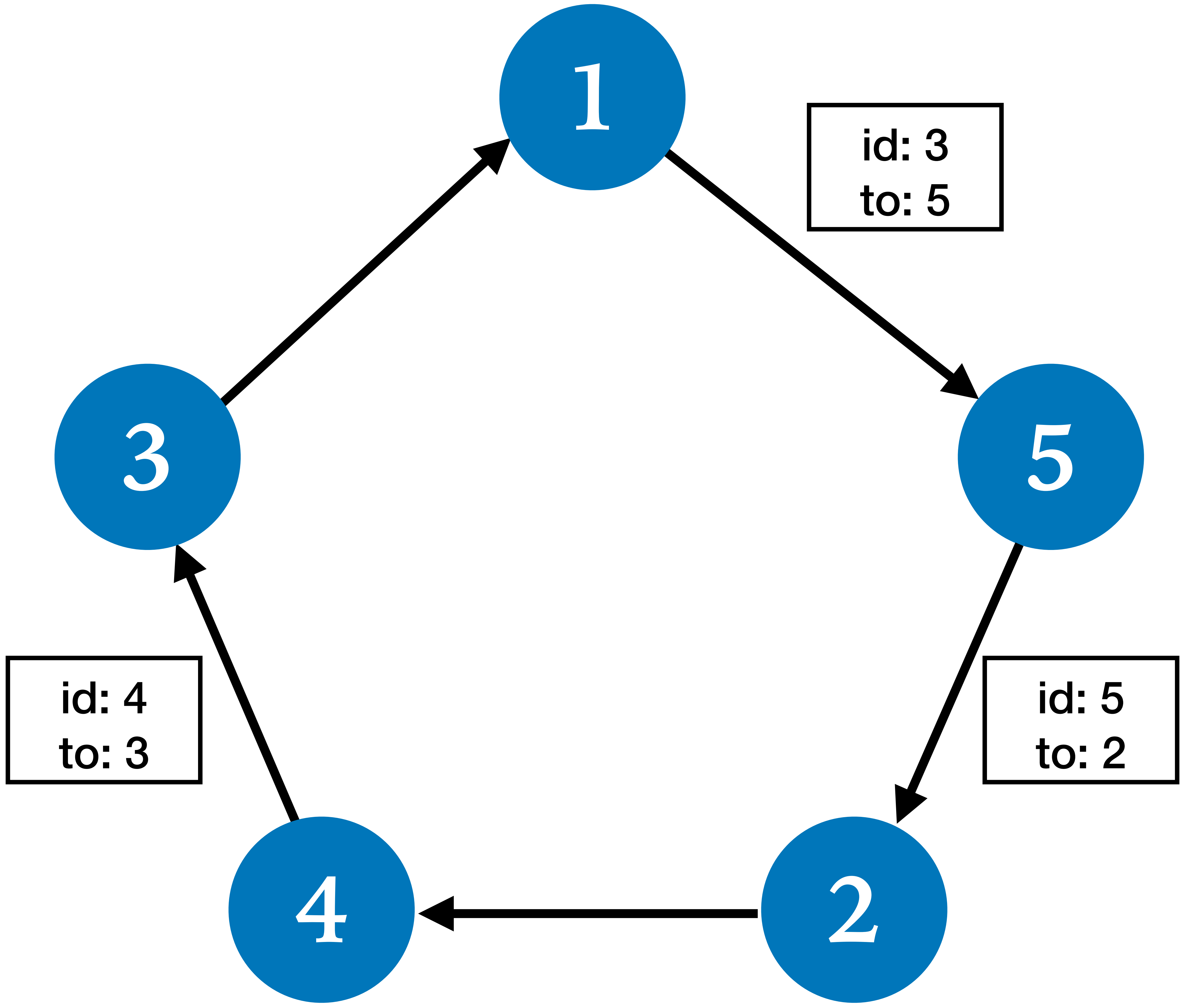
Ring Leader Election



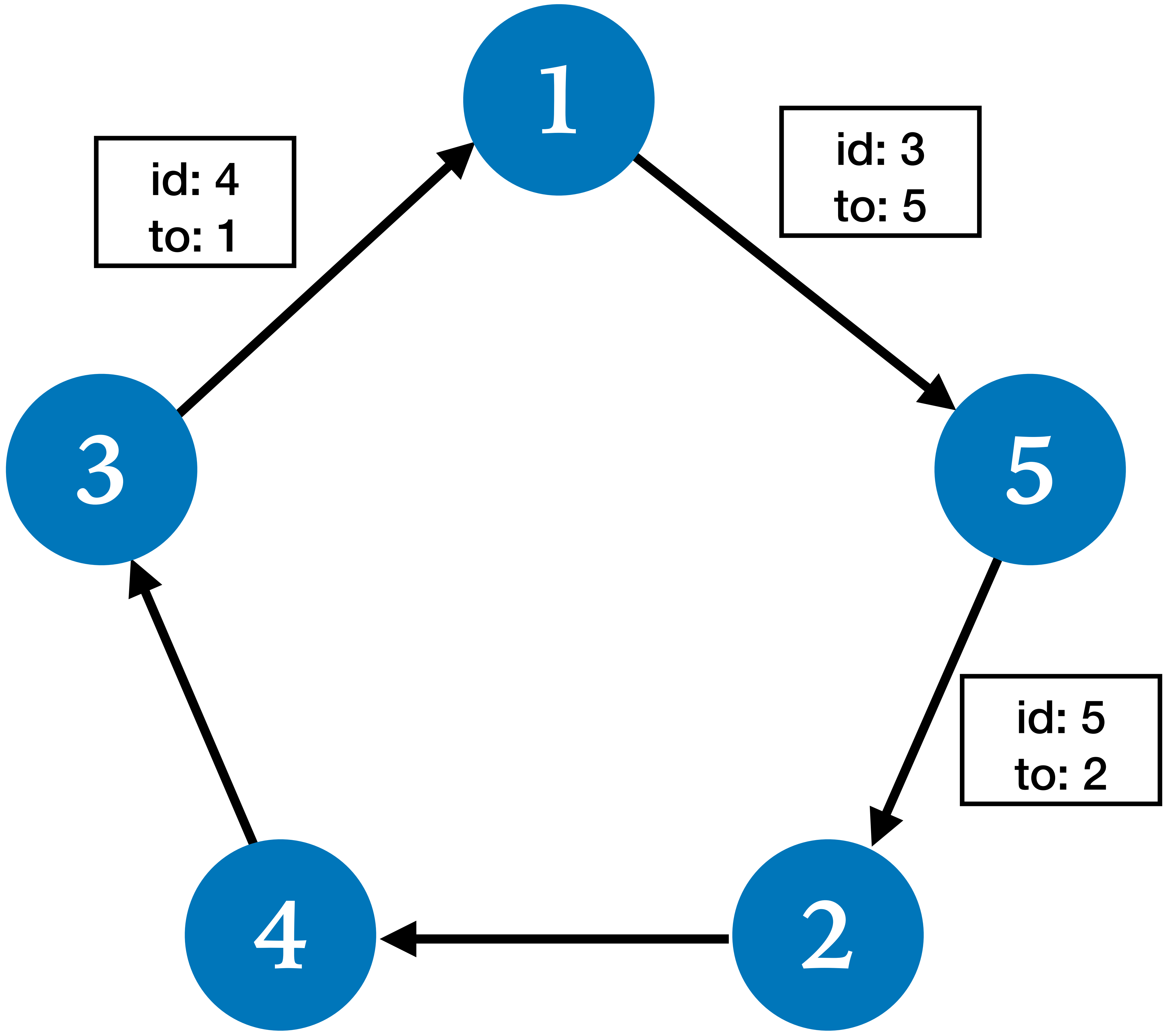
Ring Leader Election



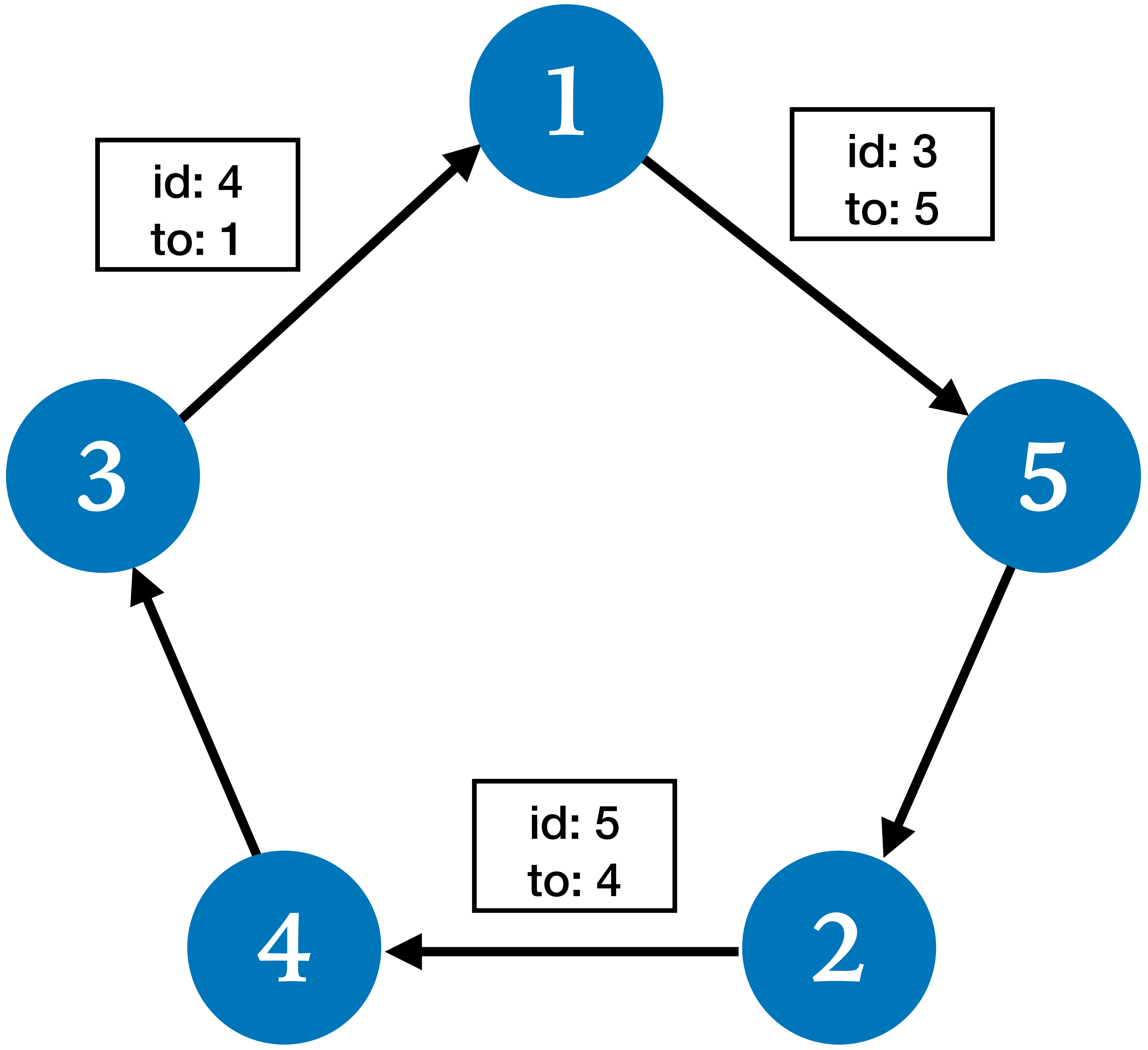
Ring Leader Election



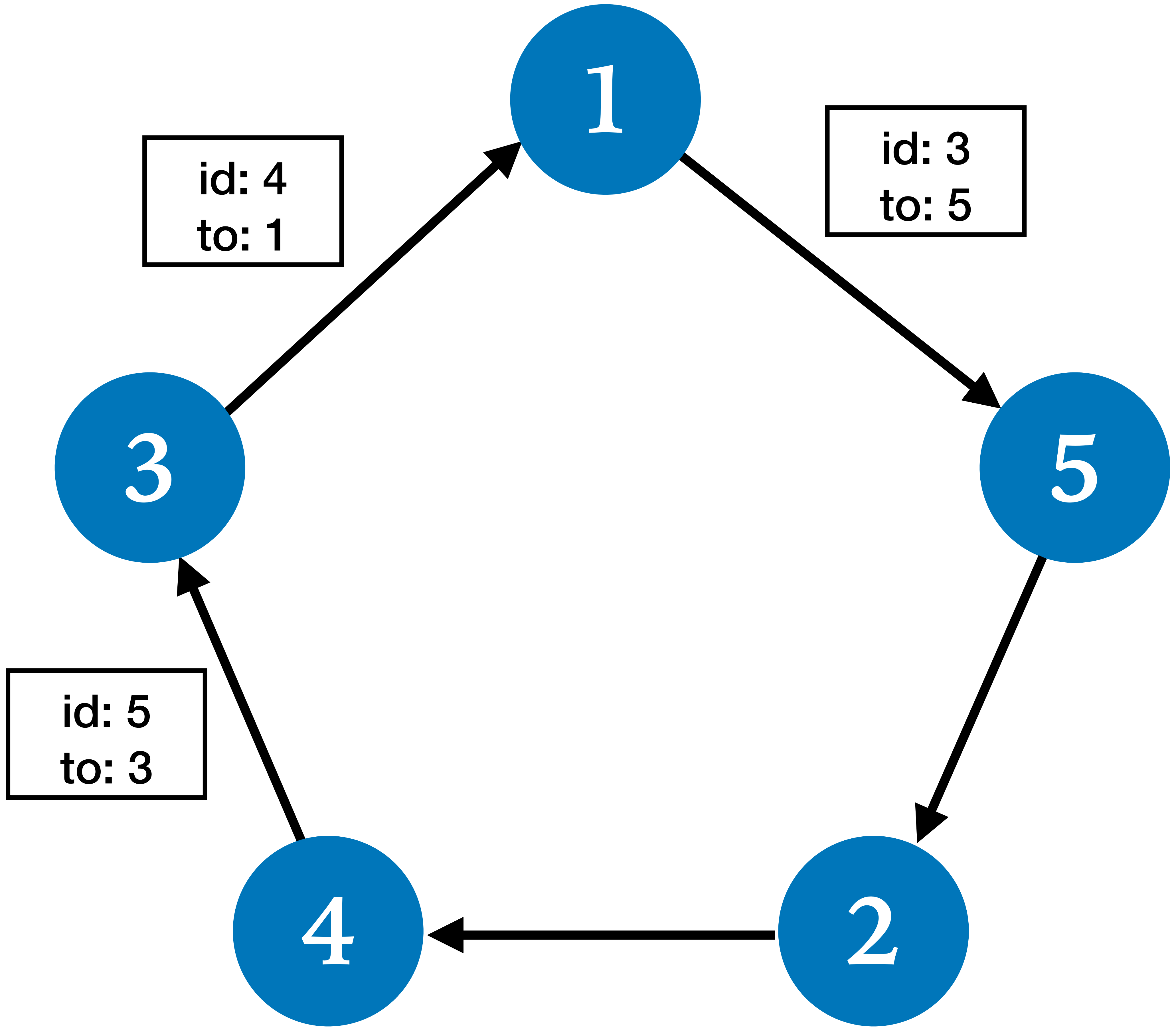
Ring Leader Election



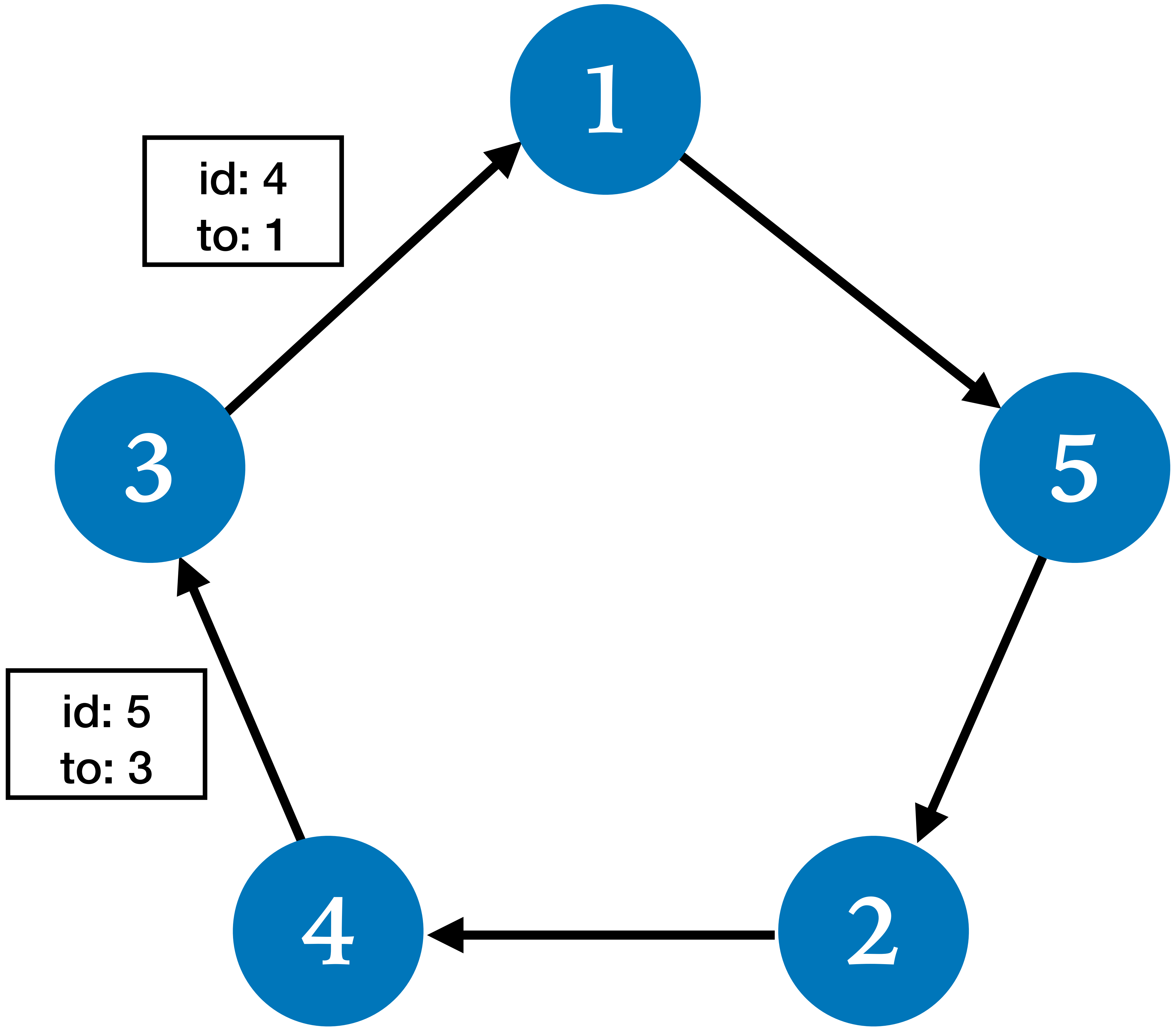
Ring Leader Election



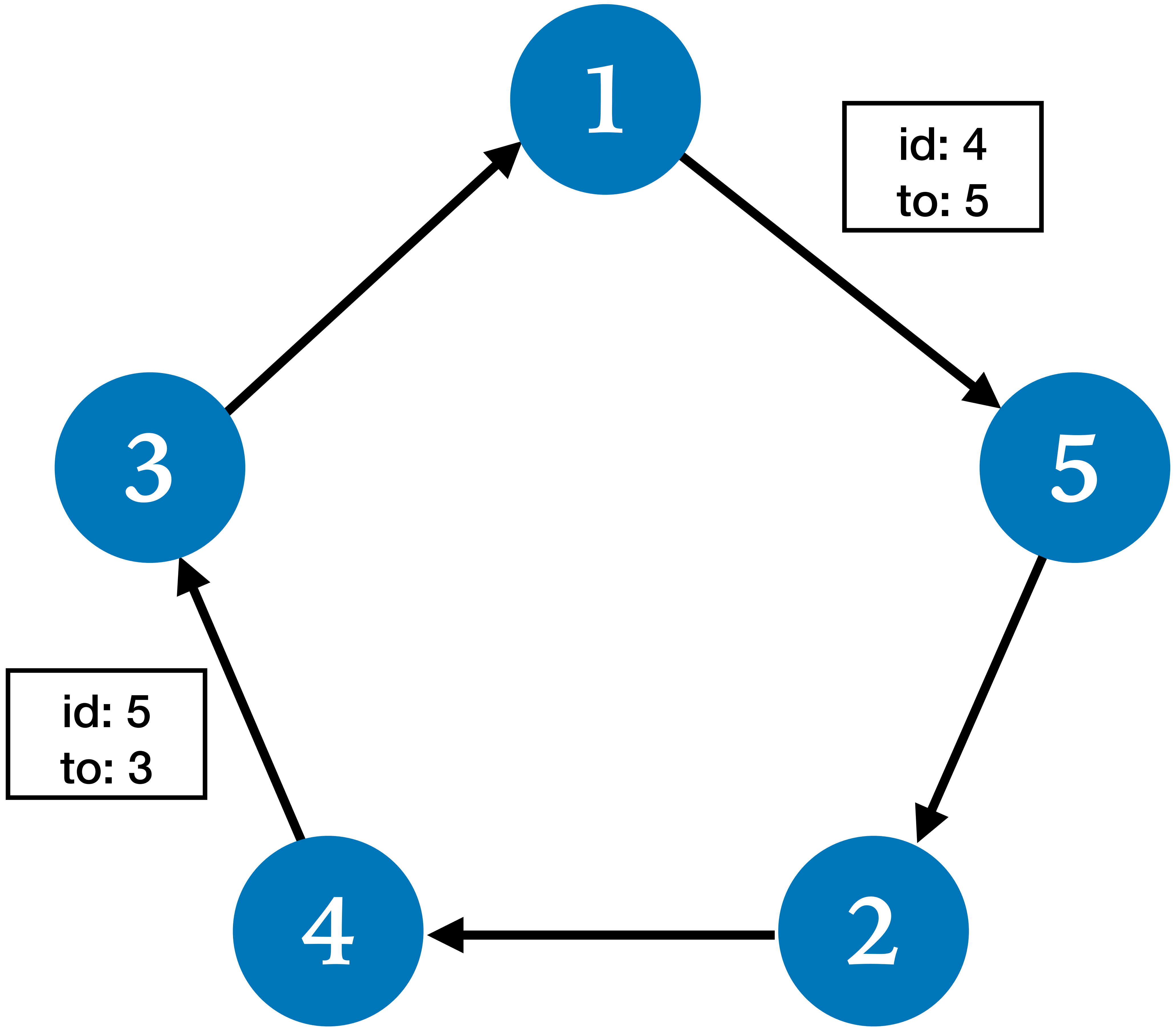
Ring Leader Election



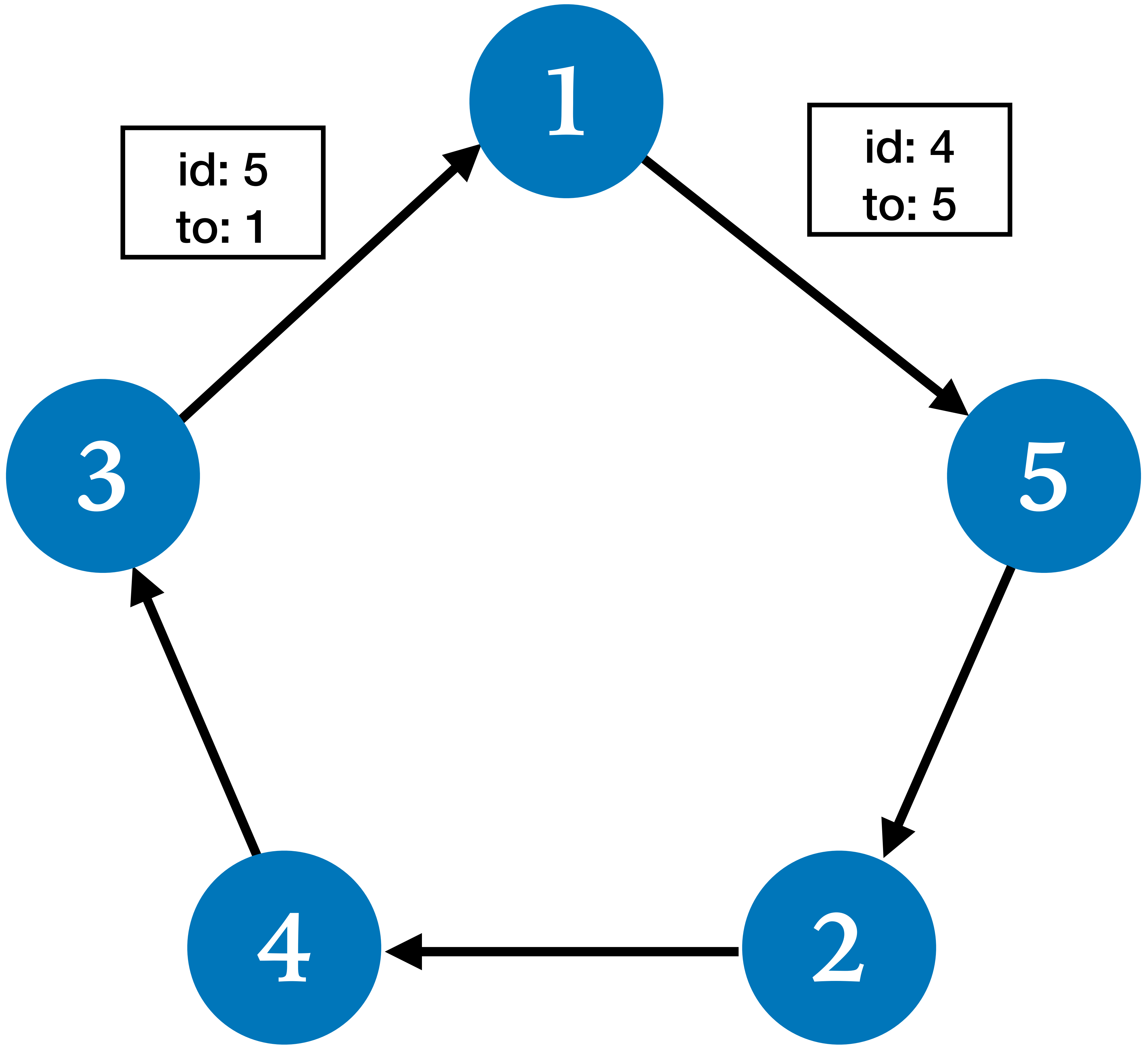
Ring Leader Election



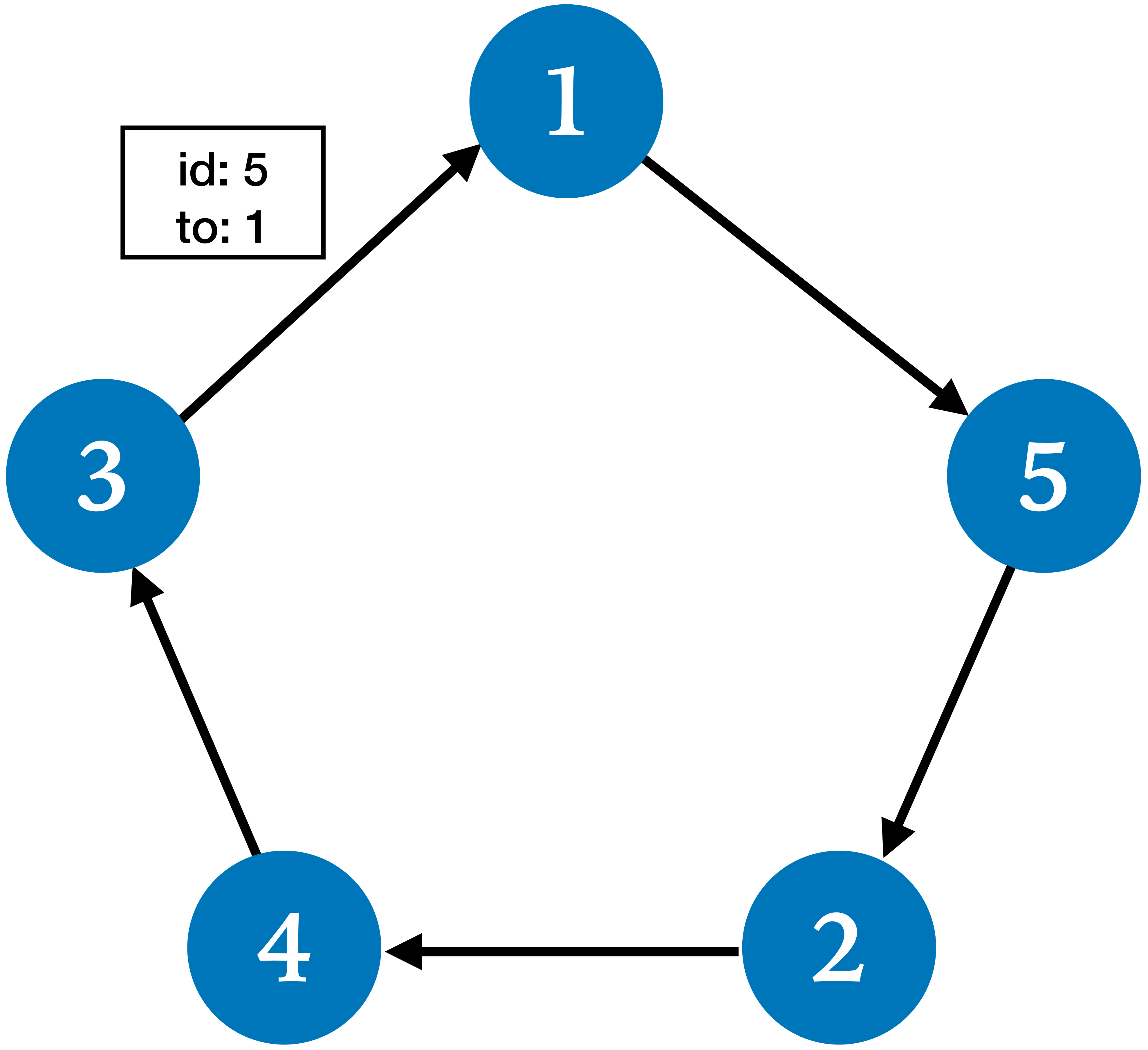
Ring Leader Election



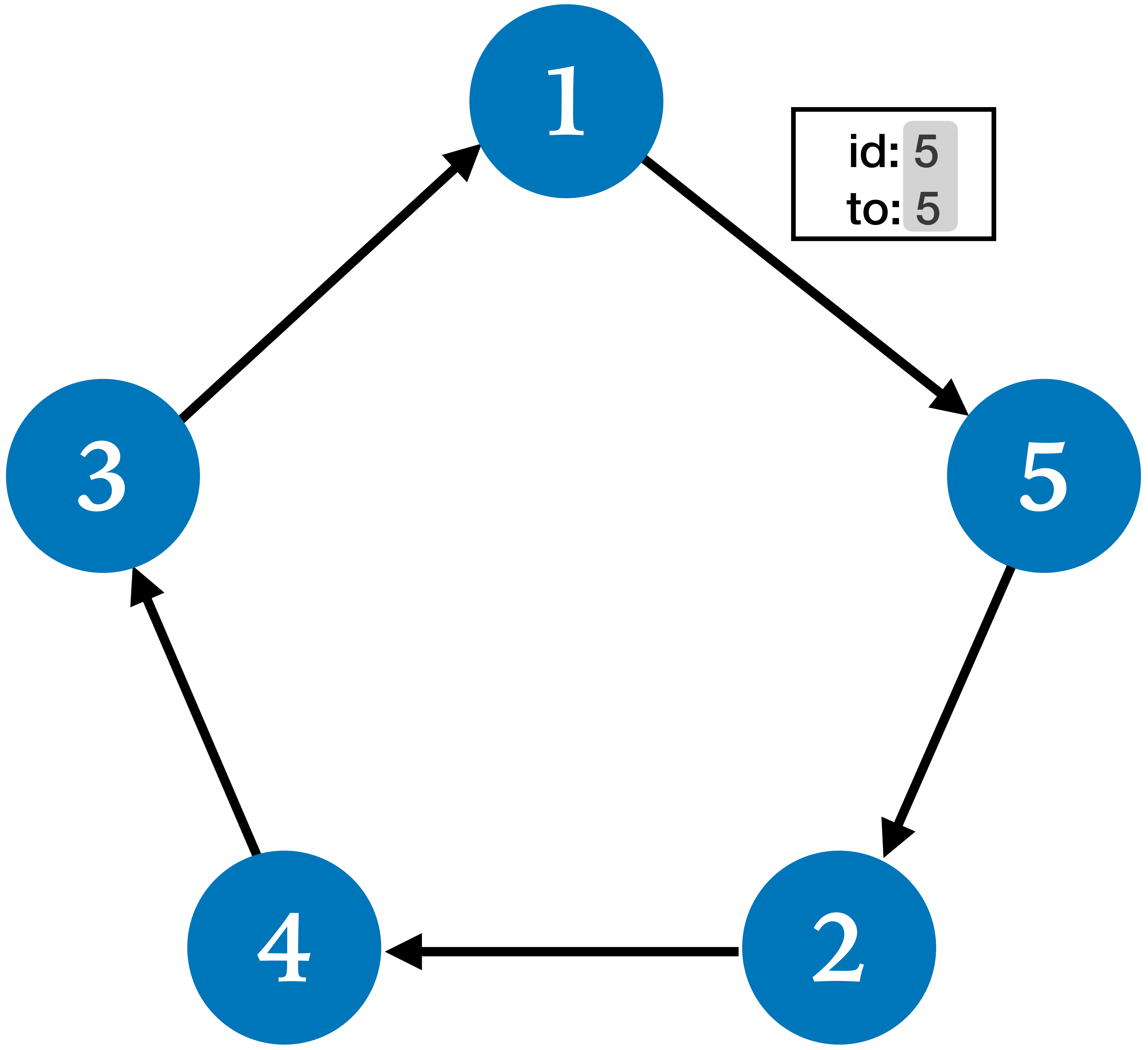
Ring Leader Election



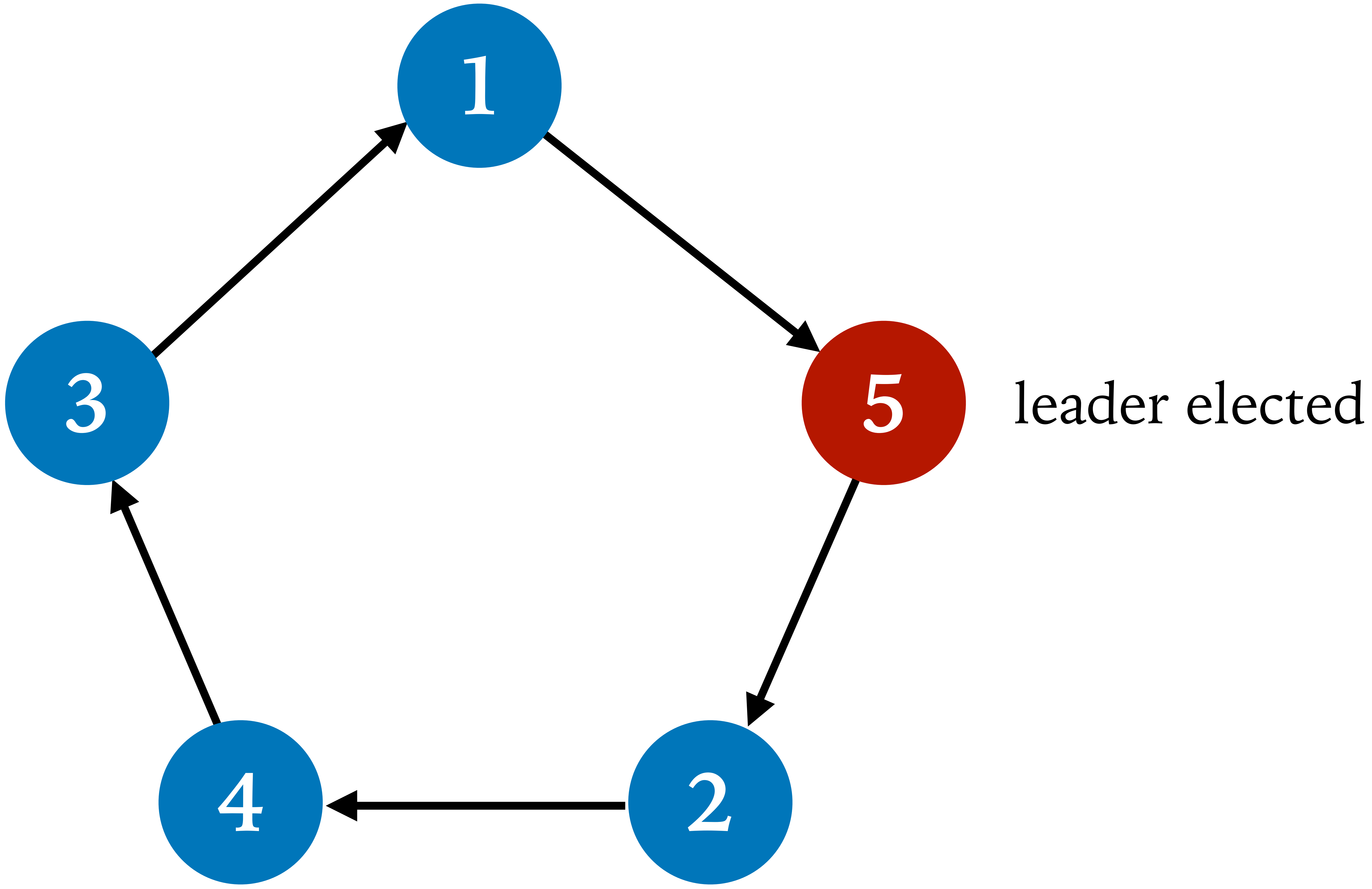
Ring Leader Election



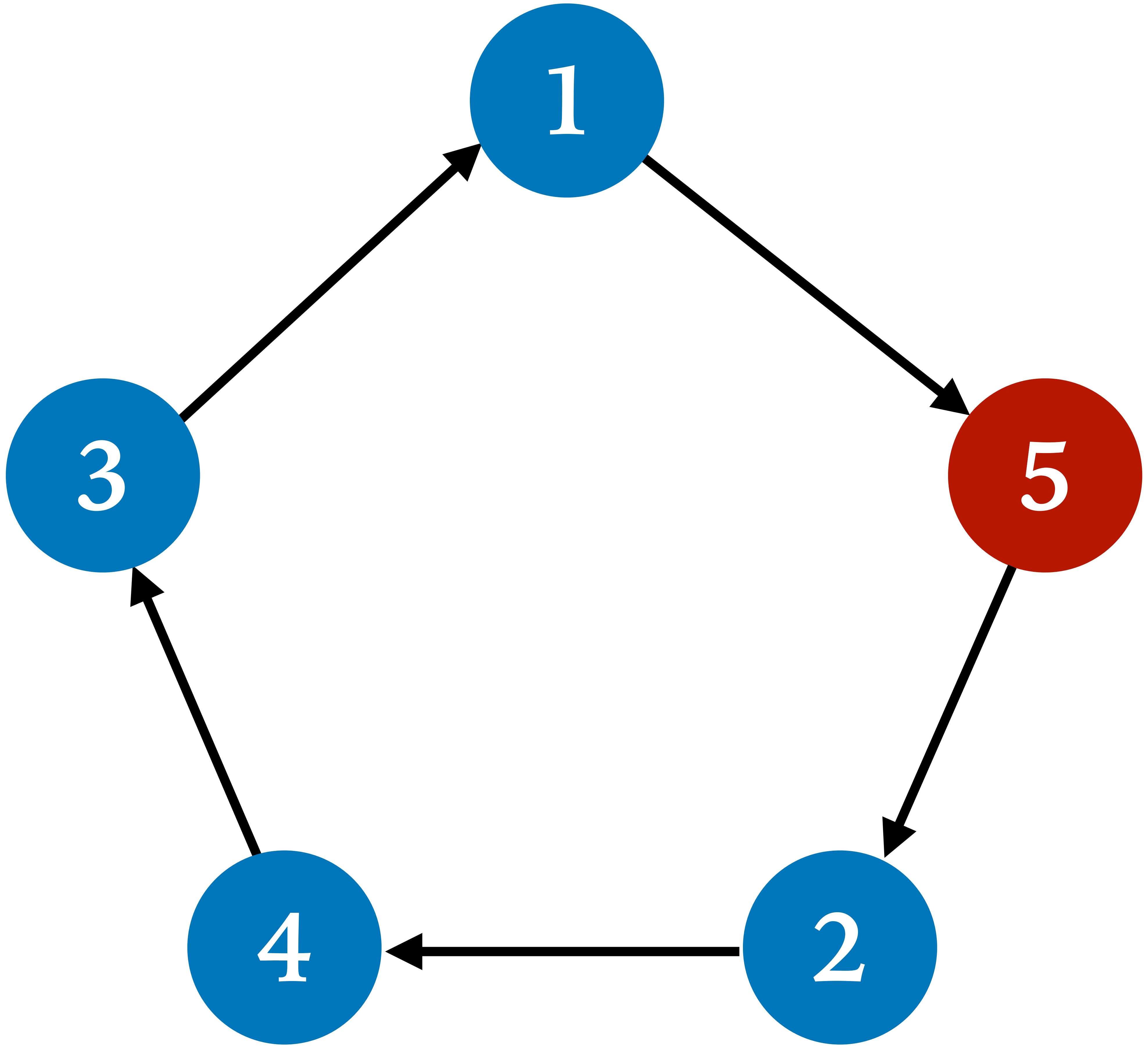
Ring Leader Election



Ring Leader Election



Safety Property to Verify



There is *at most one leader.*

Demo

System Specification

initial state

```
after_init {
  leader N := False
  pending M N := False
}
```

actions

```
action send (n next : node) = {
  require n ≠ next ∧ ∀ Z,
          ((Z ≠ n ∧ Z ≠ next) → btw n next Z)
  pending n next := True
}

action recv (id n next : node) = {
  require isNext n next
  require pending id n
  pending id n := *
  if (id = n) then
    leader n := True
  else
    if (le n id) then
      pending id next := True
}
```

safety properties

```
safety [single_leader] leader L1 ∧ leader L2 → L1 = L2

invariant [leader_greatest] leader L → le N L
invariant [self_msg_only_if_greatest] pending L L → le N L
invariant [no_bypass] pending S D ∧ btw S N D → le N S
```

Two-State Transition Semantics

initial state   result   final state

$$[\![\mathtt{act}]\!]_{\mathrm{TR}} : \underbrace{\sigma \to \rho \to \sigma \to \mathtt{Prop}}_{\text{transition relation (TR } \sigma\ \rho\text{)}}$$

Verification with Transition Semantics

```
safety [single_leader] leader L1 ∧ leader L2 → L1 = L2

invariant [leader_greatest] leader L → le N L
invariant [self_msg_only_if_greatest] pending L L → le N L
invariant [no_bypass] pending S D ∧ btw S N D → le N S
```

$[\![\text{act}]\!]_{TR} : \sigma \to \rho \to \sigma \to \text{Prop}$

$\boldsymbol{Inv} = \lambda\ (r{:}\ \rho)\ (s{:}\ \sigma).\ \text{Inv}\ s$

1. $\forall\ s_0\ s.\ \ [\![\text{after\_init}]\!]_{TR}\ s_0\ s \Rightarrow \boldsymbol{Inv}\ s$

2. $\forall\ \text{act}\ s\ r\ s'.\ \boldsymbol{Inv}\ s \wedge [\![\text{act}]\!]_{TR}\ s\ r\ s' \Rightarrow \boldsymbol{Inv}\ s'$

3. $\forall\ s.\ \text{Inv}\ s \Rightarrow \text{Safety}\ s$

Summary of the Demo

- A protocol state in Veil consists of uninterpreted *types* and *relations*
- *Actions* update the relations; they are *guarded* with `require` clauses
- Reachability of states can be *symbolically tested* using BMC
- Invariants are verified to *hold under all actions*, via SMT or interactively