Multi-Modal Verification of Transition Systems

George Pîrlea

TLA+ Community Event @ ETAPS 2026

Veil Team

George Pîrlea

Qiyuan Zhao

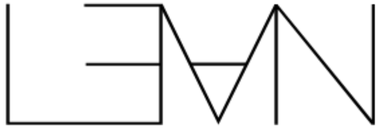
Vladimir Gladshtein

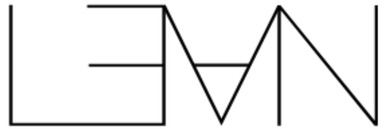
Ilya Sergey

Interns

Elad Kinsbruner (Technion)

Ziyu Mao (Hong Kong University)





Is Math the Path to Chatbots That Don't Make Stuff Up?

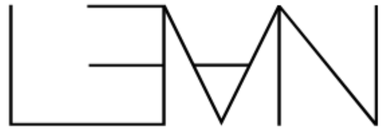
Chatbots like ChatGPT get stuff wrong. But researchers are building new A.I. systems that can verify their own math — and maybe more.

Is Math the Path to Chatbots That Don't Make Stuff Up?

Chatbots like ChatGPT get stuff wrong. But researchers are building new
maybe more

Article | Open access | Published: 12 November 2025

Olympiad-level formal mathematical reasoning with reinforcement learning



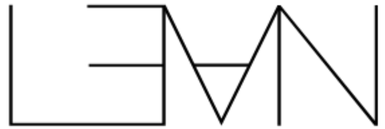
Is Math the Path to Chatbots Don't Make Stuff Up?

Chatbots like ChatGPT get stuff wrong. But researchers are building new
maybe more

Formalizing Fermat's Last Theorem in Lean: A Landmark Mathematical Project

Article | Open access | Published: 12 November 2025

Olympiad-level formal mathematical reasoning with reinforcement learning



Is Math the Path to Chatbots Don't Make Stuff Up?

Chatbots like ChatGPT get stuff wrong. But researchers are building new
maybe more

Formalizing Fermat's Last Theorem in Lean: A Landmark Mathematical Project

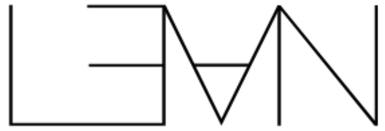
Article | Open access | Published: 12 November 2025

Olympiad-level formal mathematical reasoning with reinforcement learning

AWS Open Source Blog

Lean Into Verified Software Development

by Kesha Hietala and Emina Torlak | on 08 APR 2024 | in Amazon Verified Permissions,



Is Math the Path to Chatbots Don't Make Stuff Up?

Chatbots like ChatGPT get stuff wrong. But researchers are building new
maybe more

Formalizing Fermat's Last Theorem in Lean: A Landmark Mathematical Project

Article | Open access | Published: 12 November 2025

Olympiad-level formal mathematical reasoning with reinforcement learning

AWS Open Source Blog

Lean Into Verified Software Develop

by Kesha Hietala and Emina Torlak | on 08 APR 2024 | in Amaz

SECURITY · Nov 14, 2025

Formally Verifying Zero-Knowledge Circuits: Introducing CertiPlonk

Leonardo de Moura

The Lean Programming Language and Theorem Prover

Time: April 13, 2-3:30pm

Room: Sala 500

Lean is a general-purpose programming language and interactive theorem prover developed by the Lean Focused Research Organization (Lean FRO). It has gotten much attention as the basis for the vast mathematical library (mathlib) and successes in formalizing contemporary research mathematics, but as a general purpose system, it can also be used for authoring and verifying software among other applications. In this tutorial you will get a feeling for typical use of Lean for computer scientists: We will define a deep embedding of a simple imperative language, use Lean's flexible frontend to give it custom syntax, define semantics, and finally prove simple optimizations as well as concrete programs correct.

Veil

Veil

Multi-Modal Foundational Verifier for Distributed Protocols

Veil

Multi-Modal Foundational **Verifier for Distributed Protocols**

Veil

Multi-Modal Foundational Verifier for Distributed Protocols

Veil

Multi-Modal Foundational Verifier for Distributed Protocols

test **prove**

Veil

Multi-Modal Foundational Verifier for Distributed Protocols

concrete

test **prove**

Veil

Multi-Modal Foundational Verifier for Distributed Protocols

concrete

test **prove**

symbolic

Veil

Multi-Modal Foundational Verifier for Distributed Protocols

concrete

test

automatic

prove

symbolic

Veil

Multi-Modal Foundational Verifier for Distributed Protocols

	concrete		automatic
test		**prove**	
	symbolic		interactive

Veil

Multi-Modal **Foundational** Verifier for Distributed Protocols

Veil

Multi-Modal **Foundational** Verifier for Distributed Protocols

minimal trusted computing base

An Empirical Study on the Correctness of Formally Verified Distributed Systems

Pedro Fonseca Kaiyuan Zhang Xi Wang Arvind Krishnamurthy

University of Washington

{pfonseca, kaiyuanz, xi, arvind}@cs.washington.edu

EuroSys '17

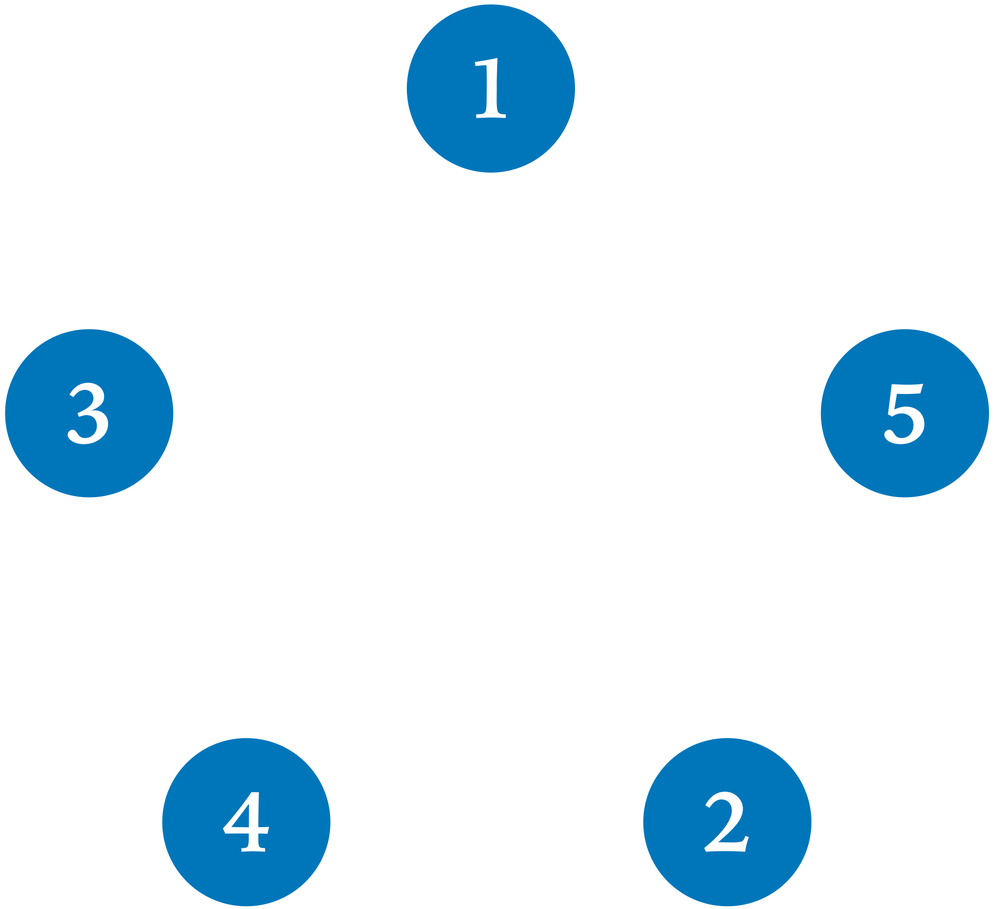
Veil

- ✅ A verifier for distributed protocols, embedded in Lean
- ✅ TLC-style explicit state model checking
- ✅ Symbolic model checking via SMT
- ✅ Out-of-the-box interactive proofs in Lean
- ✅ Small trusted computing base

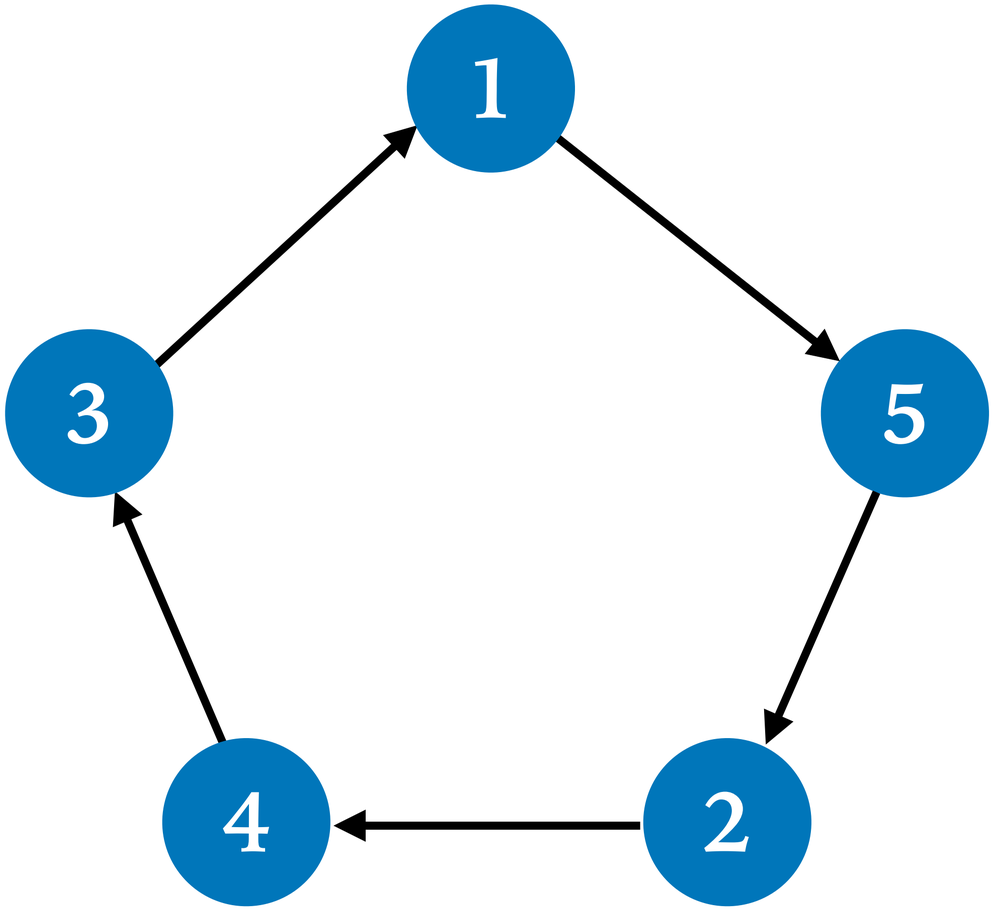
try.veil.dev

Leader Election in a Ring

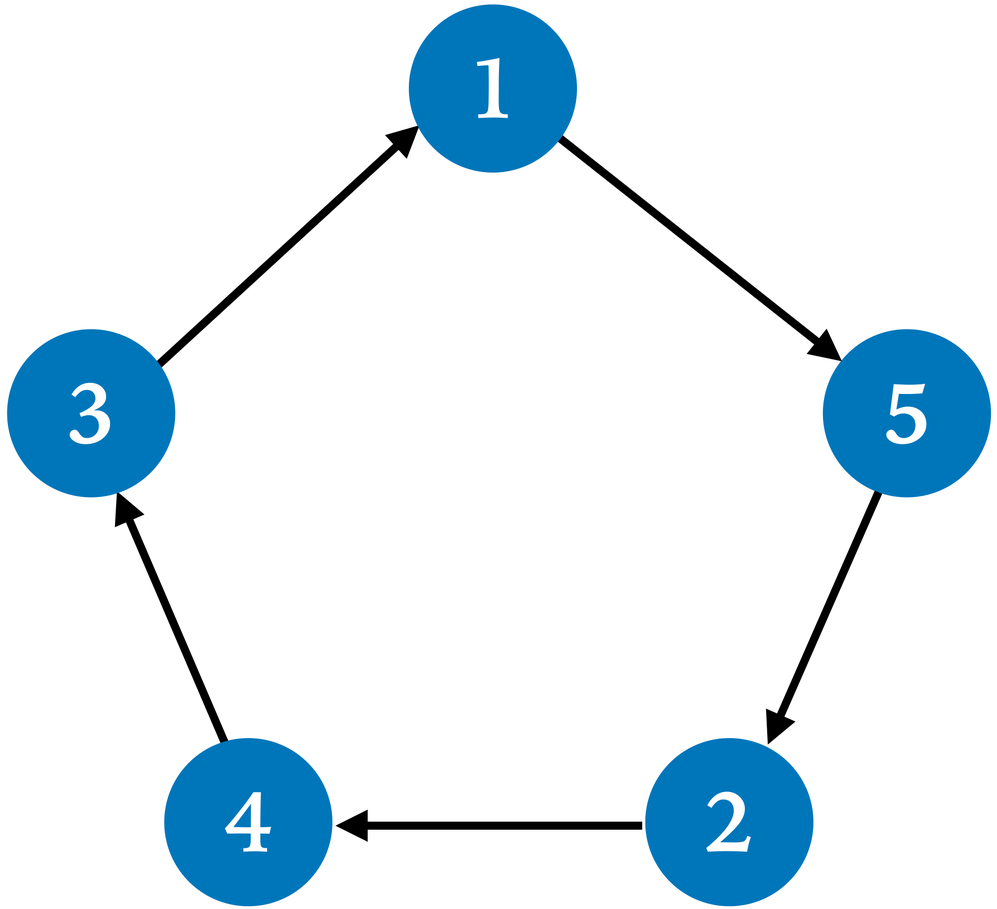
Ring Leader Election



Ring Leader Election

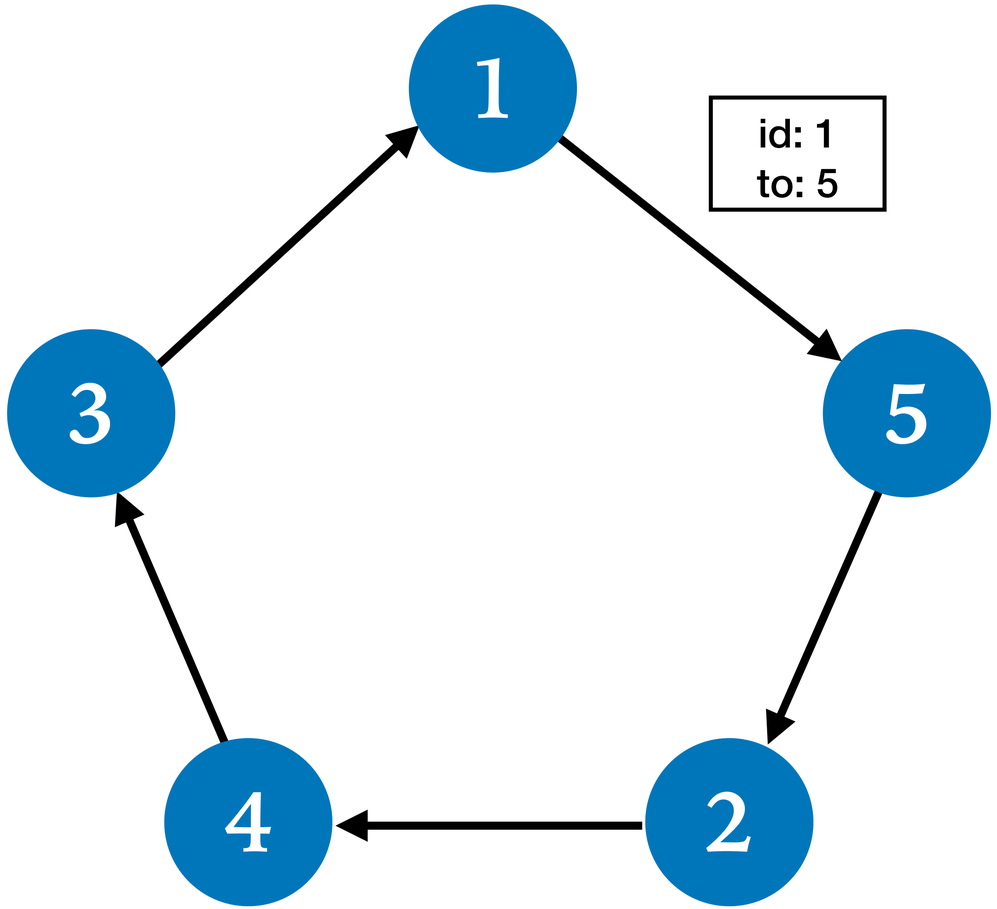


Ring Leader Election



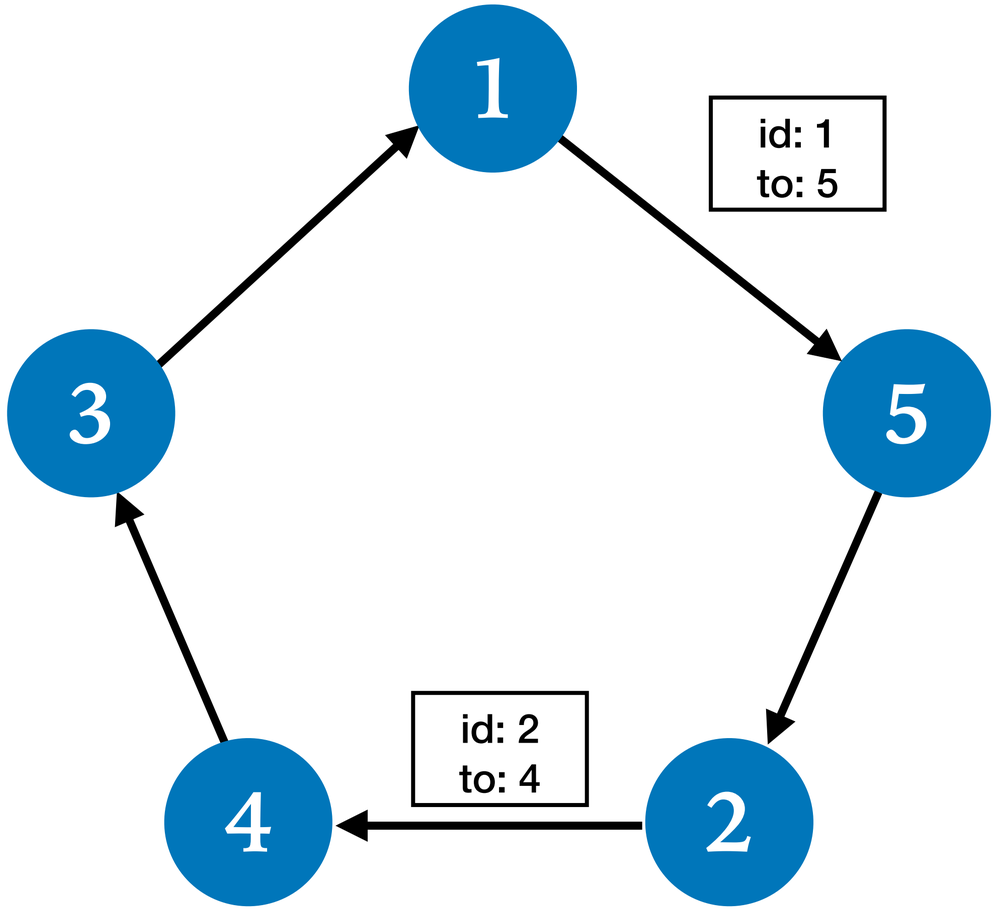
Node send its own
ID to its successor

Ring Leader Election



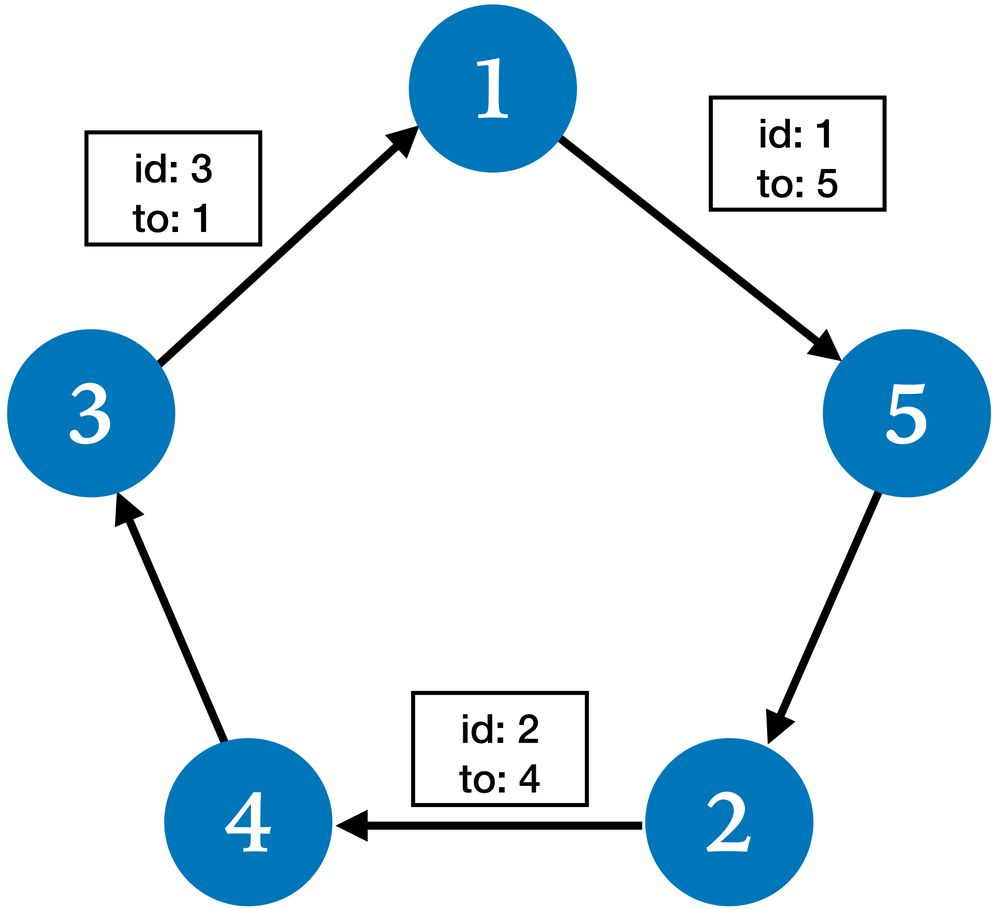
Node send its own
ID to its successor

Ring Leader Election



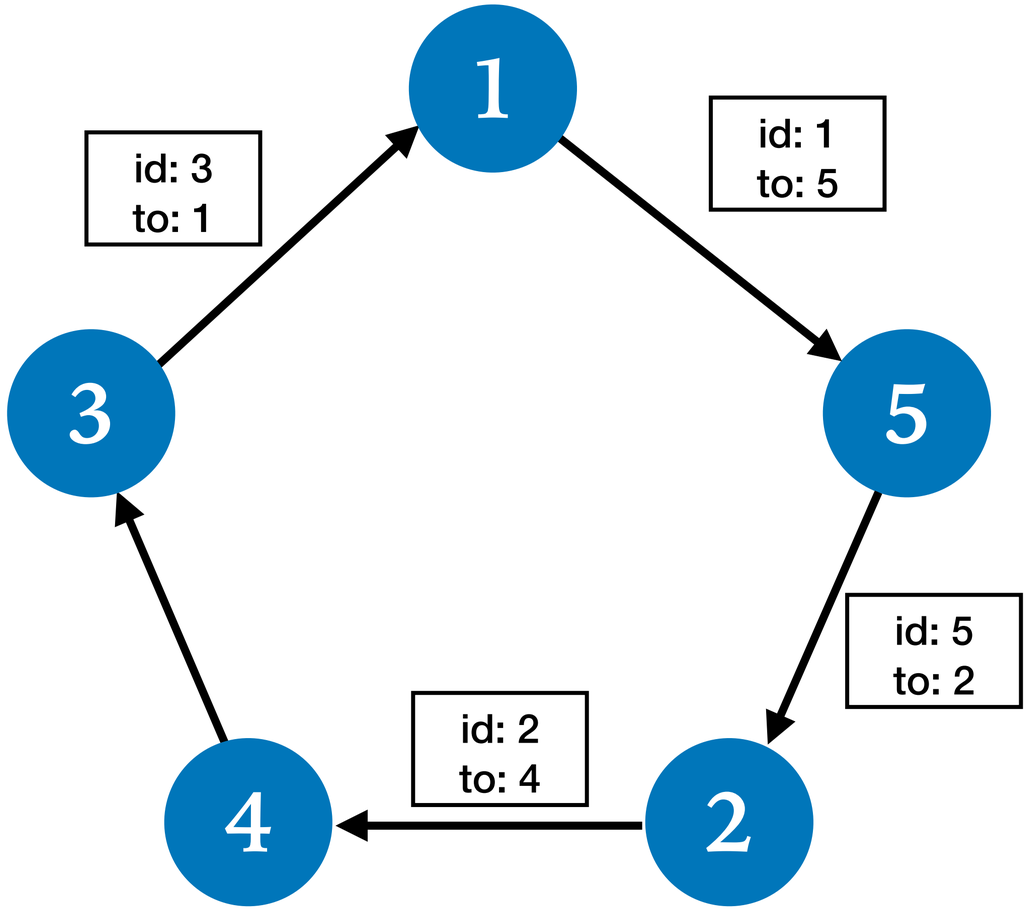
Node send its own
ID to its successor

Ring Leader Election



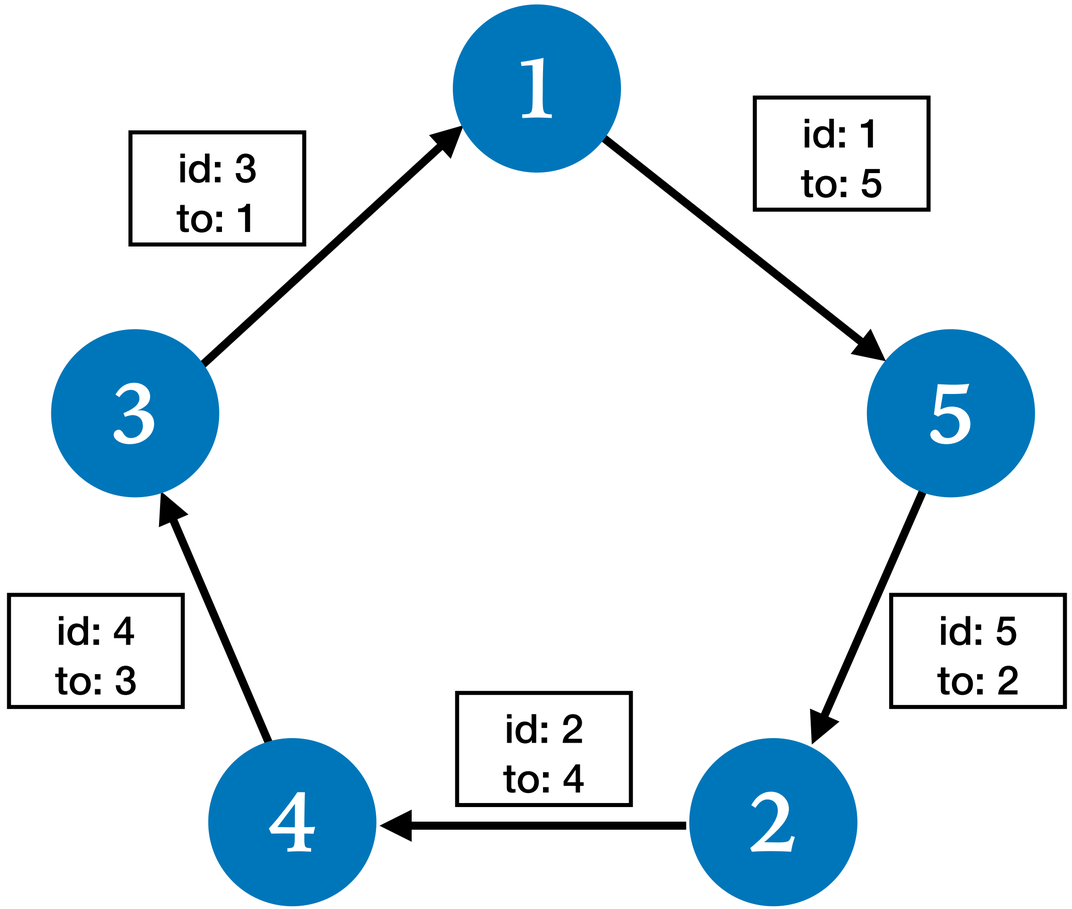
Node send its own ID to its successor

Ring Leader Election



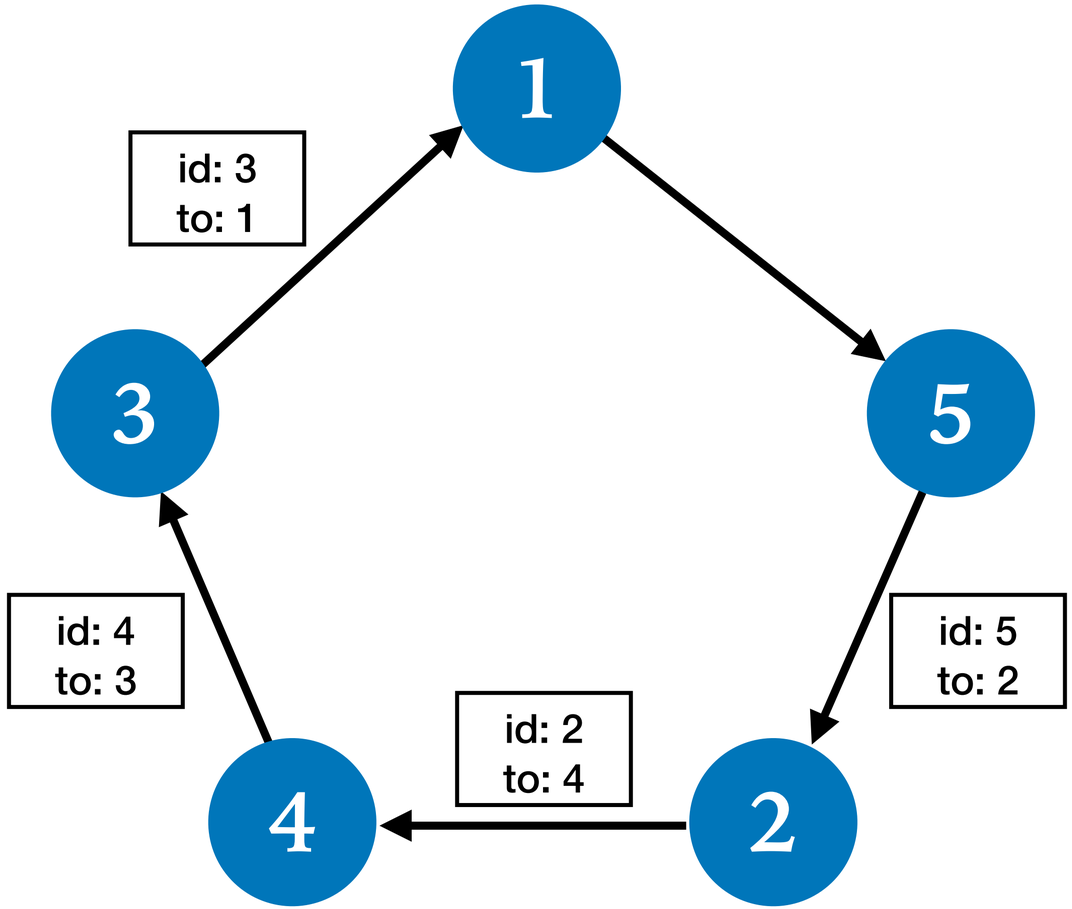
Node send its own ID to its successor

Ring Leader Election



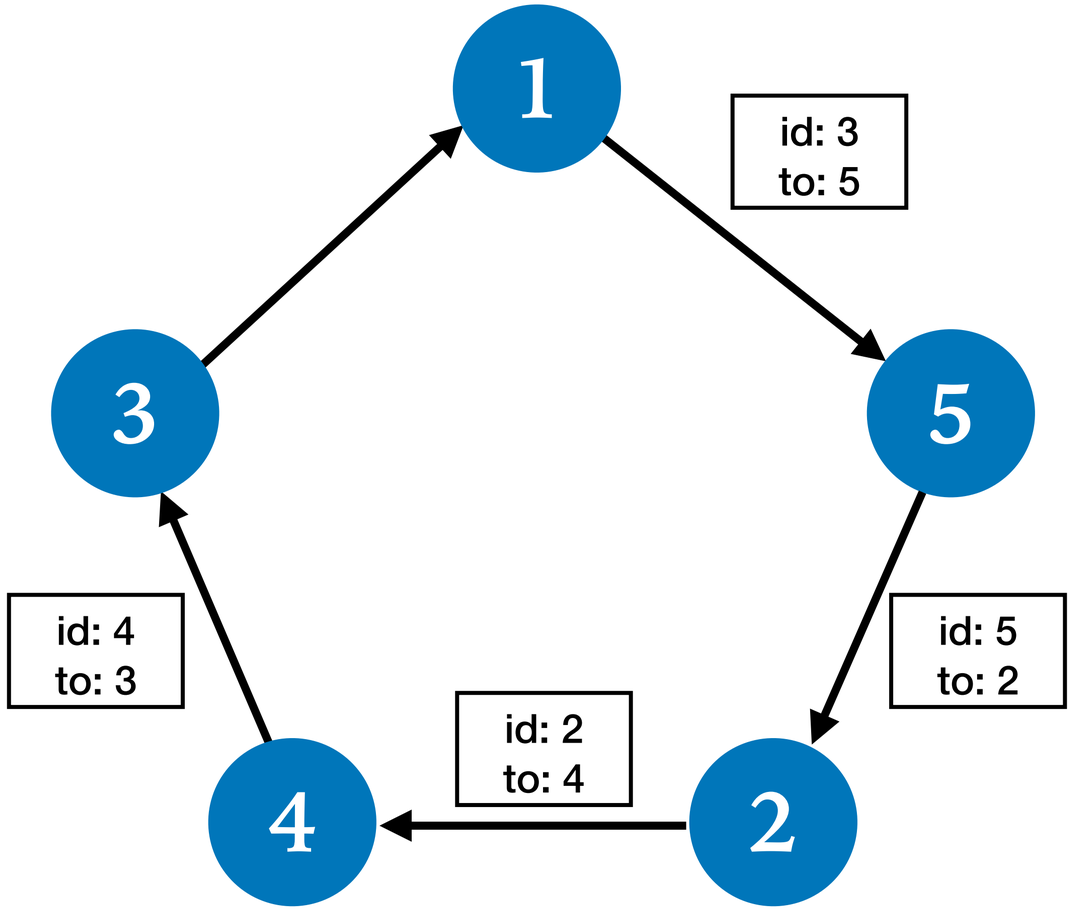
Node send its own ID to its successor

Ring Leader Election



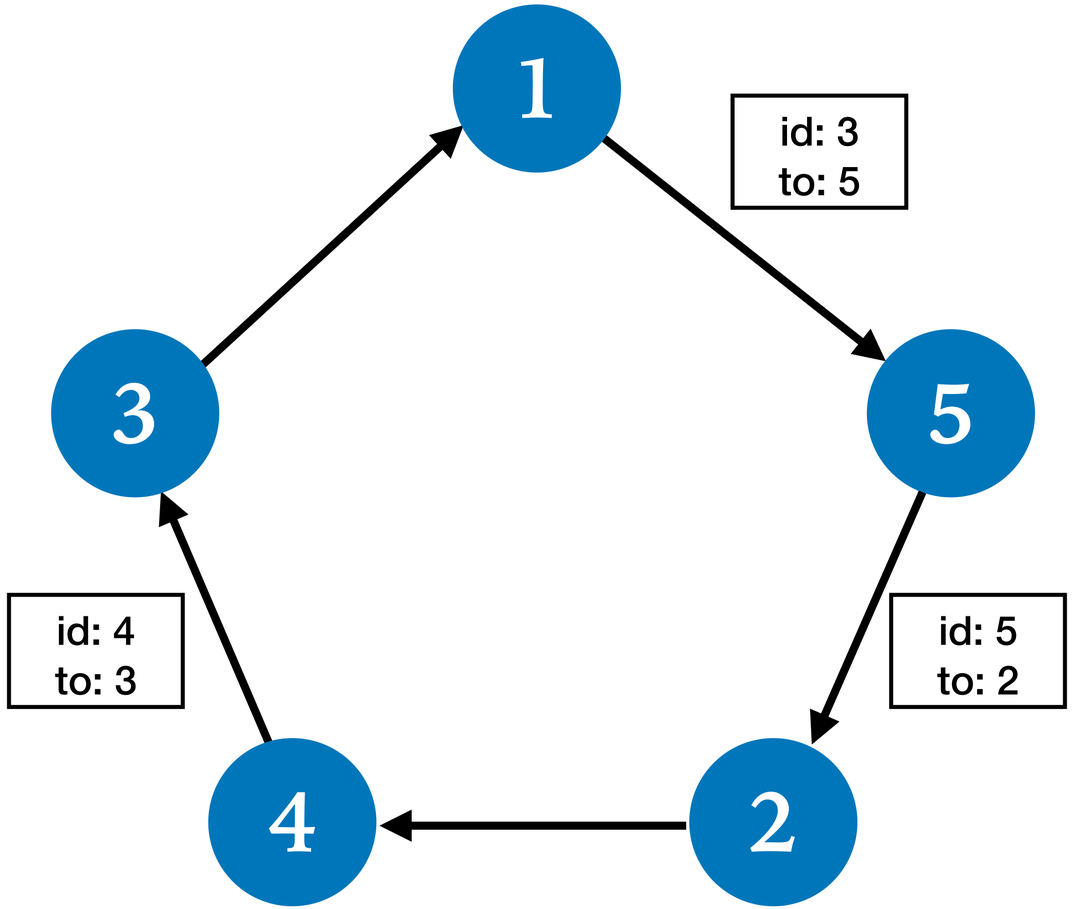
Node discards message
whose ID is smaller than its own

Ring Leader Election

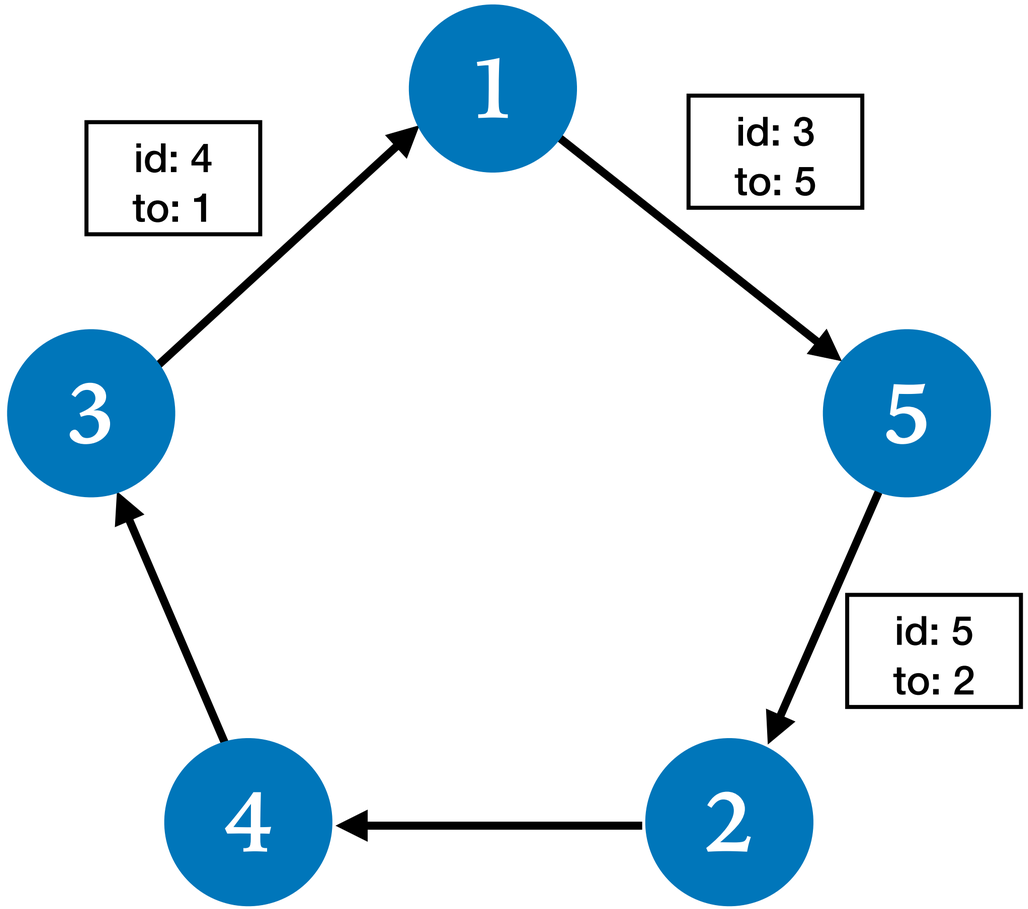


Node forwards message
whose ID is larger than its own

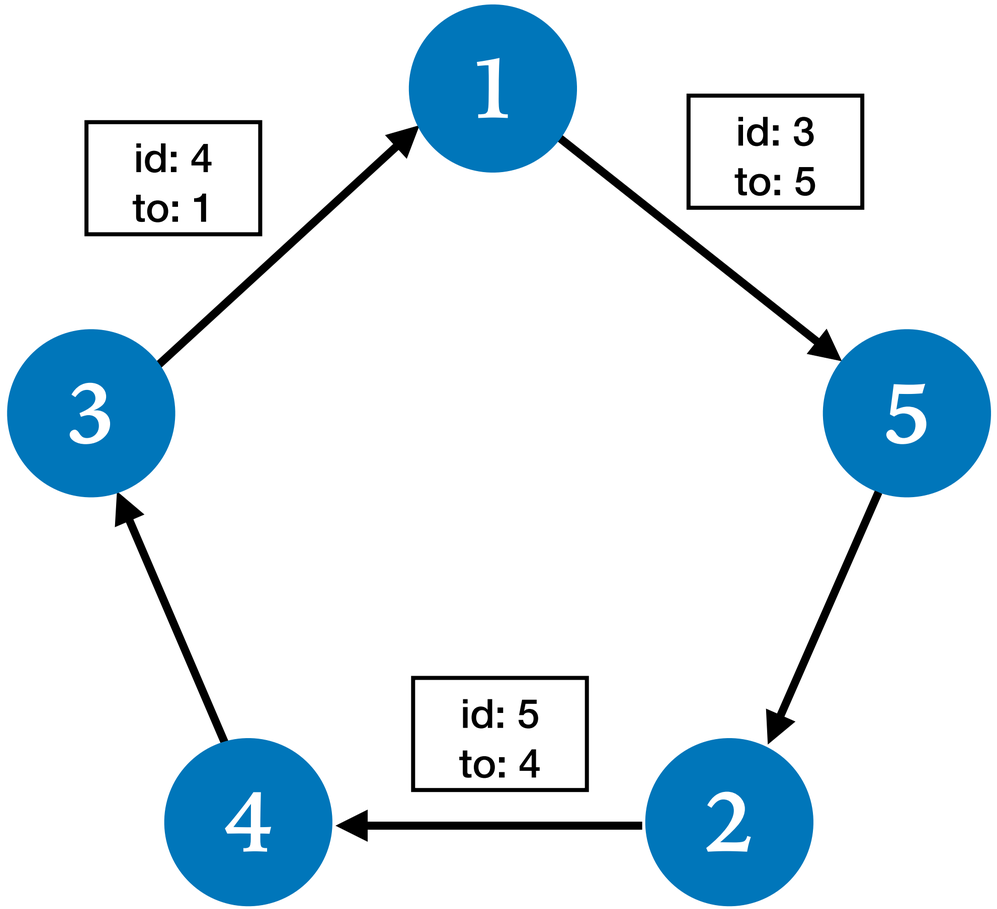
Ring Leader Election



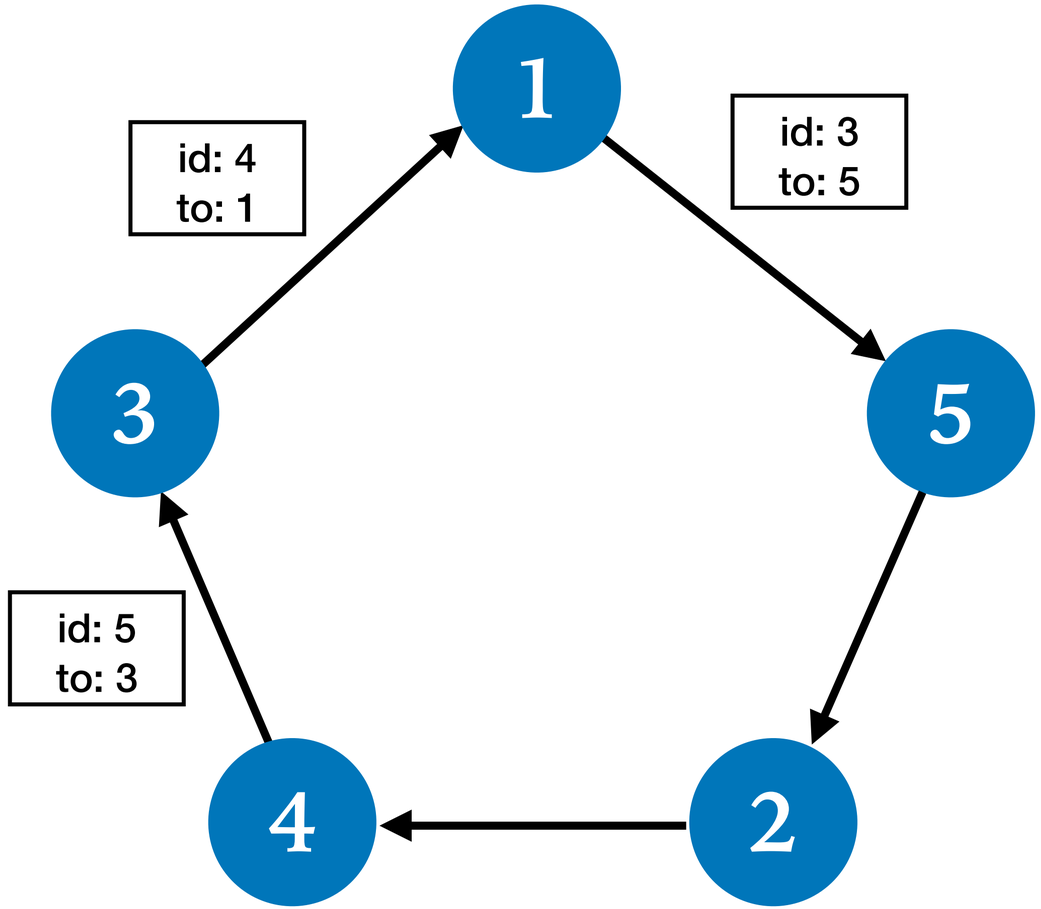
Ring Leader Election



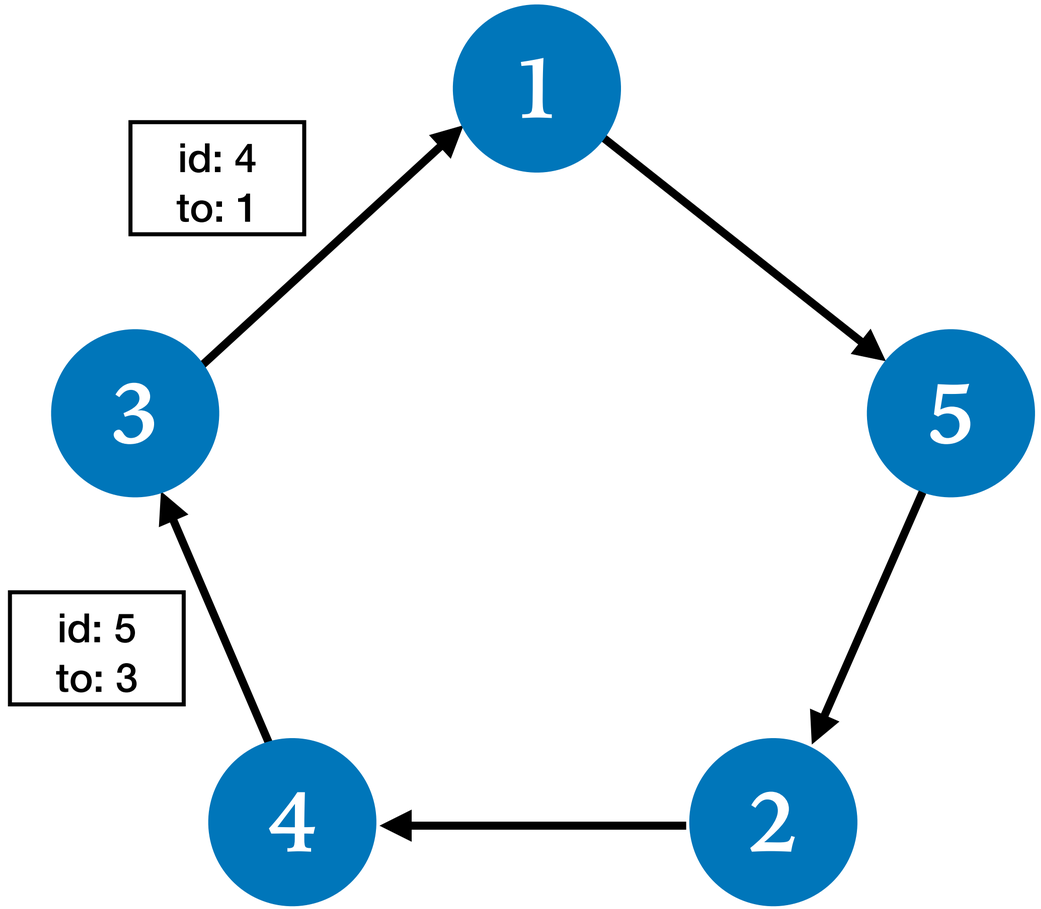
Ring Leader Election



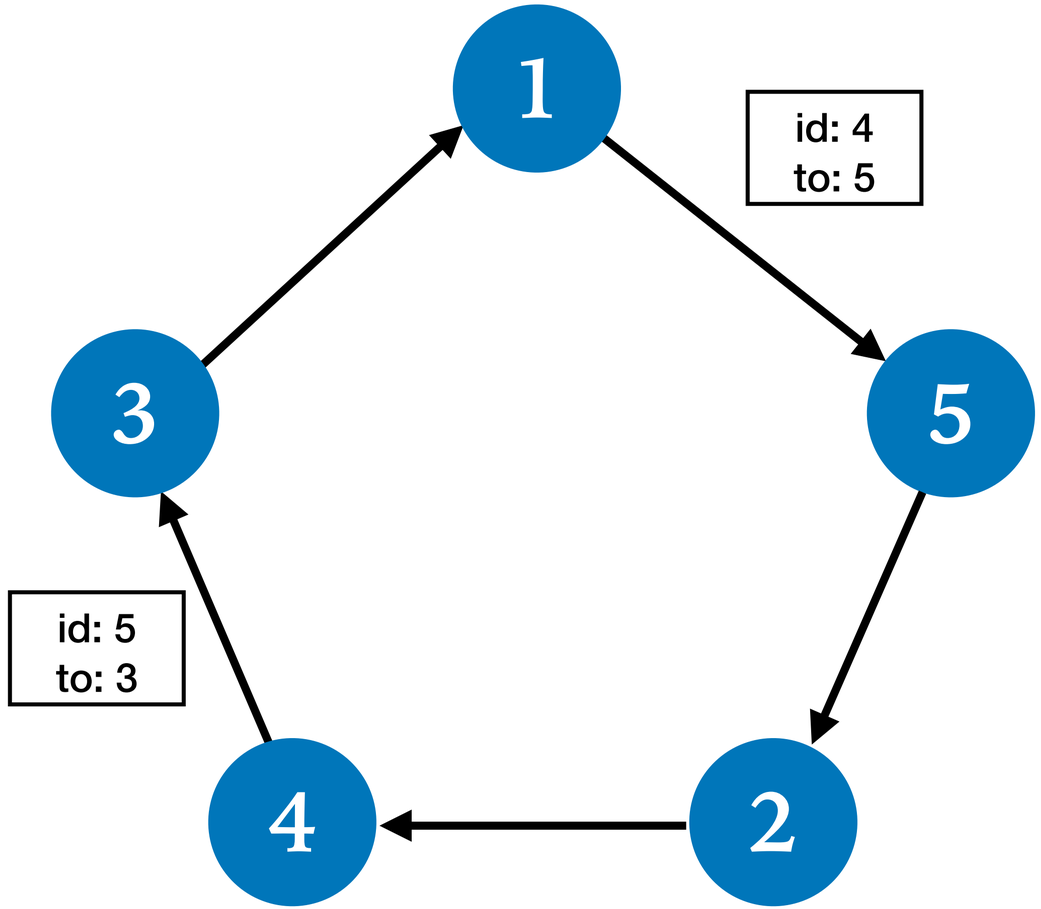
Ring Leader Election



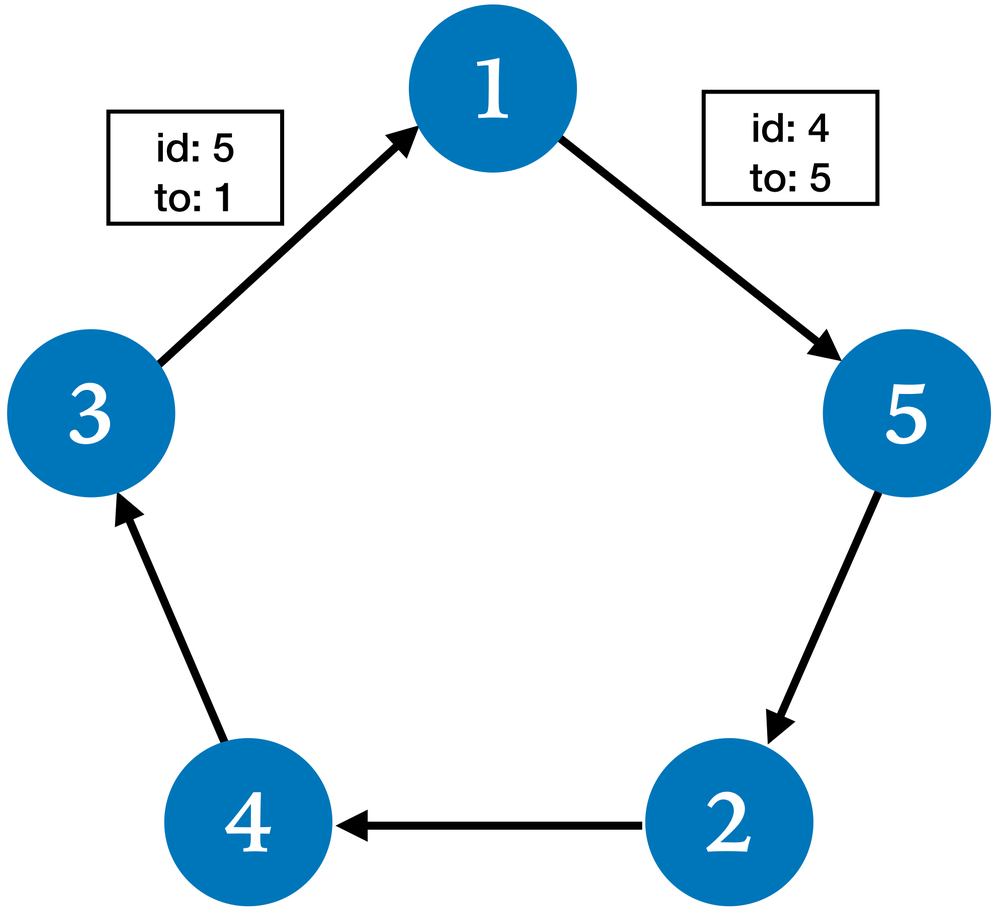
Ring Leader Election



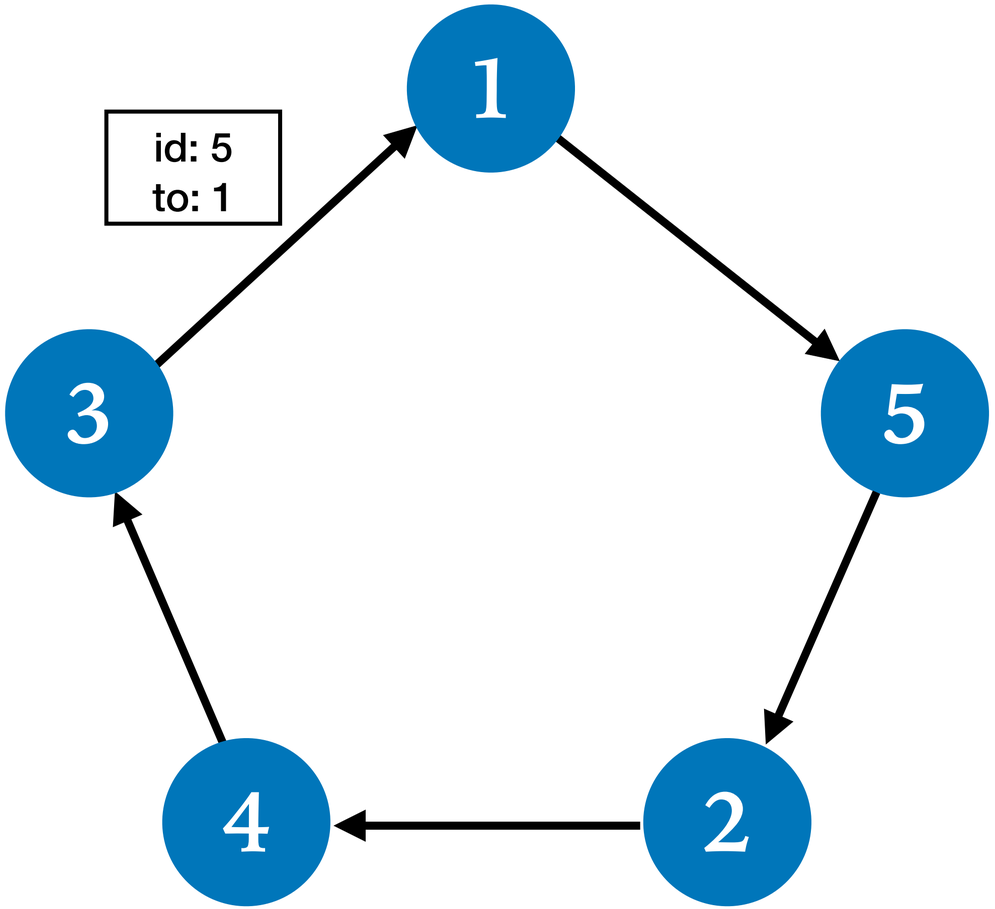
Ring Leader Election



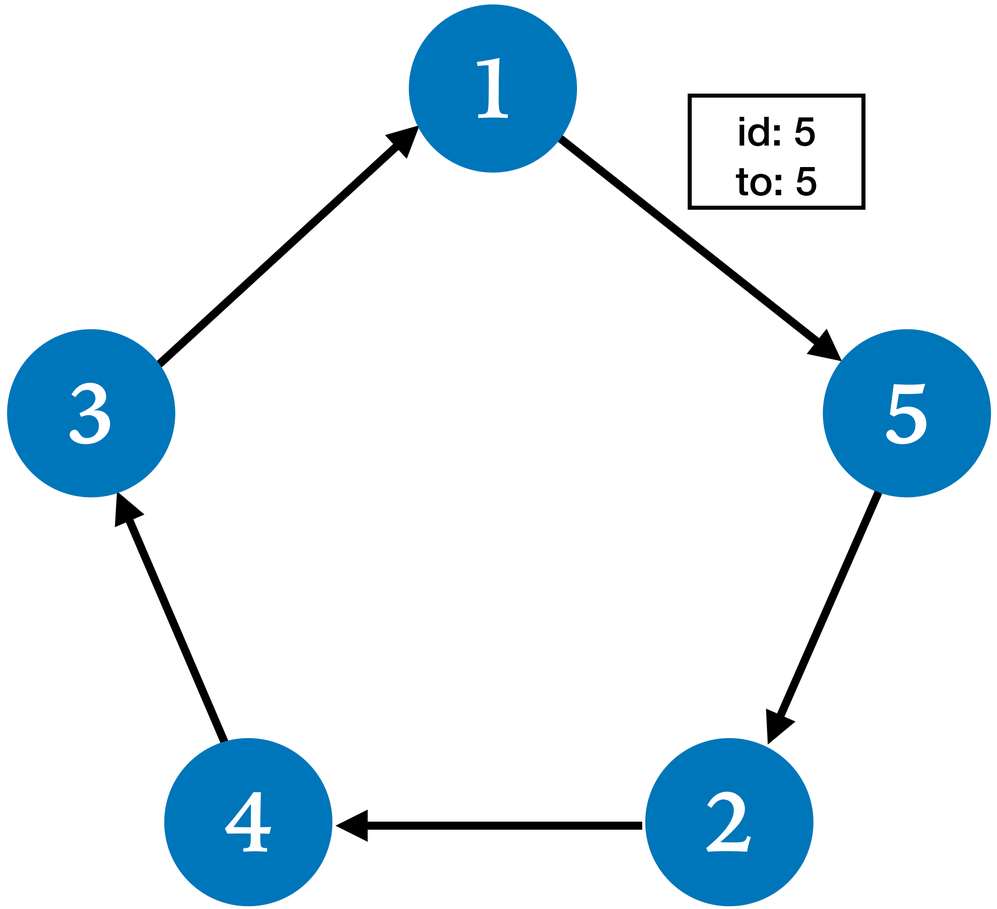
Ring Leader Election



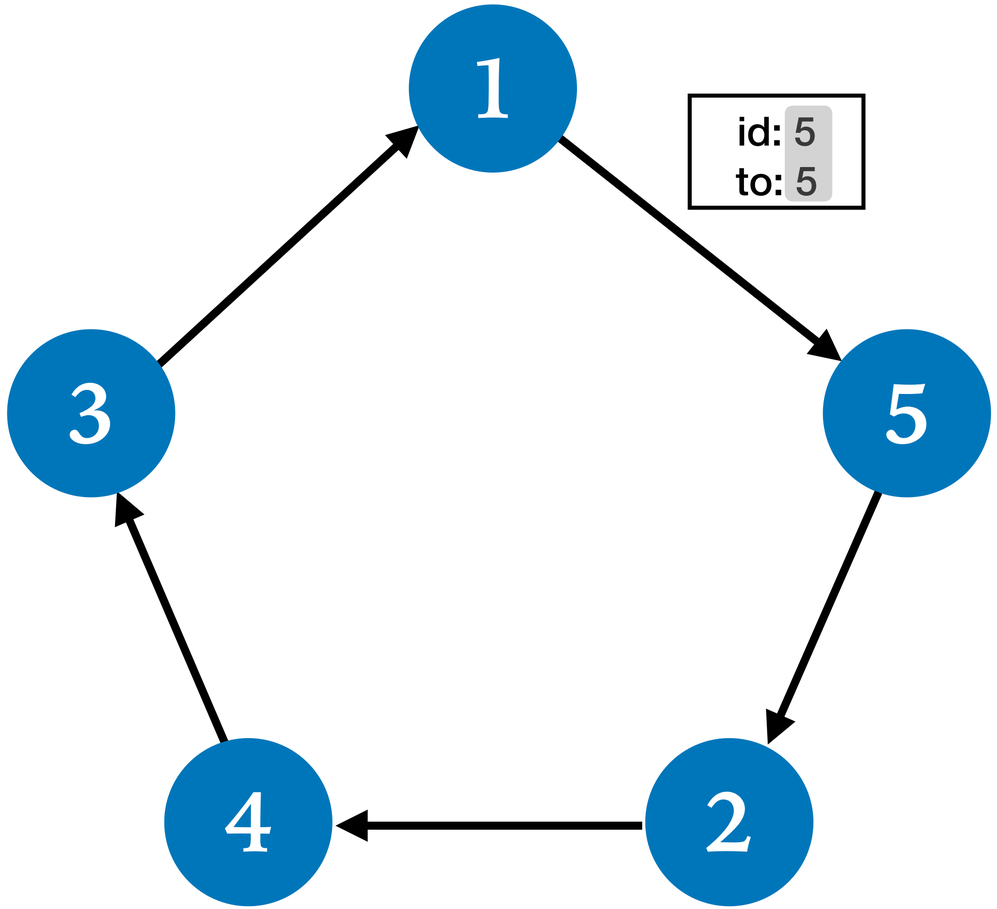
Ring Leader Election



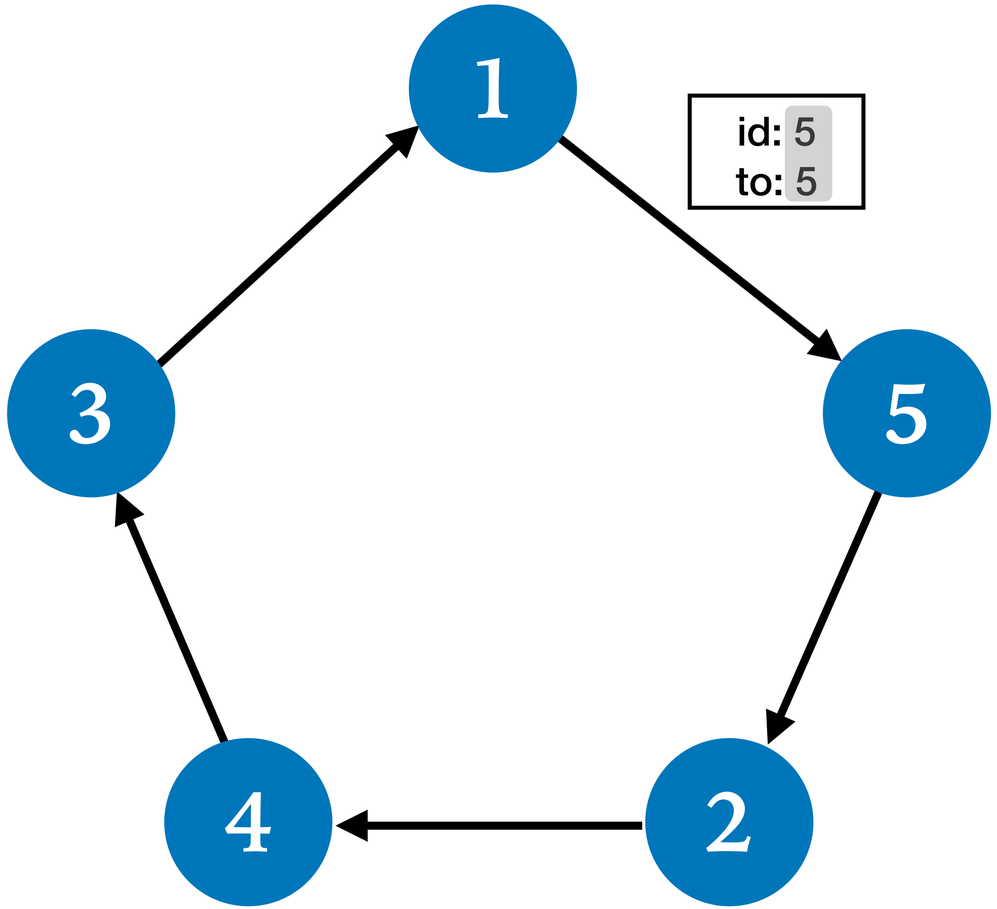
Ring Leader Election



Ring Leader Election

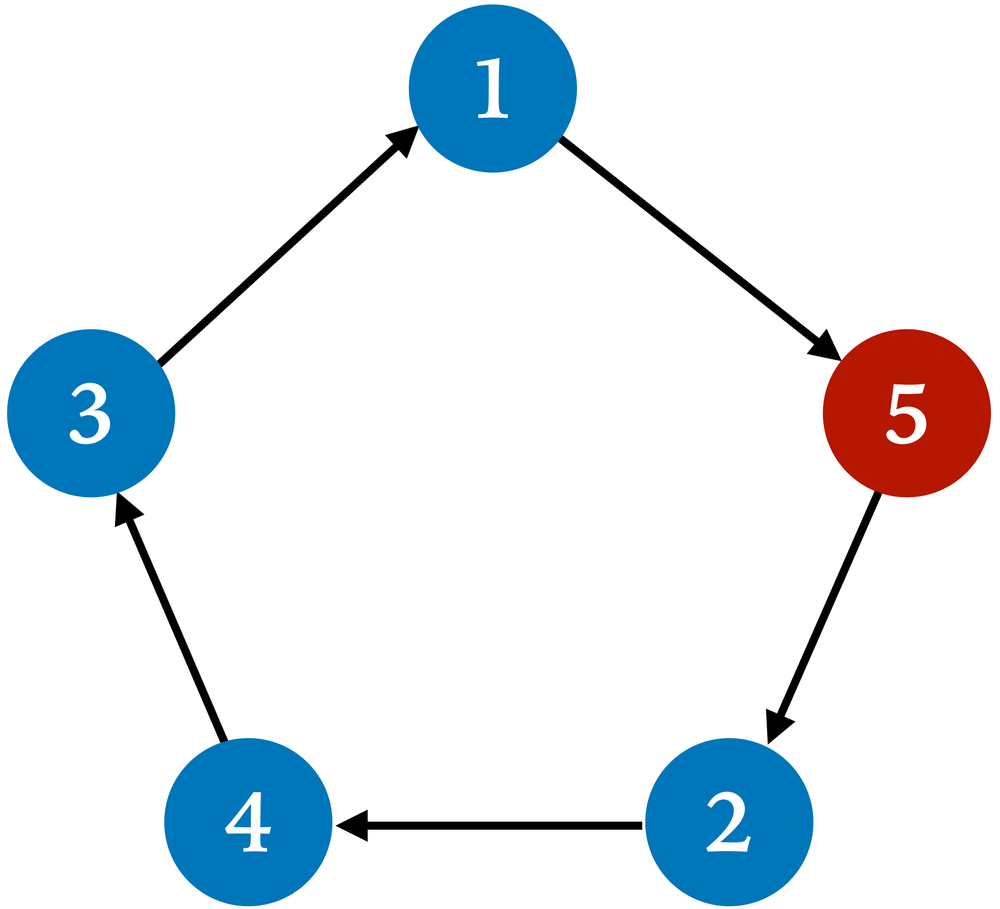


Ring Leader Election



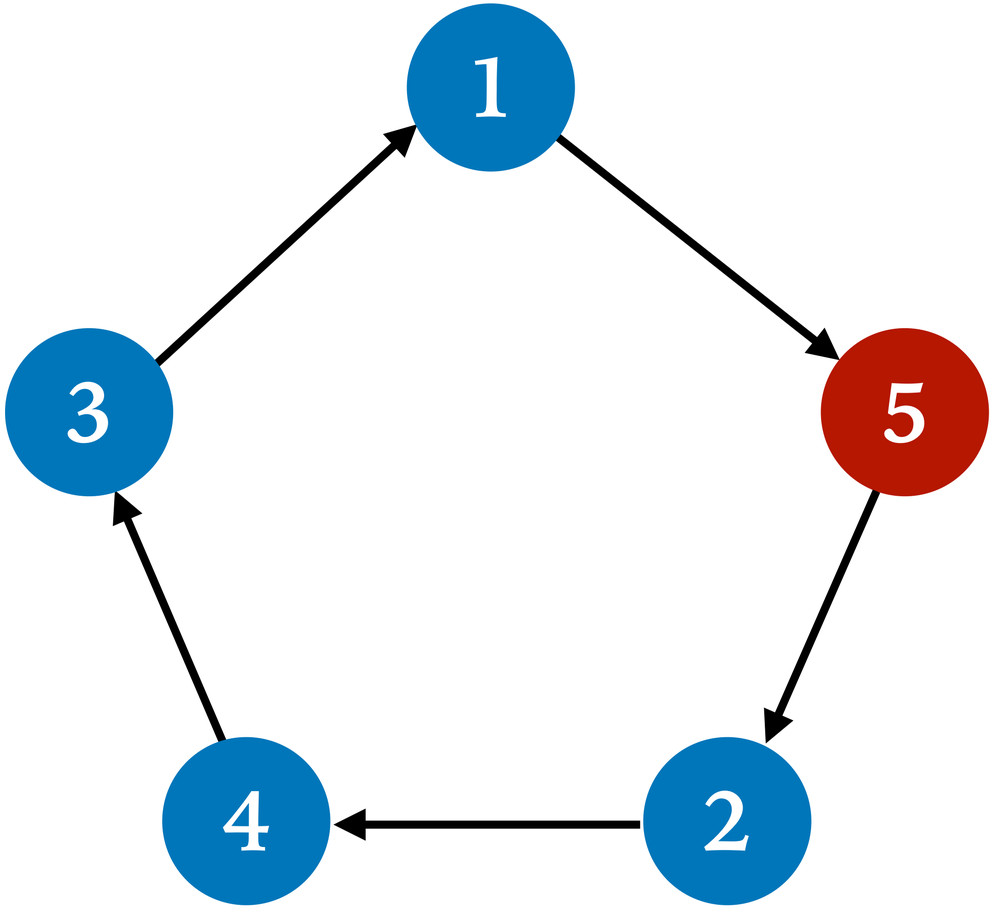
Node declares itself leader on receiving a message with its own ID

Ring Leader Election

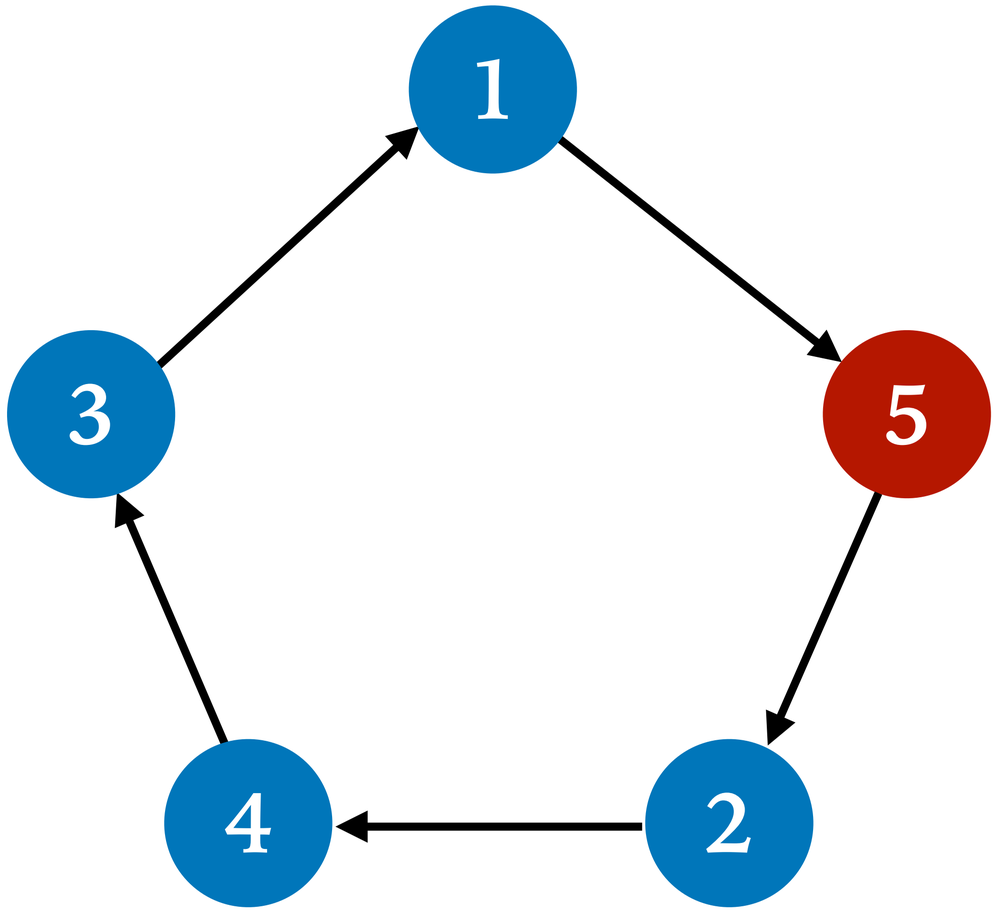


leader elected

Safety Property to Verify



Safety Property to Verify



There is at most one leader.

Demo

Veil

- ✅ A verifier for distributed protocols, embedded in Lean
- ✅ TLC-style explicit state model checking
- ✅ Symbolic model checking via SMT
- ✅ Out-of-the-box interactive proofs in Lean
- ✅ Small trusted computing base

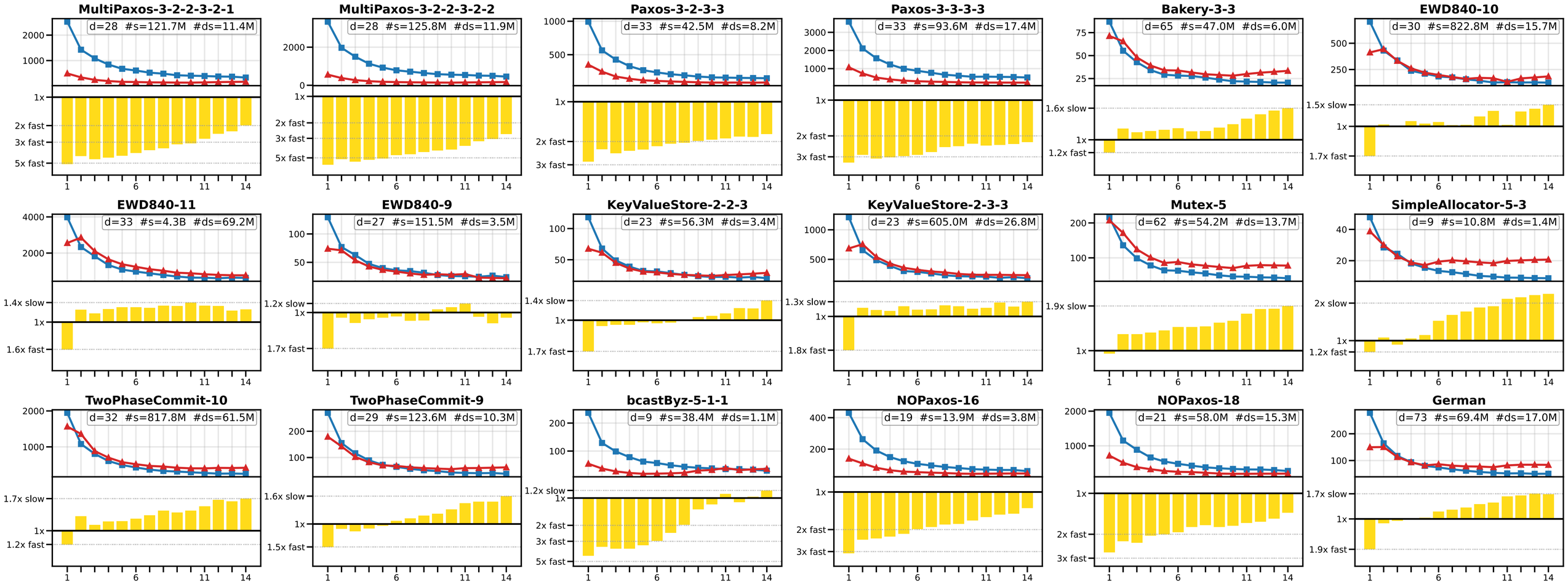
try.veil.dev

Model Checking Performance

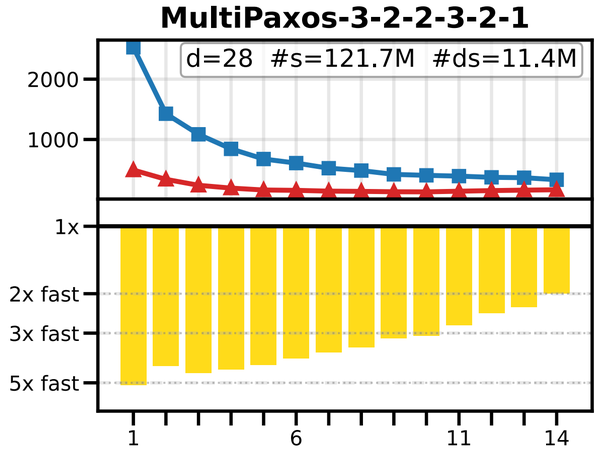
Lace

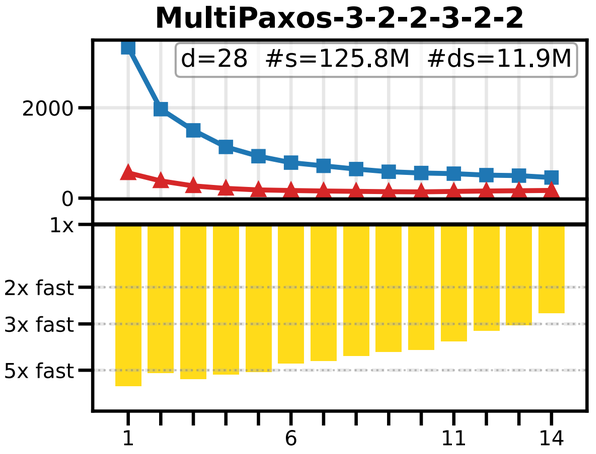
- Lace is a **formally verified** parallel model checker for Veil
  - Both sound and complete (assuming no hash collisions)
- Uses a map-reduce parallelisation strategy for easier verification
- Highly competitive performance

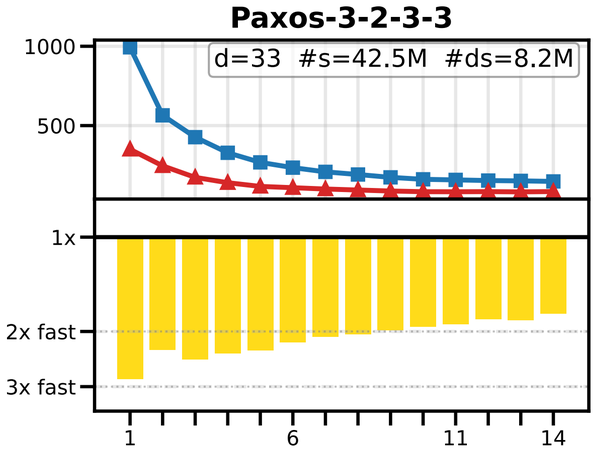
Lace vs TLC

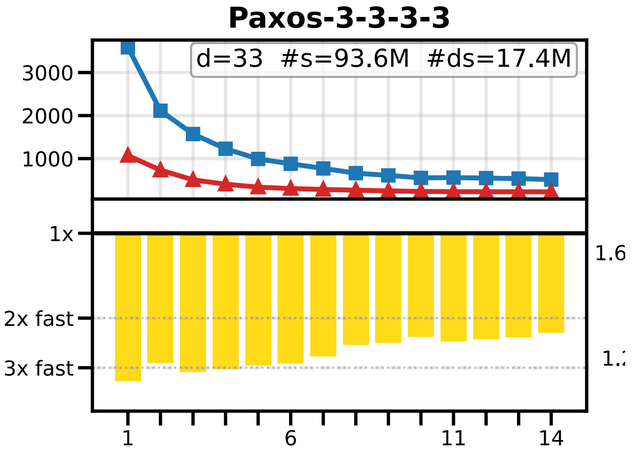

MultiPaxos-3-2-2-3-2-1
d=28 #s=121.7M #ds=11.4M
MultiPaxos-3-2-2-3-2-2
d=28 #s=125.8M #ds=11.9M
Paxos-3-2-3-3
d=33 #s=42.5M #ds=8.2M
Paxos-3-3-3-3
d=33 #s=93.6M #ds=17.4M
Bakery-3-3
d=65 #s=47.0M #ds=6.0M
EWD840-10
d=30 #s=822.8M #ds=15.7M
EWD840-11
d=33 #s=4.3B #ds=69.2M
EWD840-9
d=27 #s=151.5M #ds=3.5M
KeyValueStore-2-2-3
d=23 #s=56.3M #ds=3.4M
KeyValueStore-2-3-3
d=23 #s=605.0M #ds=26.8M
Mutex-5
d=62 #s=54.2M #ds=13.7M
SimpleAllocator-5-3
d=9 #s=10.8M #ds=1.4M
TwoPhaseCommit-10
d=32 #s=817.8M #ds=61.5M
TwoPhaseCommit-9
d=29 #s=123.6M #ds=10.3M
bcastByz-5-1-1
d=9 #s=38.4M #ds=1.1M
NOPaxos-16
d=19 #s=13.9M #ds=3.8M
NOPaxos-18
d=21 #s=58.0M #ds=15.3M
German
d=73 #s=69.4M #ds=17.0M

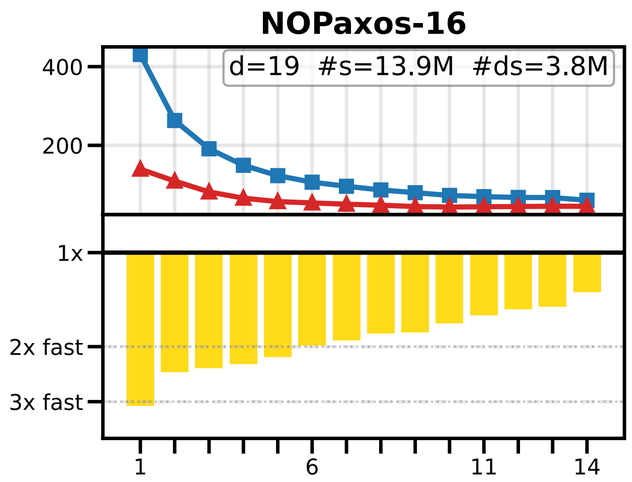
Lace vs TLC

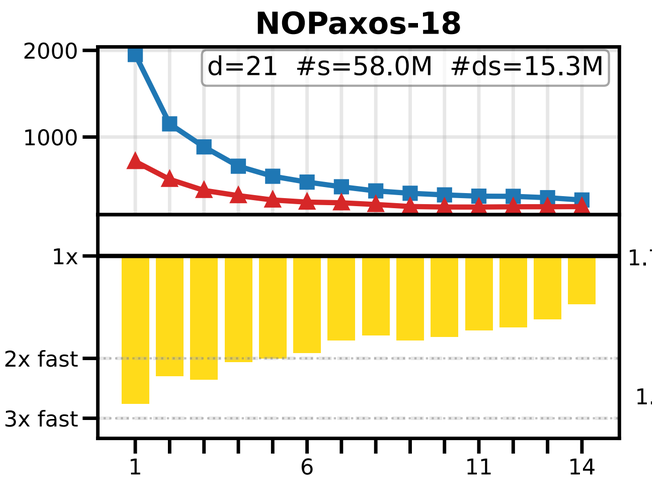

MultiPaxos-3-2-2-3-2-1
d=28 #s=121.7M #ds=11.4M
2000
1000
1x
2x fast
3x fast
5x fast
1
6
11
14


MultiPaxos-3-2-2-3-2-2
d=28 #s=125.8M #ds=11.9M
2000
0
1x
2x fast
3x fast
5x fast
1
6
11
14

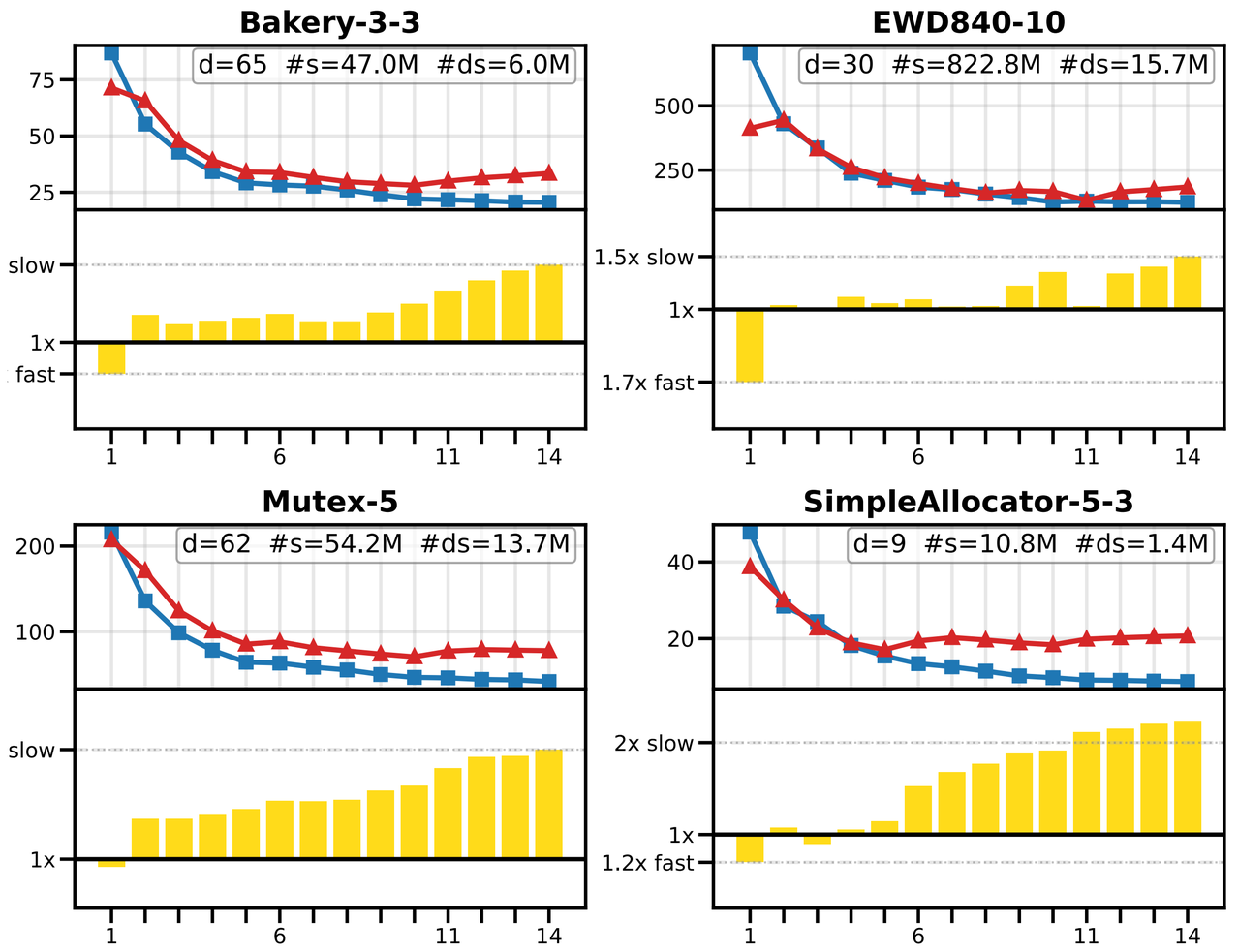

Paxos-3-2-3-3
d=33 #s=42.5M #ds=8.2M
1000
500
1x
2x fast
3x fast
1
6
11
14


Paxos-3-3-3-3
d=33 #s=93.6M #ds=17.4M
3000
2000
1000
1x
2x fast
3x fast
1
6
11
14

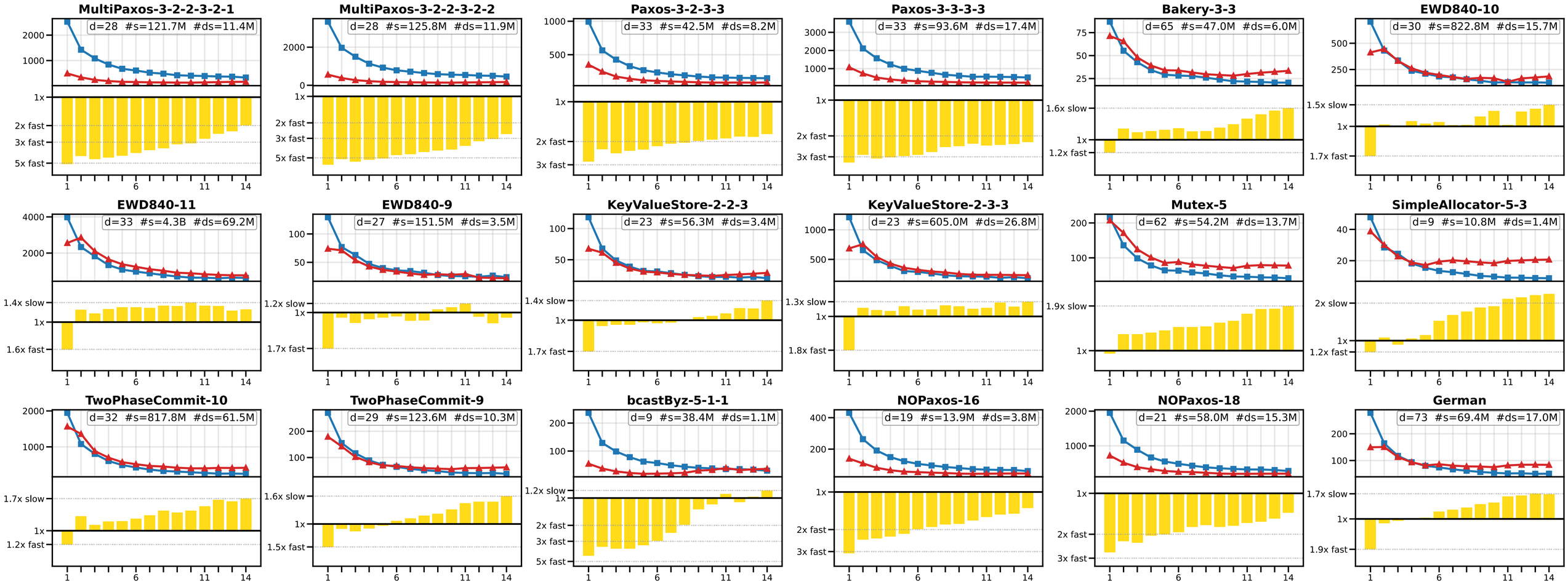

NOPaxos-16
d=19 #s=13.9M #ds=3.8M
400
200
1x
2x fast
3x fast
1
6
11
14


NOPaxos-18
d=21 #s=58.0M #ds=15.3M
2000
1000
1x
2x fast
3x fast
1
6
11
14

Lace vs TLC


Bakery-3-3
d=65 #s=47.0M #ds=6.0M
75
50
25
slow
1x
fast
1
6
11
14
EWD840-10
d=30 #s=822.8M #ds=15.7M
500
250
1.5x slow
1x
1.7x fast
1
6
11
14
Mutex-5
d=62 #s=54.2M #ds=13.7M
200
100
slow
1x
1
6
11
14
SimpleAllocator-5-3
d=9 #s=10.8M #ds=1.4M
40
20
2x slow
1x
1.2x fast
1
6
11
14

Lace vs TLC


MultiPaxos-3-2-2-3-2-1
d=28 #s=121.7M #ds=11.4M
MultiPaxos-3-2-2-3-2-2
d=28 #s=125.8M #ds=11.9M
Paxos-3-2-3-3
d=33 #s=42.5M #ds=8.2M
Paxos-3-3-3-3
d=33 #s=93.6M #ds=17.4M
Bakery-3-3
d=65 #s=47.0M #ds=6.0M
EWD840-10
d=30 #s=822.8M #ds=15.7M
EWD840-11
d=33 #s=4.3B #ds=69.2M
EWD840-9
d=27 #s=151.5M #ds=3.5M
KeyValueStore-2-2-3
d=23 #s=56.3M #ds=3.4M
KeyValueStore-2-3-3
d=23 #s=605.0M #ds=26.8M
Mutex-5
d=62 #s=54.2M #ds=13.7M
SimpleAllocator-5-3
d=9 #s=10.8M #ds=1.4M
TwoPhaseCommit-10
d=32 #s=817.8M #ds=61.5M
TwoPhaseCommit-9
d=29 #s=123.6M #ds=10.3M
bcastByz-5-1-1
d=9 #s=38.4M #ds=1.1M
NOPaxos-16
d=19 #s=13.9M #ds=3.8M
NOPaxos-18
d=21 #s=58.0M #ds=15.3M
German
d=73 #s=69.4M #ds=17.0M

Take Aways

is a multi-modal verifier for distributed protocols

- It combines explicit and symbolic model checking with automated and interactive proofs in a single cohesive tool
- High-performance verified concurrent model checker
- Future work: liveness support and refinement reasoning coming soon

veil.dev