Combinatorial Constructions in Testing Concurrent Programs

Rupak Majumdar

Max Planck Institute for Software Systems (MPI-SWS), Germany

Joint Work with Filip Niksic and Dmitry Chistikov

Despite Many Formal Approaches…

… practitioners test their code

… by providing random inputs

And despite our best judgment, …

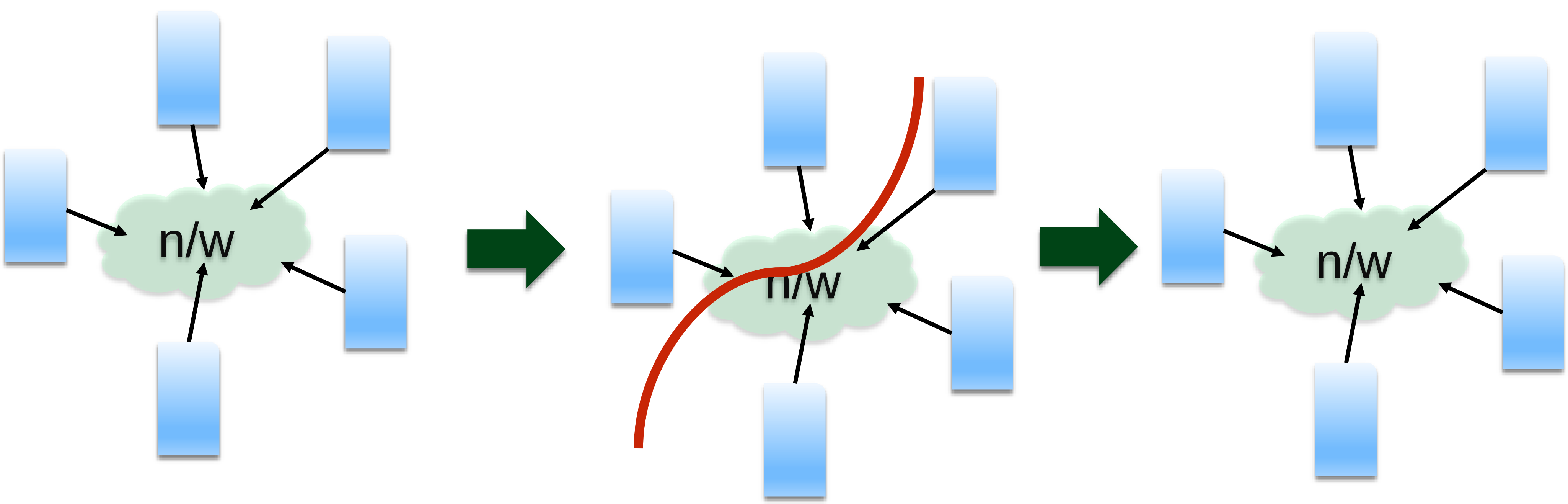
… testing is *surprisingly effective in finding bugs*

In this talk, we explore this unexpected effectiveness

Example: Jepsen

http://jepsen.io

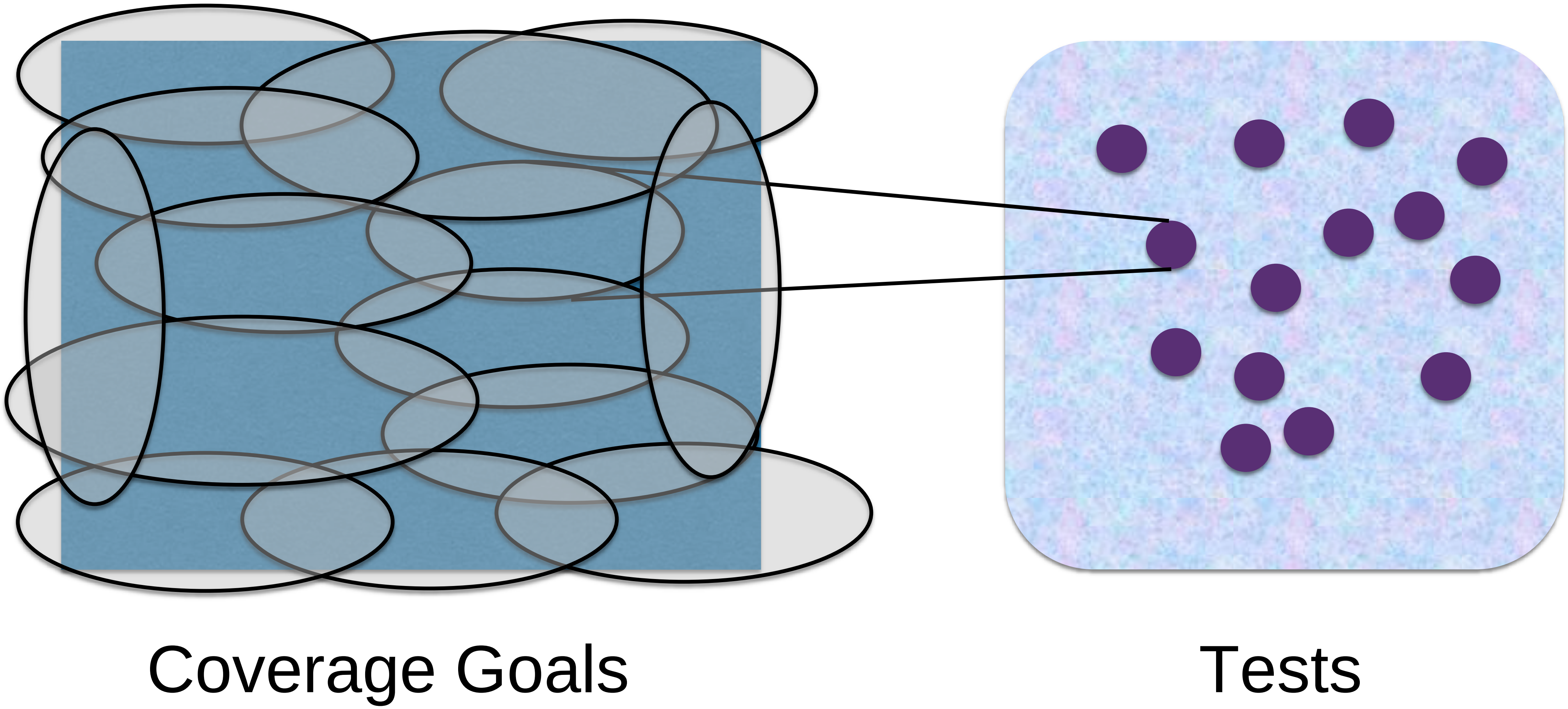
- A framework for black-box testing of distributed systems by randomly inserting network partition faults



etcd
Postgres
Redis
Riak
MongoDB
Cassandra
Chronos
Kafka
RabbitMQ
Consul
Elasticsearch
Aerospike
Zookeeper

- CAP Theorem: No system has consistency, availability, and partition tolerance

Tests and Coverage

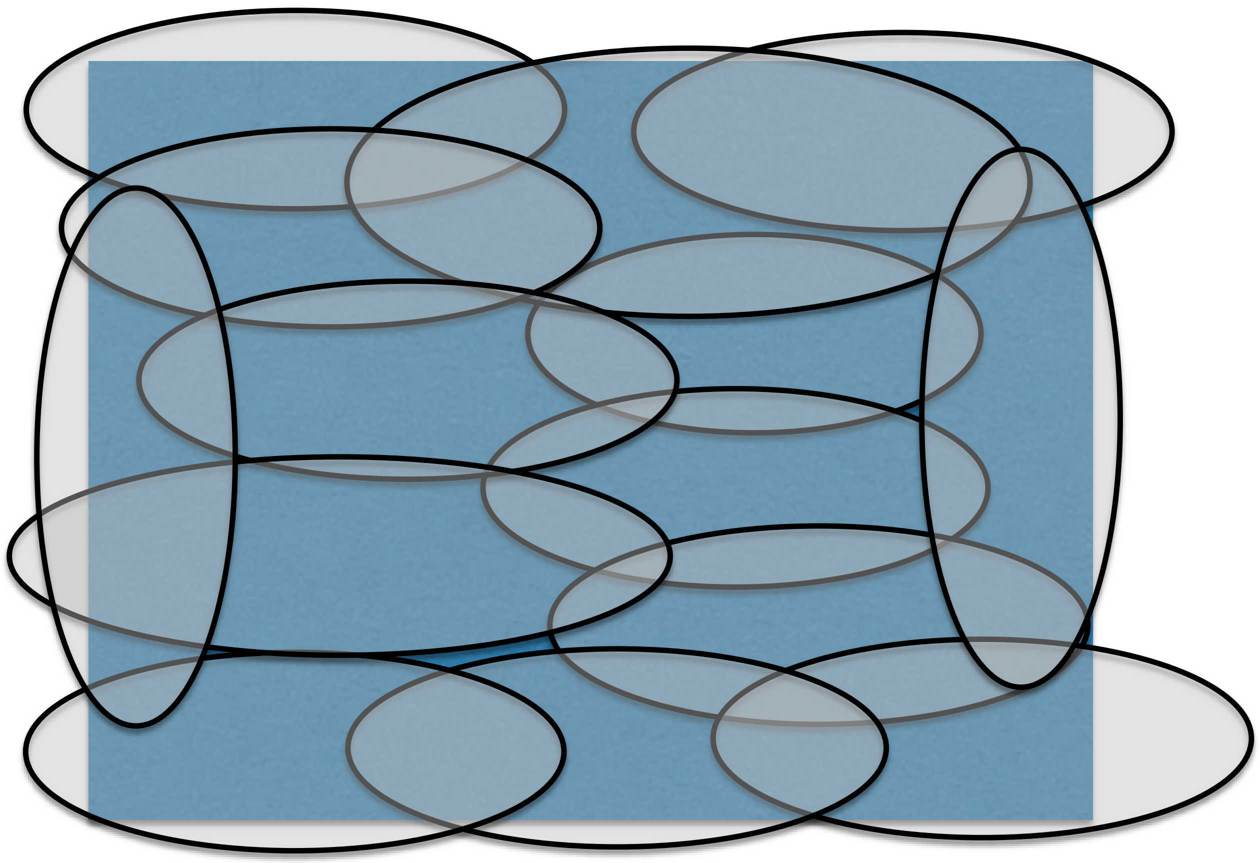


Covering Family of Tests = Set of tests covering all goals

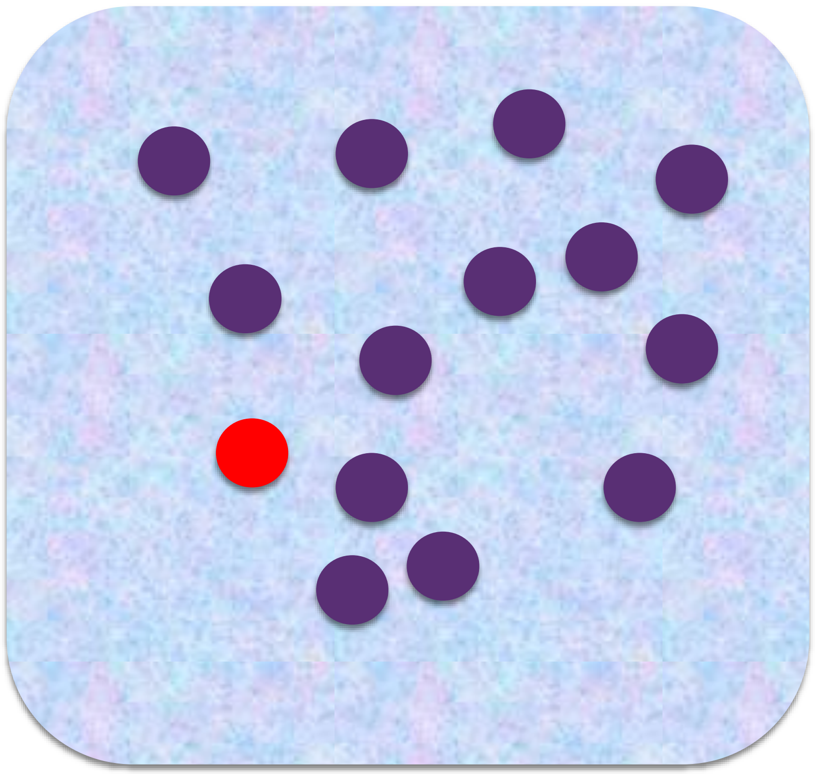
"Small" covering family = More efficient testing

Random Testing

Pick tests at random, independently



Coverage Goals

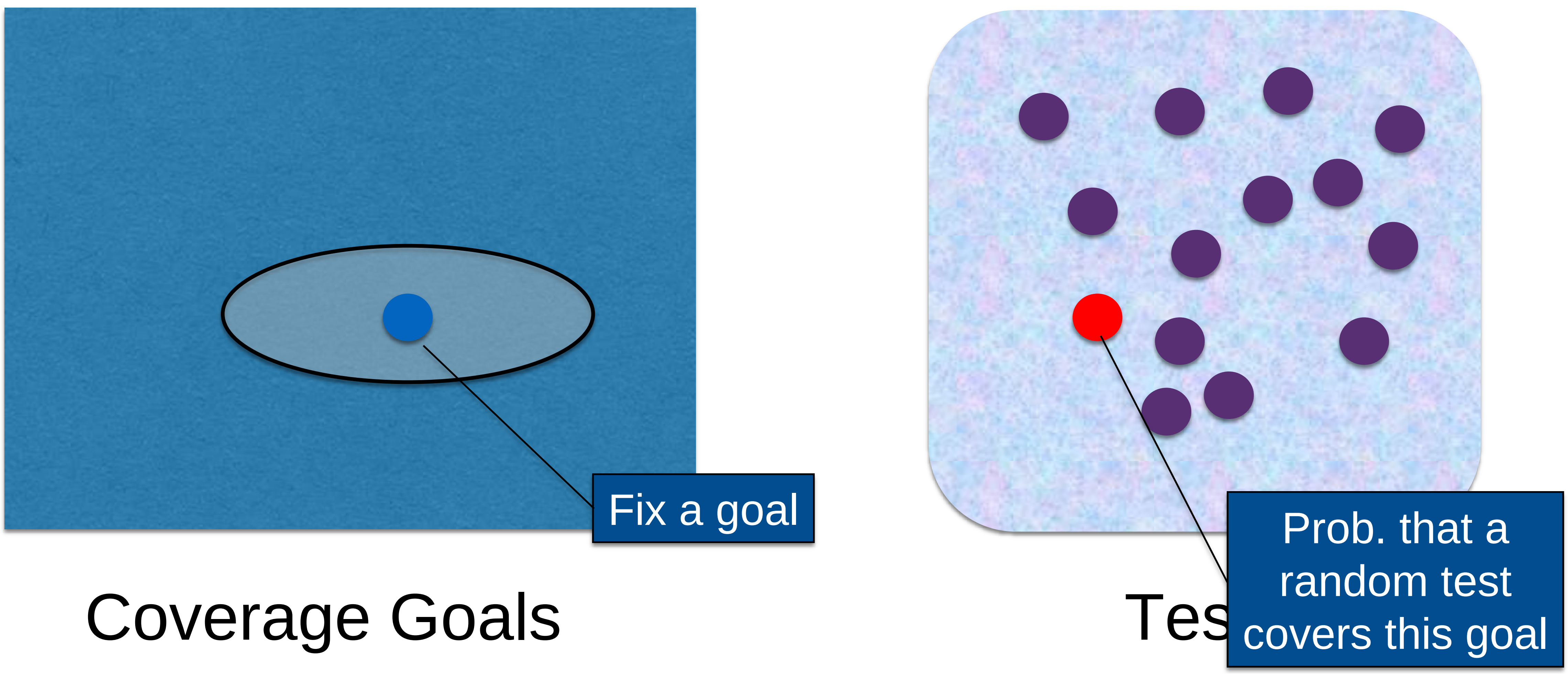


Tests

Question: Will random testing find a small covering family?

Random Testing

Pick tests at random, independently



Suppose Pr[Test covers goal] ≥ p

Goal: Characterize covering families w.r.t. p and |G|

Probabilistic Method

Let G be the set of goals and Pr[Random test covers a goal] ≥ p

Theorem. *There exists a covering family of size p^{-1} log|G|.*

Proof.

Pr[Random test does not cover goal g] ≤ 1 – p

Pr[K ind tests do not cover goal g] ≤ $(1 - p)^K$

Pr[K ind tests is not a covering family] ≤ |G| $(1 - p)^K$

For K = p^{-1} log|G|, this probability is < 1

Then: *There must exist K tests that is a covering family!*

$\boxed{\text{Existence, not constructive!}}$

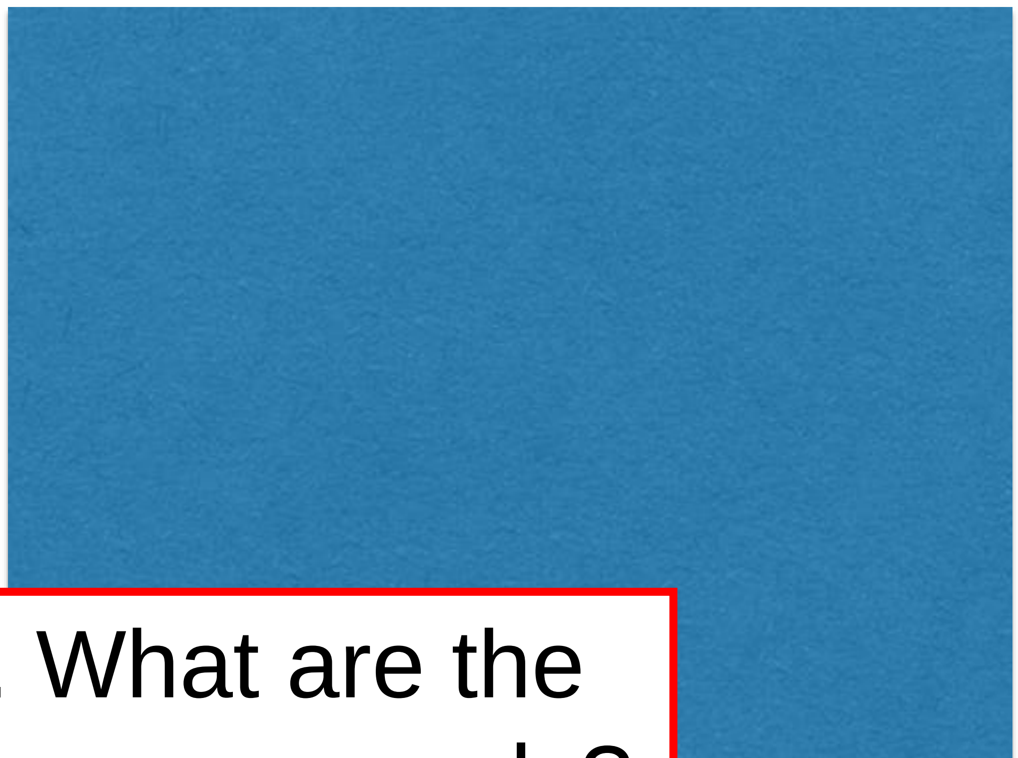
Probabilistic Method

Let G be the set of goals and Pr[Random test covers a goal] $\geq$ p

Theorem. *There exists a covering family of size $p^{-1} \log|G|$.*

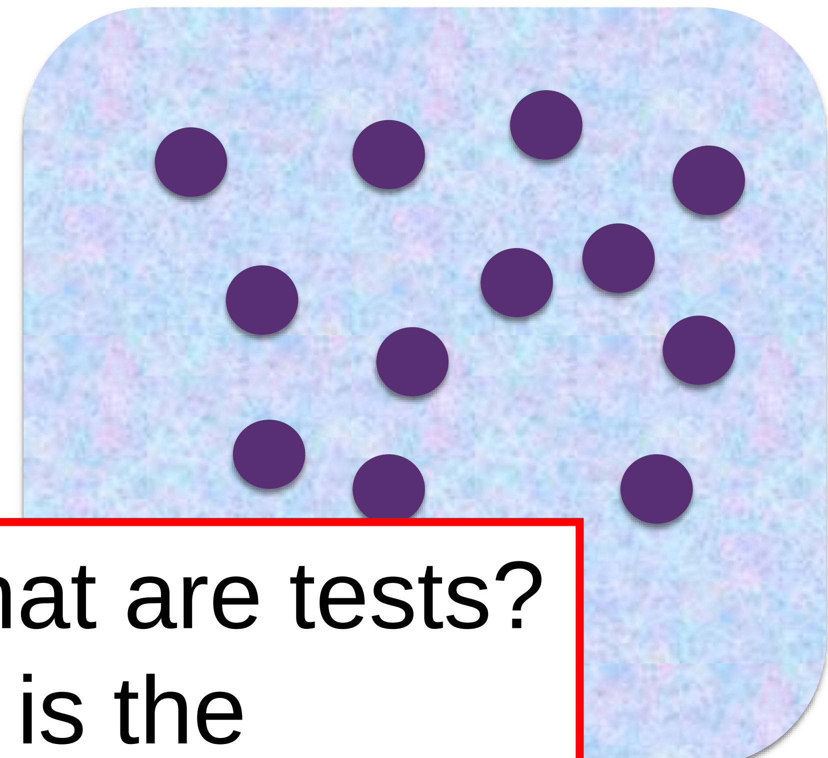
By repeated sampling, random testing overwhelmingly likely to find a covering family!

Tests and Coverage



1. What are the coverage goals?

Coverage Goals



2. What are tests? What is the Probability space?

Tests

3. Can we bound Pr[Test covers a goal]?

Plan

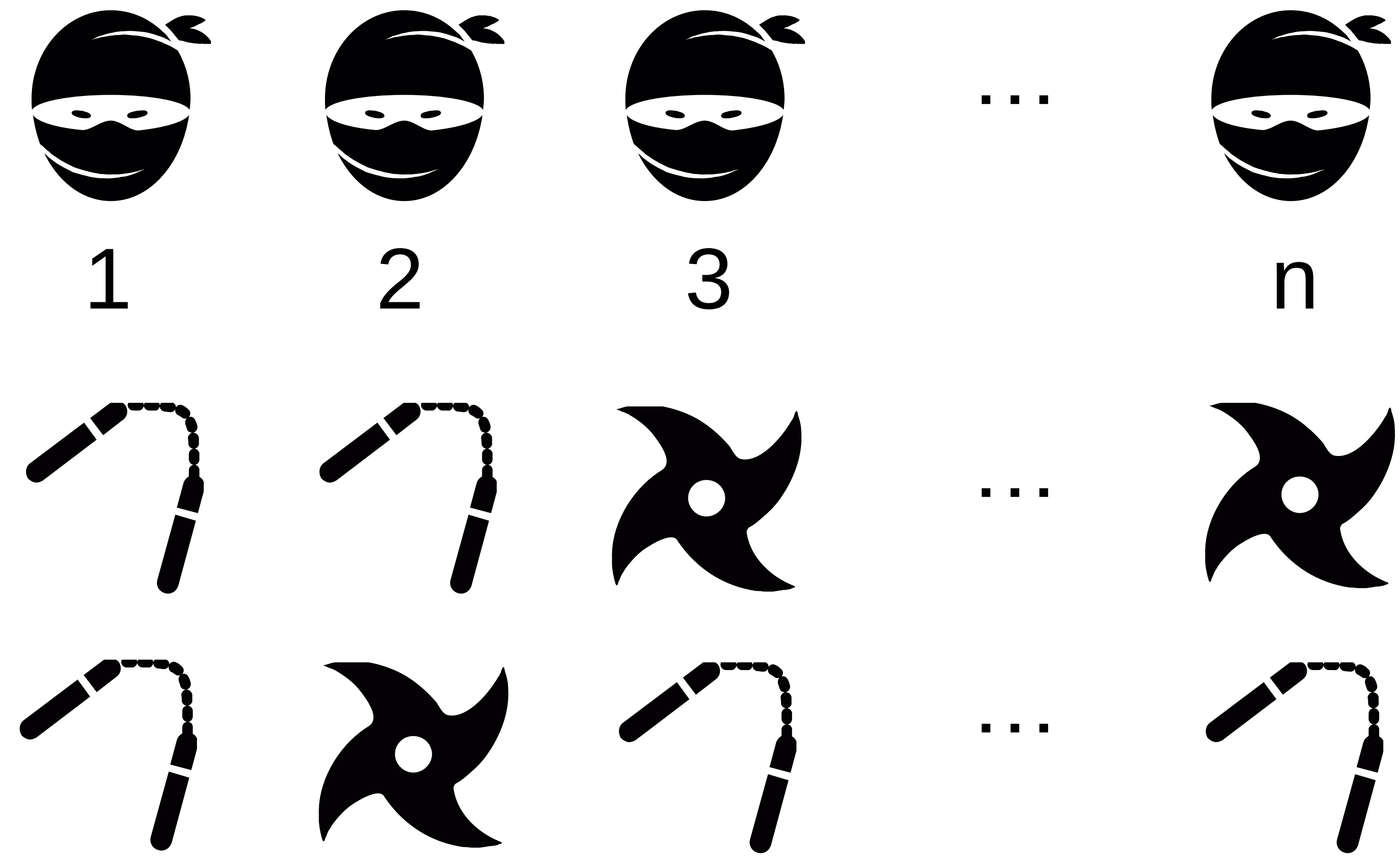
- Start with some combinatorial puzzles
- Connect these puzzles to testing

1. Come up with coverage goals
2. Show how the puzzles relate to coverage goals
3. Bound the probabilities

Puzzle I:
Ninjas Training

Ninjas in Training

In a dojo in Kaiserslautern, **n** ninjas are in training:



Round 1:

Round 2:

Training is **complete** if for every pair of ninjas, there is a round where they are in opposing teams

Ninjas in Training

Training is **complete** if for every pair of ninjas there is a round where they are in opposing teams

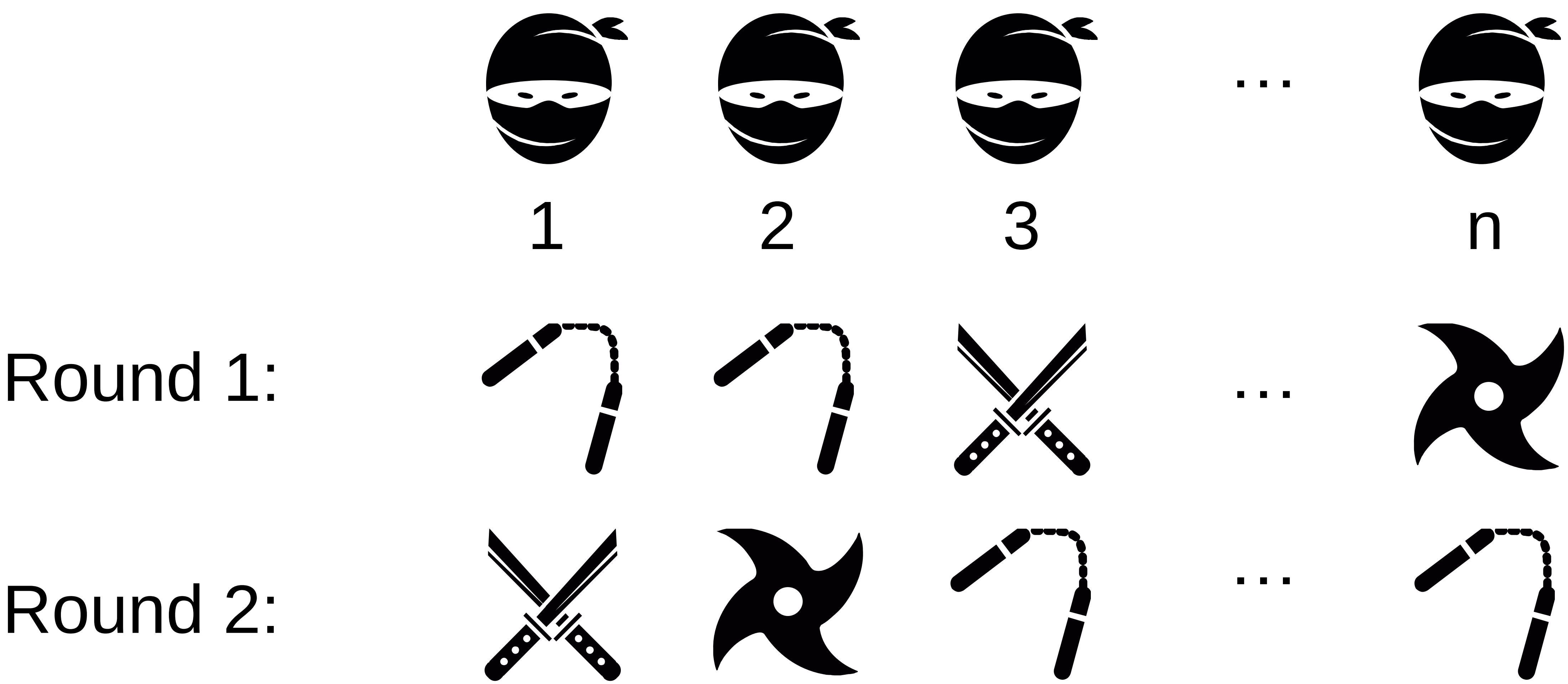
How many rounds make the training complete?

Naïve: $O(n^2)$ rounds

Can you do it in log n rounds?

Ninjas in Training

Now **n** ninjas are practicing in **k**-way fights:



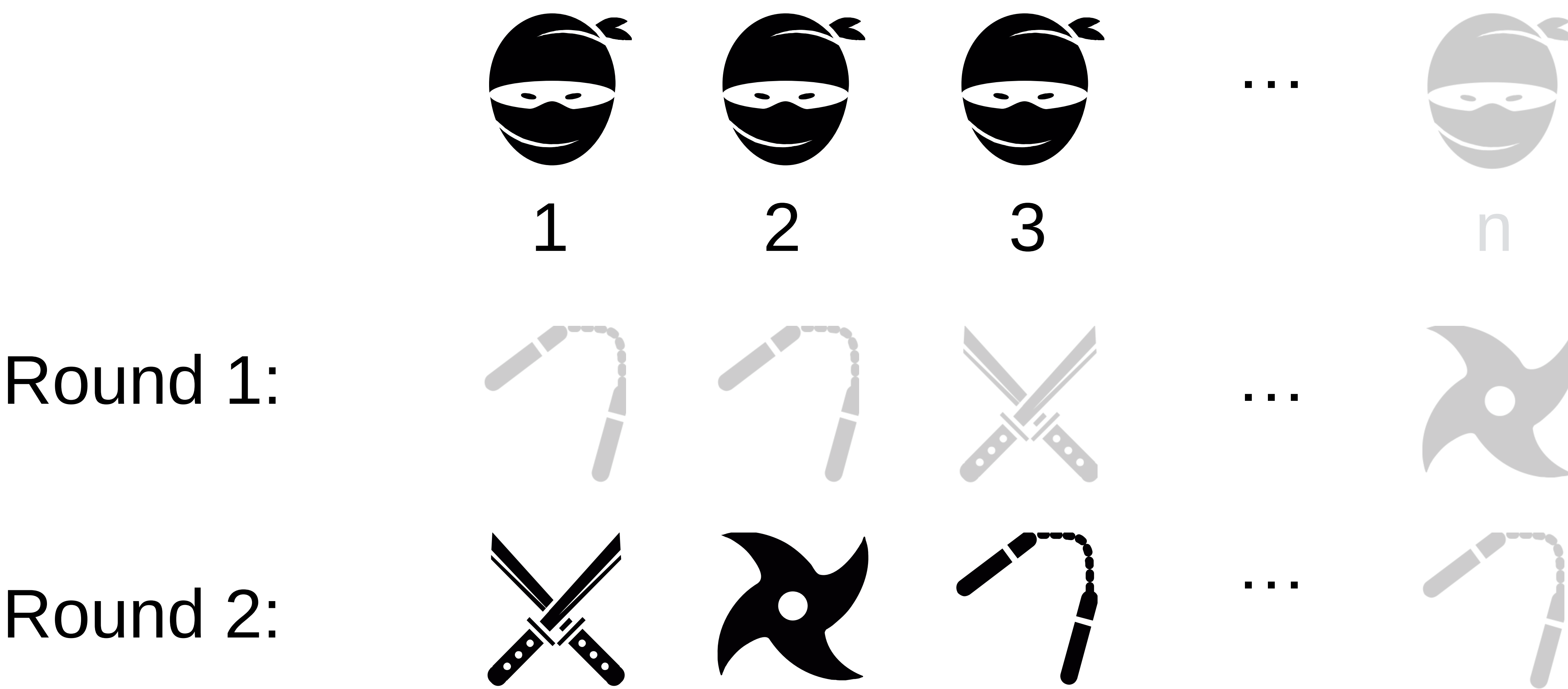
Ninjas in Training

Training is **complete** if for every choice of **k** ninjas there is a round where they are each in a different team

How many rounds make the training complete?

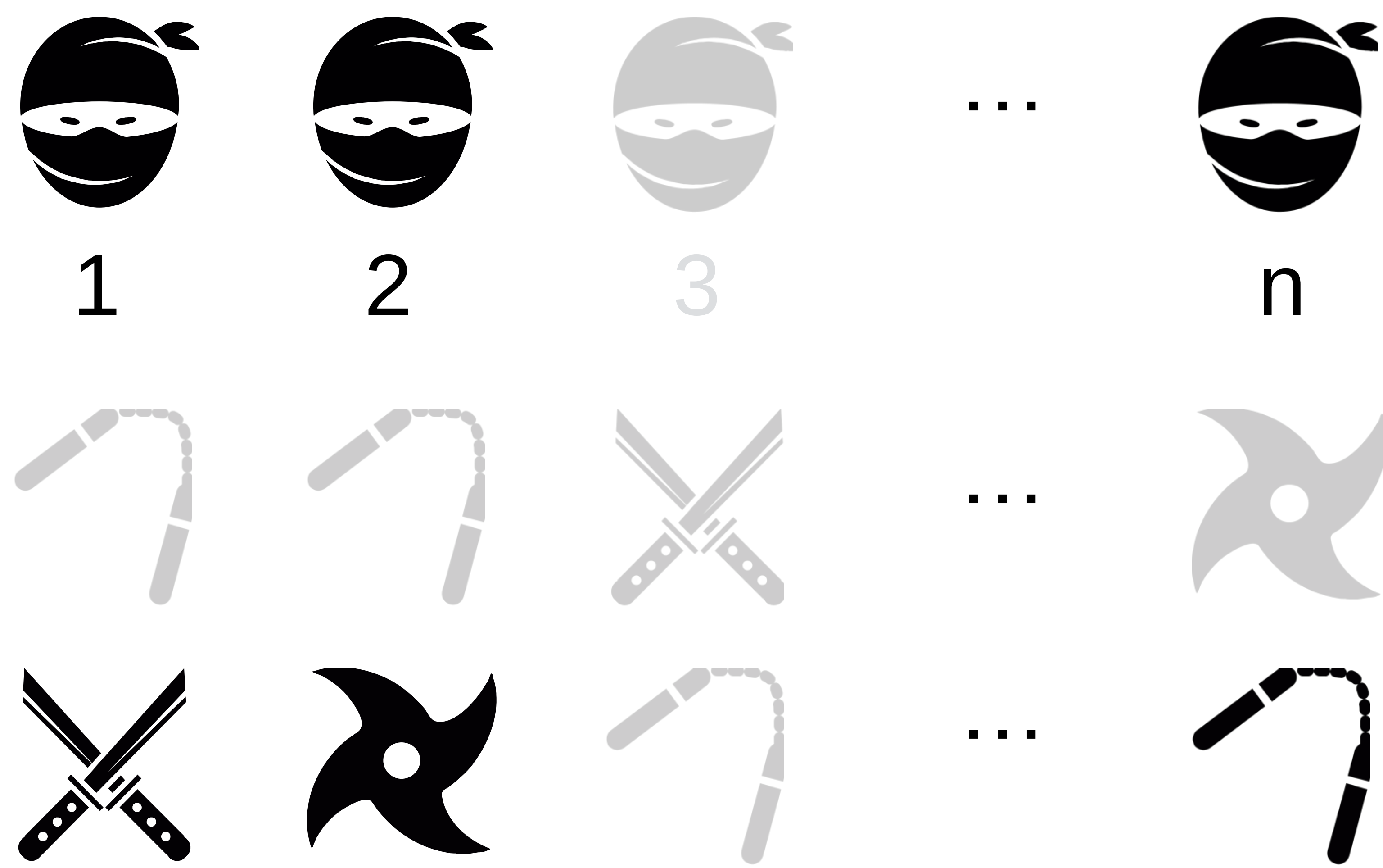
Ninjas in Training

Example:



Ninjas in Training

Example:

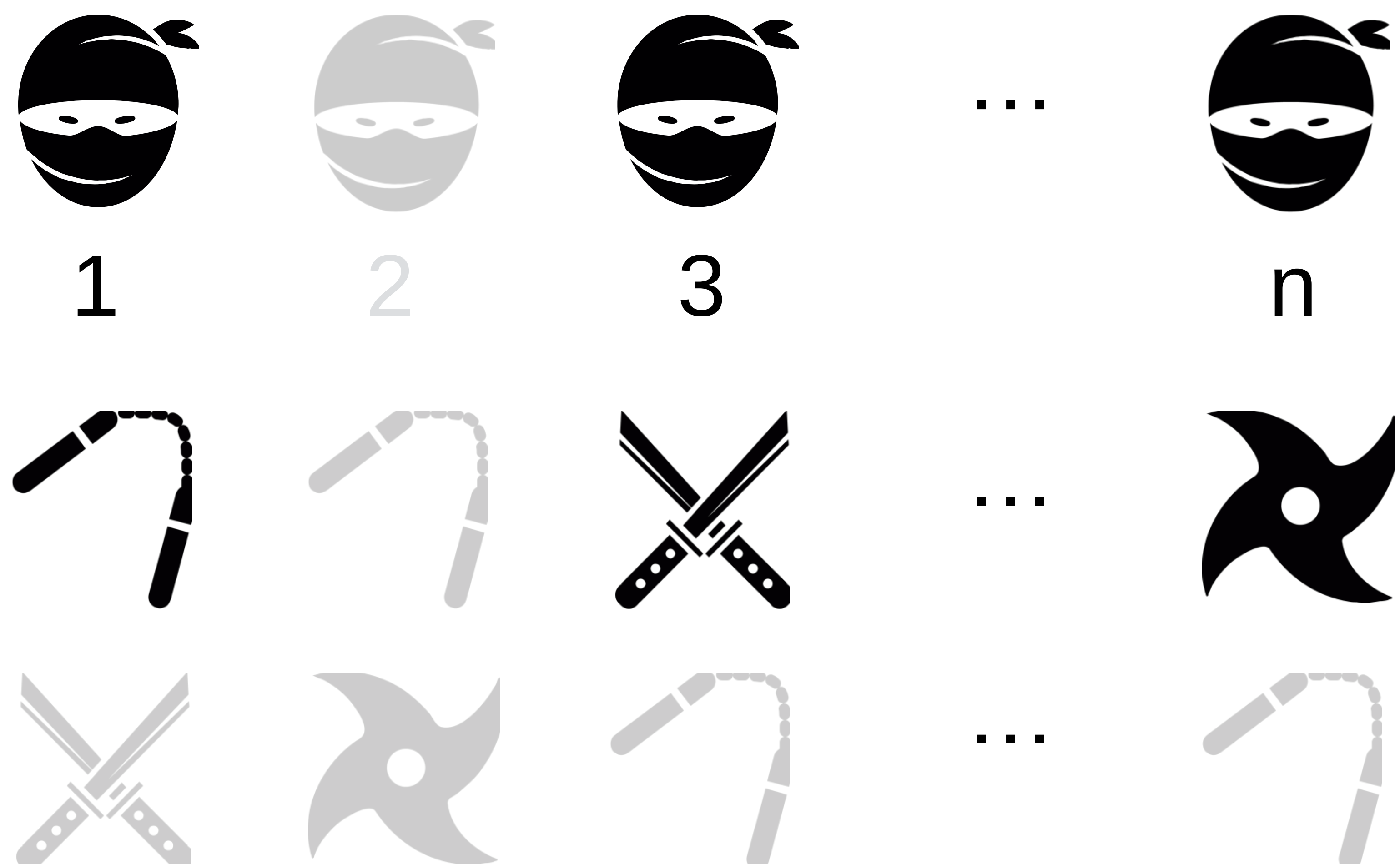


Round 1:

Round 2:

Ninjas in Training

Example:

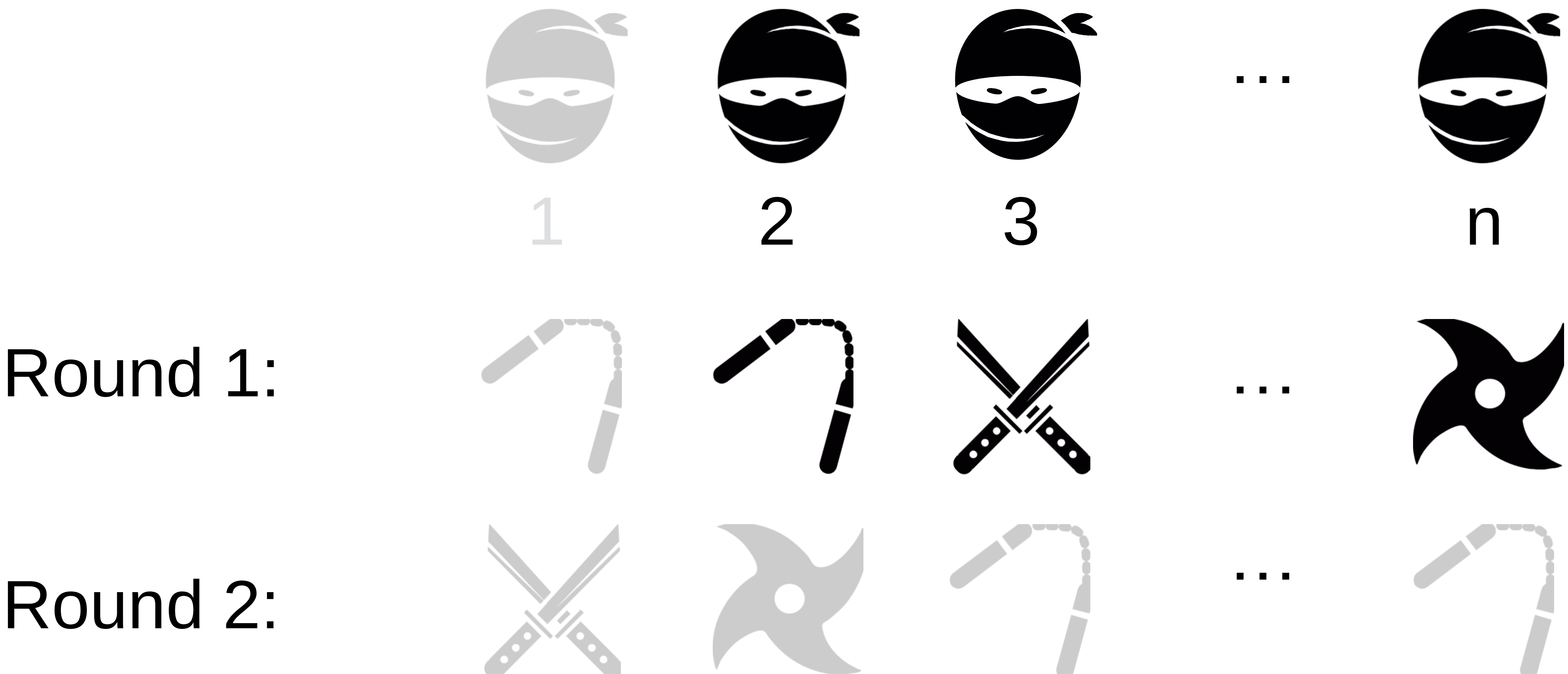


Round 1:

Round 2:

Ninjas in Training

Example:



Ninjas in Training

Training is **complete** if for every choice of **k** ninjas there is a round where they are each in a different team

How many rounds make the training complete?

Naïve: $O(n^k)$

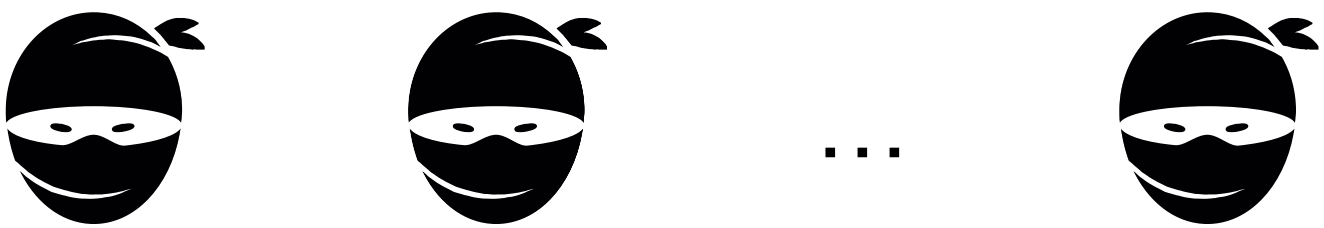
Can you do it in $O(k^{k+1} (k!)^{-1} \log n)$ rounds?

Exponentially better in n, when k is constant

Puzzle II:
Ninjas Eating

Hungry Ninjas

After training, the ninjas retire to a bucolic Biergarten…



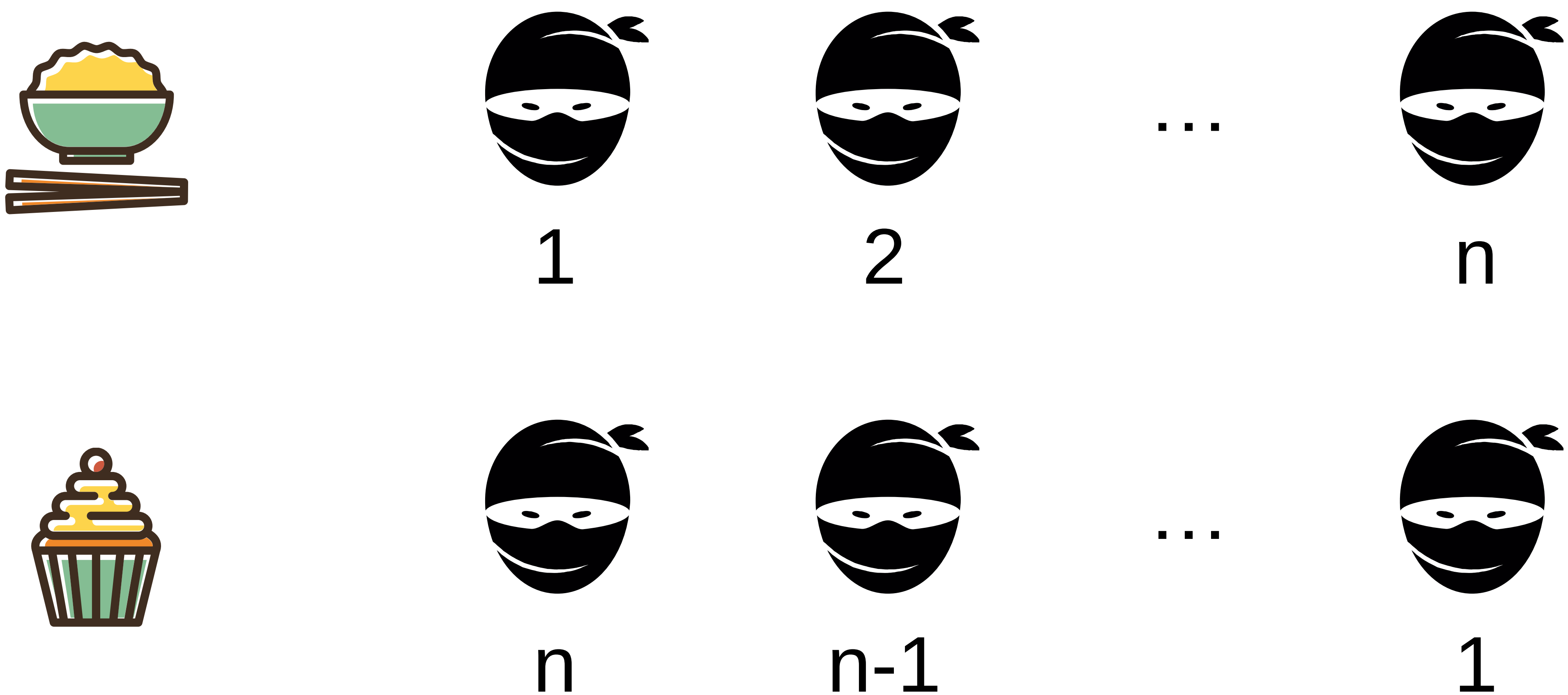
1 2 n

A banquet is **complete** if for every pair of ninjas (**i**, **j**), there's a course that is served to **i** before **j** and one that is served to **j** before **i**

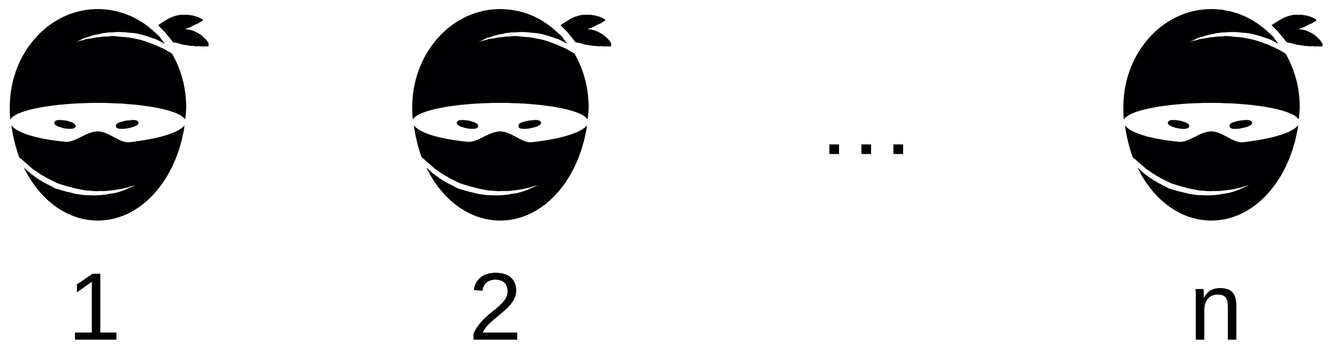
How many courses make a banquet complete?

A Complete Banquet

Two courses suffice:



3-Complete Banquets



A banquet is **3-complete** if
for every **triple** of ninjas (**i**, **j, k**),
there's a course served in the order **$i<j<k$**

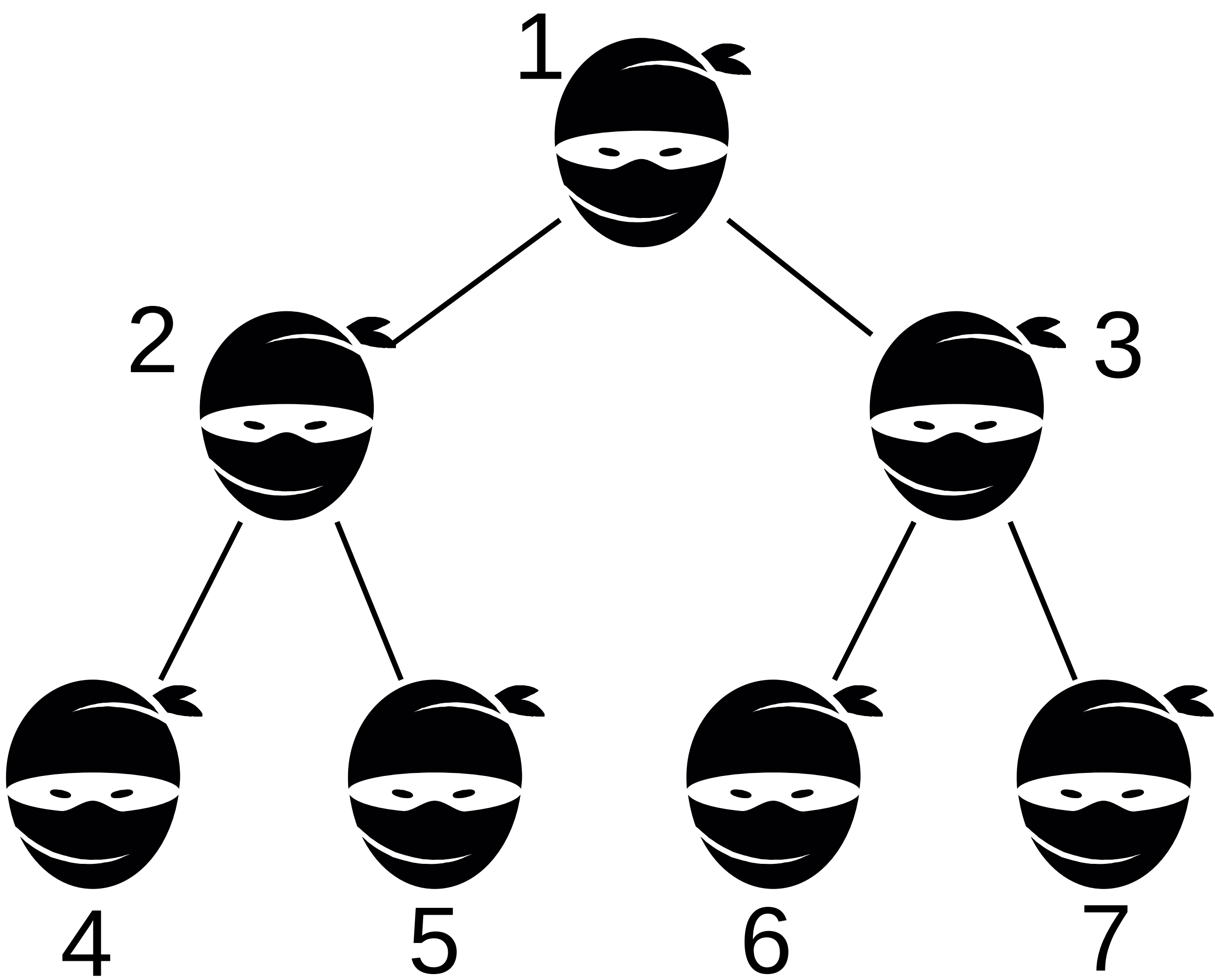
How many courses make a banquet 3-complete?

There is a 3-complete banquet with n^3 courses
Can you do it in **O(exp(d) log n)** rounds?

Masters at the Banquet

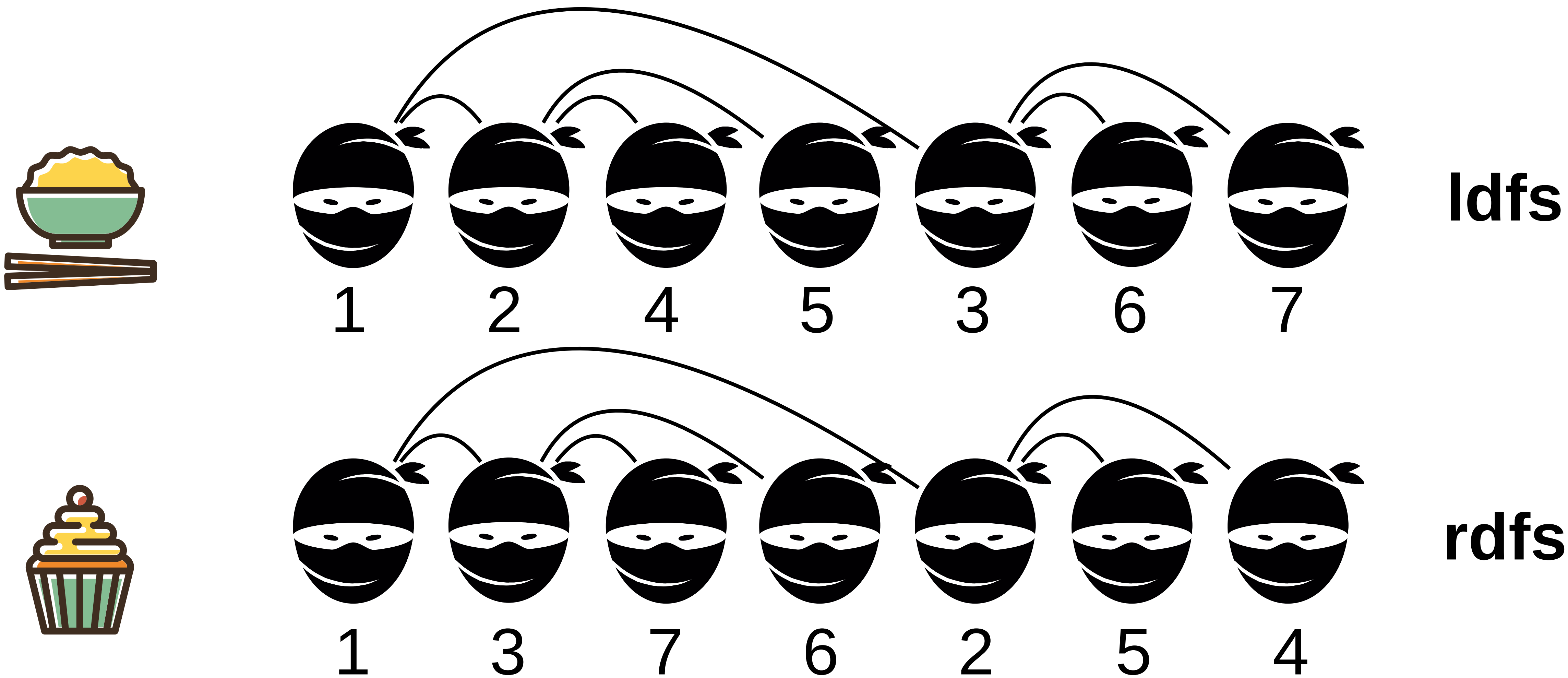
Ninjas, of course, form a hierarchy

A **master** is always served **before** their **student**



Masters at the Banquet

Again, **two** courses suffice for 2-completeness:



Ninjas at the Banquet

admissible

A banquet is **3-complete** if for every triplet (**i**, **j**, **k**), there's a course served to ninja **i** before **j**, and **j** before **k**

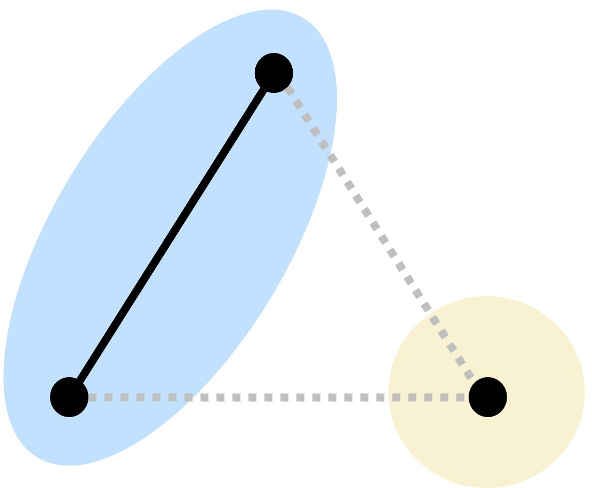
Naive approach with $\mathbf{n^3}$ courses:

Pick a course for each ${}^{n}C_{3}$. 3! orders

Can be done with **O(log n)** courses!

From Ninjas
to Testing…

From Training Ninjas to Distributed Systems with Partition Faults



ninjas
weapons
rounds
complete training

nodes in a network
blocks in a partition
tests = partitions
splitting family of partitions

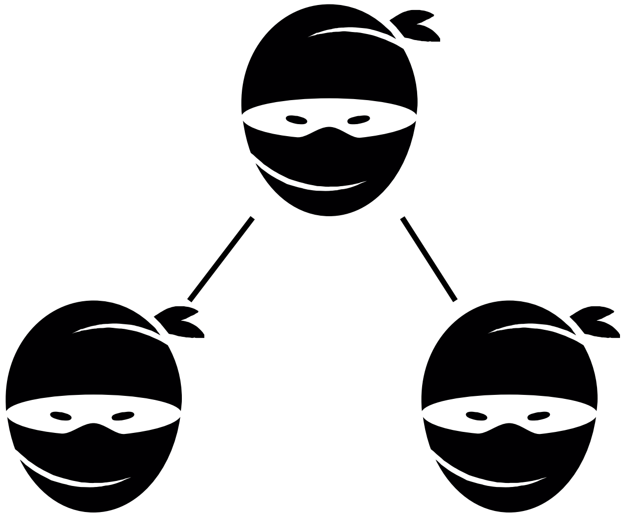
Coverage Goal: Splitting Families

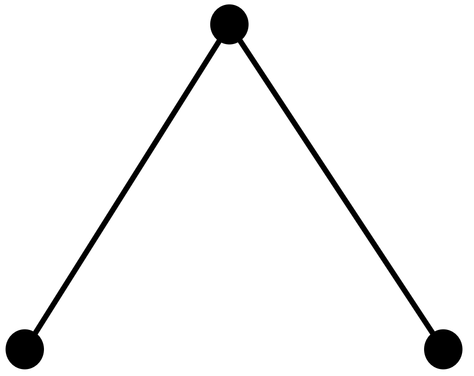
Given n nodes and $k \le n$,

a partition of nodes $P = \{B_1, \ldots, B_k\}$ **splits** a set of nodes $S = \{x_1, \ldots, x_k\}$ if $x_1 \in B_1, \ldots, x_k \in B_k$.

A set of partitions F is a **k-splitting family** if for every k-subset of nodes there is a partition in F that splits it.

From Eating Ninjas to Testing Concurrent Systems





ninjas	events
hierarchy	partial order on events
courses	tests = schedules/linearization
d-complete banquet	**d-hitting family of schedules**

Coverage Goal: Hitting Families

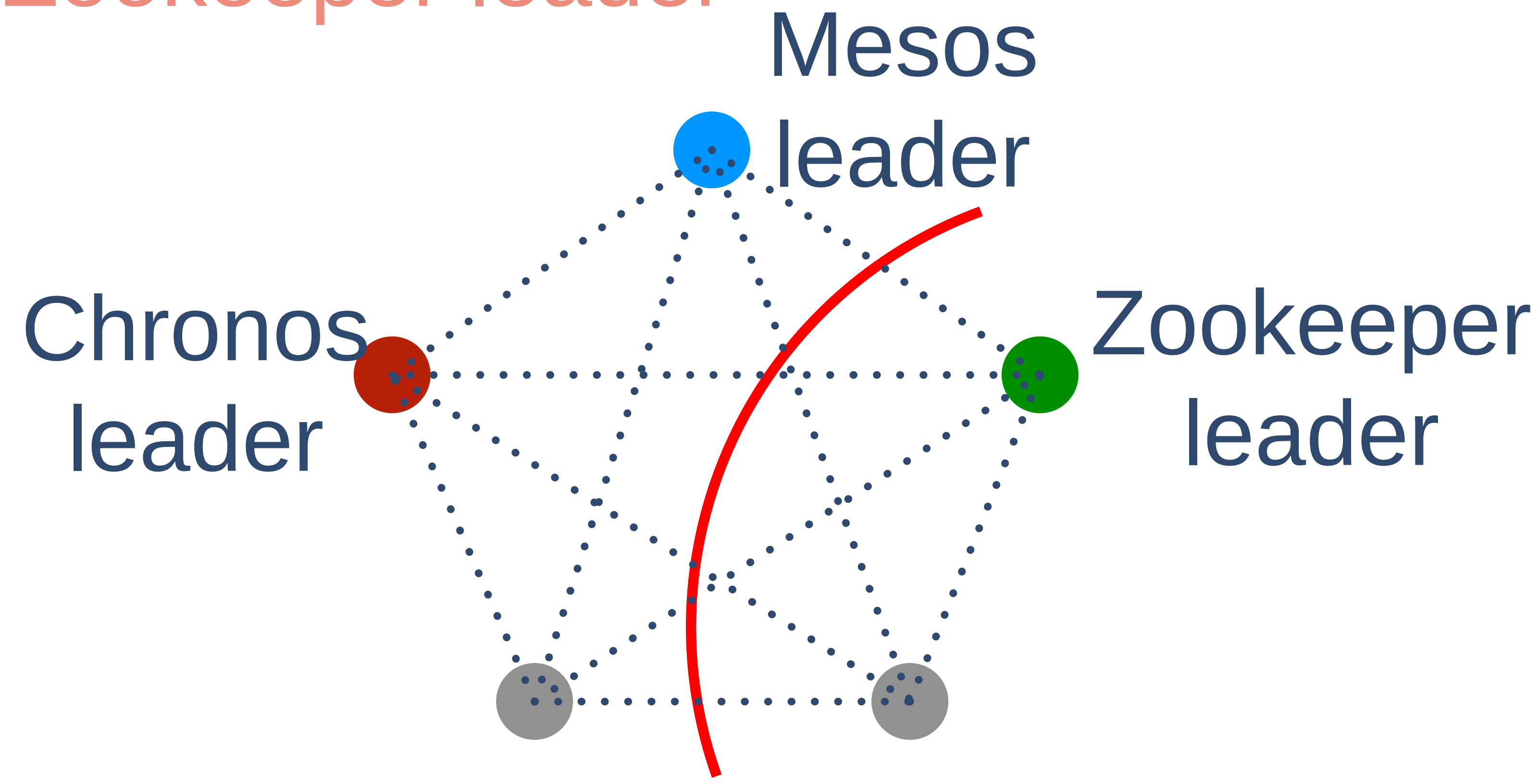
Given a poset of events, a schedule **hits** a d-tuple of events $\mathbf{(e_1,\ldots,e_d)}$ if it executes the events in the order $\mathbf{e_1<\ldots<e_d}$.

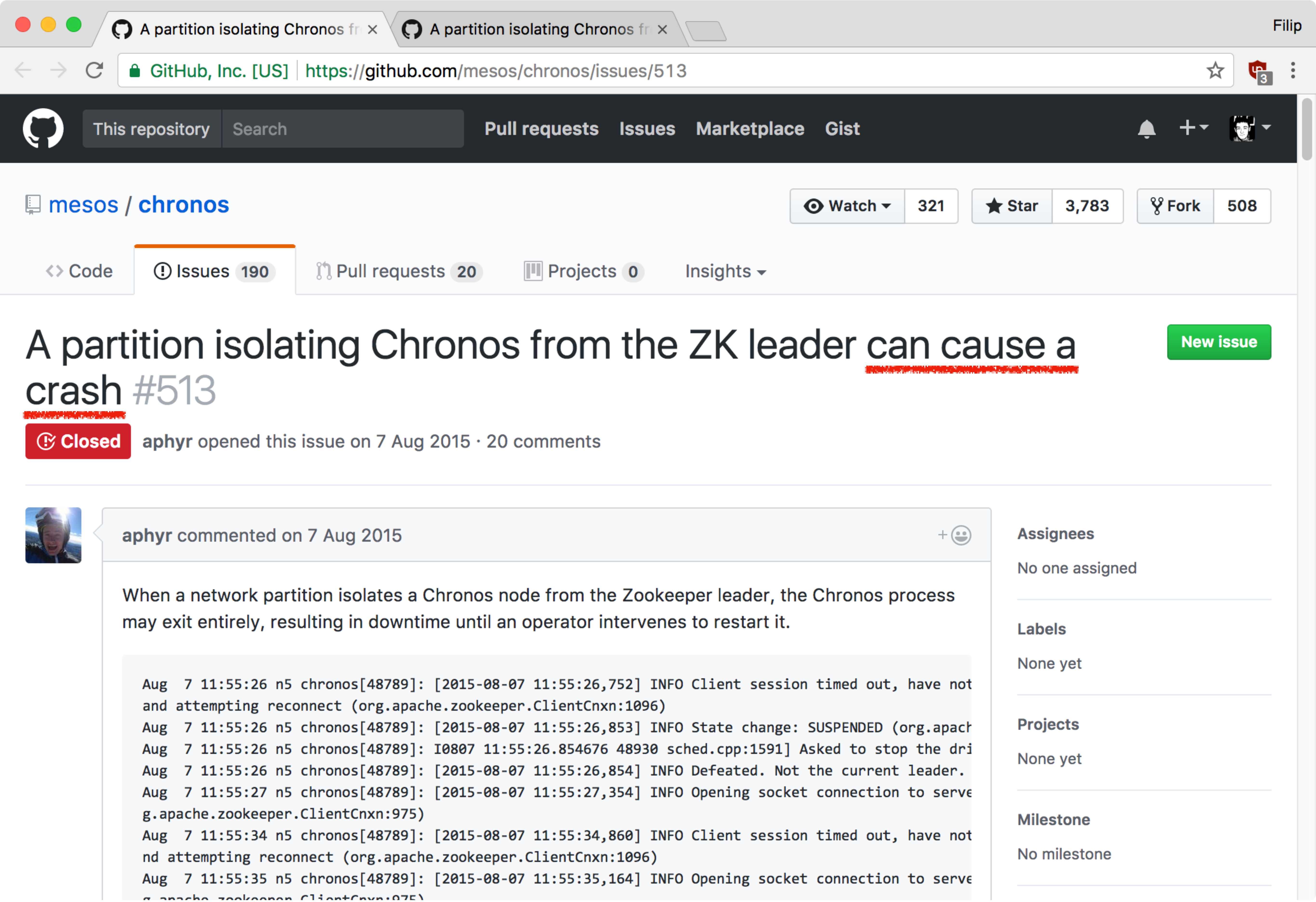
Given a poset of events, a **family of schedules F** is **d-hitting** if for every admissible d-tuple of events there is a schedule in **F** that hits it.

From Jepsen
https://jepsen.io

Why k-Splitting?

- Chronos: A distributed fault-tolerant job scheduler
- Works in conjunction with Mesos and Zookeeper
- Three special nodes: Chronos leader, Mesos leader, Zookeeper leader





A partition isolating Chronos fr × A partition isolating Chronos fr ×

Filip

GitHub, Inc. [US] | https://github.com/mesos/chronos/issues/513

This repository | Search | Pull requests | Issues | Marketplace | Gist

mesos / chronos

Watch 321 | Star 3,783 | Fork 508

Code | Issues 190 | Pull requests 20 | Projects 0 | Insights

A partition isolating Chronos from the ZK leader can cause a crash #513

New issue

Closed **aphyr** opened this issue on 7 Aug 2015 · 20 comments

aphyr commented on 7 Aug 2015

When a network partition isolates a Chronos node from the Zookeeper leader, the Chronos process may exit entirely, resulting in downtime until an operator intervenes to restart it.

```
Aug  7 11:55:26 n5 chronos[48789]: [2015-08-07 11:55:26,752] INFO Client session timed out, have not
and attempting reconnect (org.apache.zookeeper.ClientCnxn:1096)
Aug  7 11:55:26 n5 chronos[48789]: [2015-08-07 11:55:26,853] INFO State change: SUSPENDED (org.apach
Aug  7 11:55:26 n5 chronos[48789]: I0807 11:55:26.854676 48930 sched.cpp:1591] Asked to stop the dri
Aug  7 11:55:26 n5 chronos[48789]: [2015-08-07 11:55:26,854] INFO Defeated. Not the current leader.
Aug  7 11:55:27 n5 chronos[48789]: [2015-08-07 11:55:27,354] INFO Opening socket connection to serve
g.apache.zookeeper.ClientCnxn:975)
Aug  7 11:55:34 n5 chronos[48789]: [2015-08-07 11:55:34,860] INFO Client session timed out, have not
nd attempting reconnect (org.apache.zookeeper.ClientCnxn:1096)
Aug  7 11:55:35 n5 chronos[48789]: [2015-08-07 11:55:35,164] INFO Opening socket connection to serve
```

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

A partition isolating Chronos fr × A partition isolating Chronos fr × Filip

GitHub, Inc. [US] https://github.com/mesos/chronos/issues/522

This repository Search Pull requests Issues Marketplace Gist

mesos / chronos

Watch 321 Star 3,783 Fork 508

Code Issues 190 Pull requests 20 Projects 0 Insights

A partition isolating Chronos from the ZK leader can *not* cause a crash #522

New issue

Open aphyr opened this issue on 14 Aug 2015 · 7 comments

aphyr commented on 14 Aug 2015

Per #513, Chronos is expected to crash when a leader loses its Zookeeper connection. In this test case, Chronos detects the loss of its Zookeeper connection and, instead of crashing, sleeps quietly and reconnects when the partition heals. #513 argues that to keep running would violate unspecified correctness constraints. To preserve safety, should Chronos *also* crash here?

air commented on 15 Aug 2015

Hi - you're referring to a statement that doesn't represent the design (it wasn't expressed carefully enough). Please disregard it and refer to the clarification in the thread. Make sense?

Assignees

No one assigned

Labels

bug

Projects

None yet

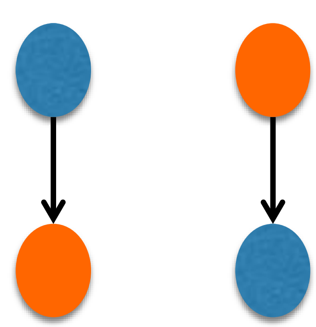
Milestone

No milestone

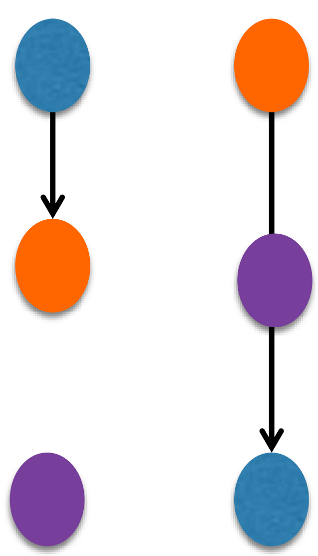
Why d-Hitting?

Many bugs in asynchronous programs involve small number of events —**bug depth d**

[Lu et al. ASPLOS ’08] [Burckhardt et al. ASPLOS ’10] [Jensen et al. OOPSLA ’15] [Petrov et al. 2012]



d = 2: order violation



d = 3: atomicity violation

Combinatorics: Bounding the Probabilities

Plan

- Start with some combinatorial puzzles
- Connect these puzzles to testing

1. Come up with coverage goals
2. Show how the puzzles relate to coverage goals
3. Bound the probabilities

Small k-Splitting Families

- Fix a k-element set S
- What is the probability a random partition splits S?
- A splitting partition uniquely corresponds to a map $U \setminus S \to S$. There are $k^{(n-k)}$ such maps
- So probability = $k^{(n-k)} / \{n \text{ P } k\}$

Stirling number of 2nd kind
=
Number of partitions of n elements into k parts

Small k-Splitting Families

- Probability that a fixed set is split = $k^{(n-k)} / \{n\ P\ k\}$
- Can we get rid of n?
- Yes: $k^n \geq k!\ \{n\ P\ k\}$

All functions from n to k

All surjections from n to k

$\geq k^{-k} . k!$

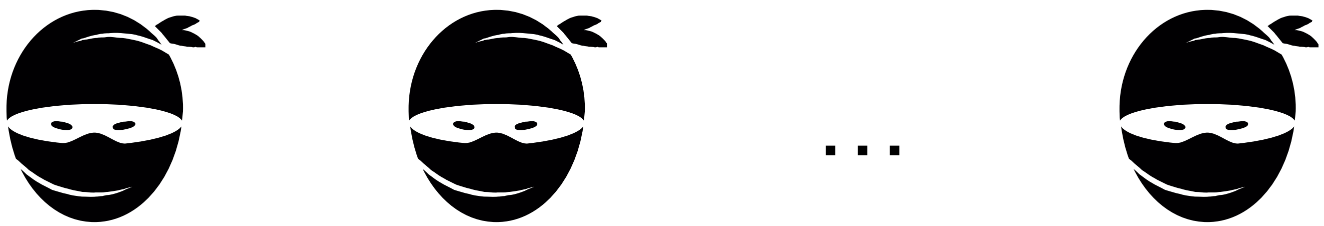
Small k-Splitting Families

- Pr[Random partition splits S] ≥ $k!/k^k$
- From our general theorem: *There is a k-splitting family of size $(k^k/k!)\, k \log n$*
- Turns out: uniformly sampling k-partitions is hard
  - But sampling balanced partitions is sufficient
  - The combinatorial arguments are harder

$$k^n \binom{n}{k} \le n^k \left\{ {n \atop k} \right\}$$

Problem 11957 AMM 2017

Small d-Hitting Families



1 2 n

Probabilistic argument: Fix a tuple of d ninjas

What is the probability that a random schedule covers it?

$1/d!$

What is the probability k schedules don’t cover it?

$(1 - 1/d!)^k$

What is the probability k schedules are not a hitting family?

At most $(n\ C\ 3) \cdot (1 - 1/d!)^k$

Pick $k > d!\ d \log n$

Plan

- Start with some combinatorial puzzles
- Connect these puzzles to testing

1. Come up with coverage goals
2. Show how the puzzles relate to coverage goals
3. Bound the probabilities

Combinatorics I: Splitting Families & Perfect Hashing

- Andrew Yao. *Should Tables Be Sorted?* 1981

- Uses k-splitting families to construct **“perfect hash functions”**: for every k-subset S of an n-element domain there is a hash function that is 1-to-1 on S

- Uses perfect hash functions to construct **hash tables with constant lookup**

- Follow-up work by Fredman, Komlós, Szemerédi

Combinatorics II: Hitting Families & Order Dimension of Posets

- Dushnik and Miller. *Partially Ordered Sets.* 1941
- Define and study **order dimension of a poset:** Number of linearizations whose intersection is the poset
- Hitting families generalize order dimension
  order dimension = size of the smallest 2-hitting family
- d-completeness: Not studied in the p.o. literature!

Explicit Constructions

Probabilistic method shows existence:

- k-splitting families of size: $\mathbf{(k^{k+1} / k!) \log n}$
- d-hitting families of size: $\mathbf{d\, d! \log n}$

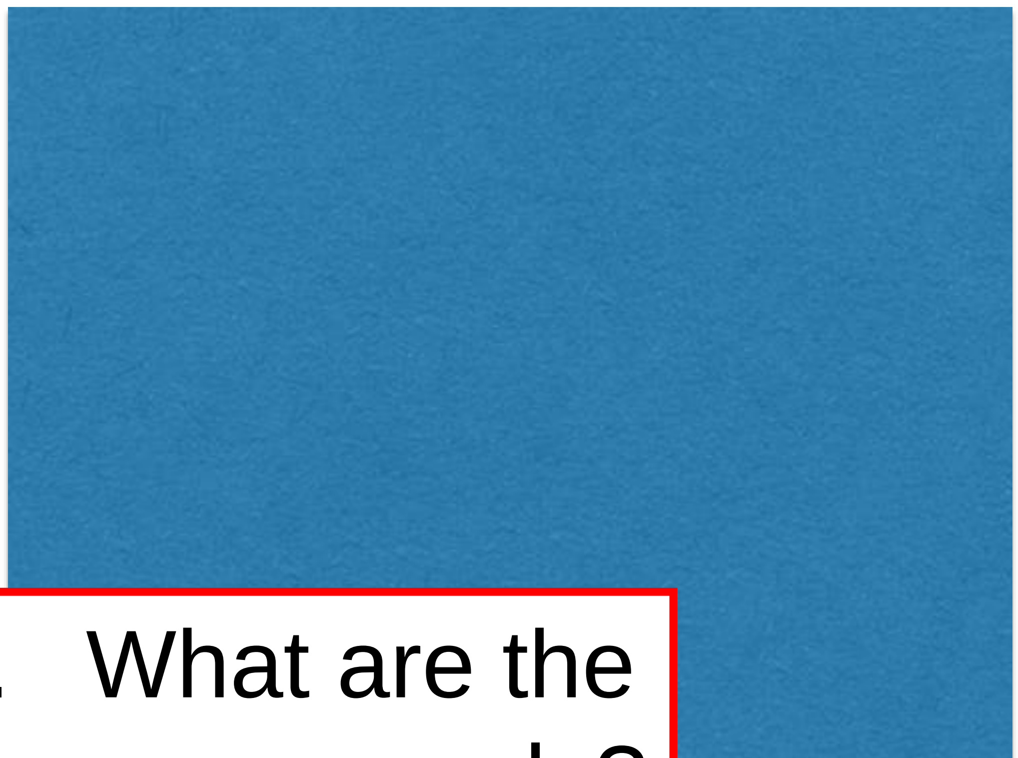
But may not be optimal:

- 2-splitting family of size log n exists; prob. method gives 2 log n
- d-hitting families for trees of size h of size $\mathbf{O(\exp(d) \cdot h^{d-1})}$

OTOH, matching explicit constructions are open:

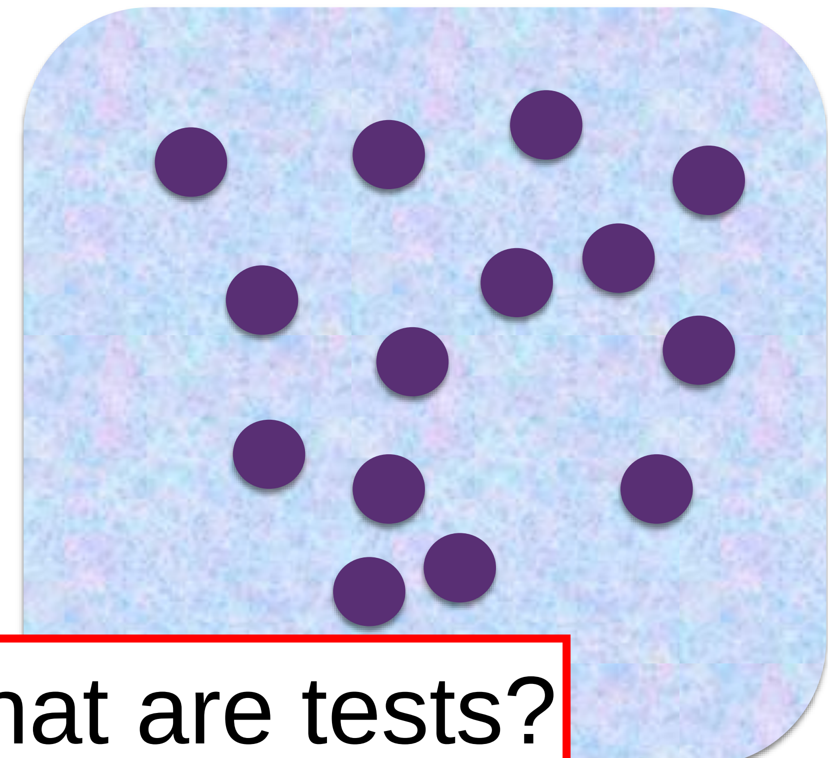
- k-splitting families of size: $\mathbf{4^{sqr(k)} (\log_2 n)^{k-1}}$ by Yao

Summary



1. What are the coverage goals?

Coverage Goals

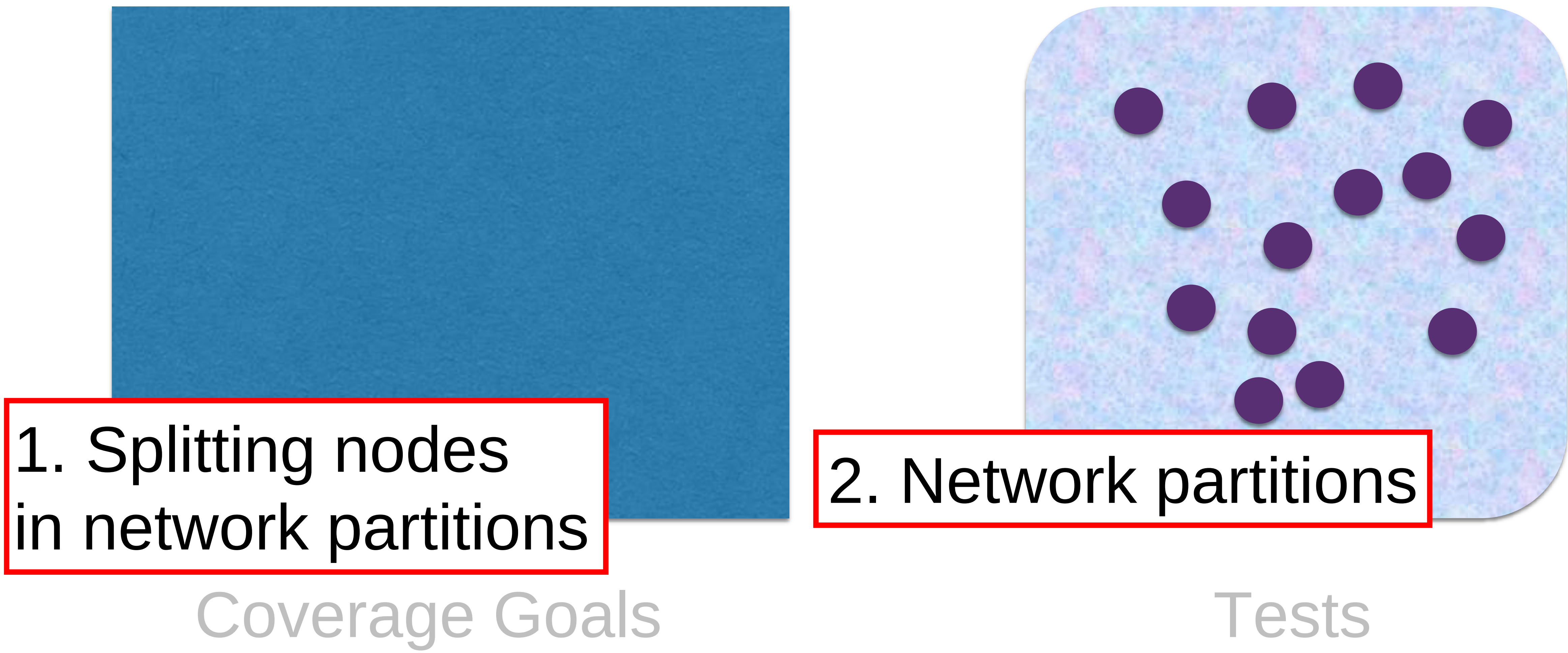


2. What are tests?

Tests

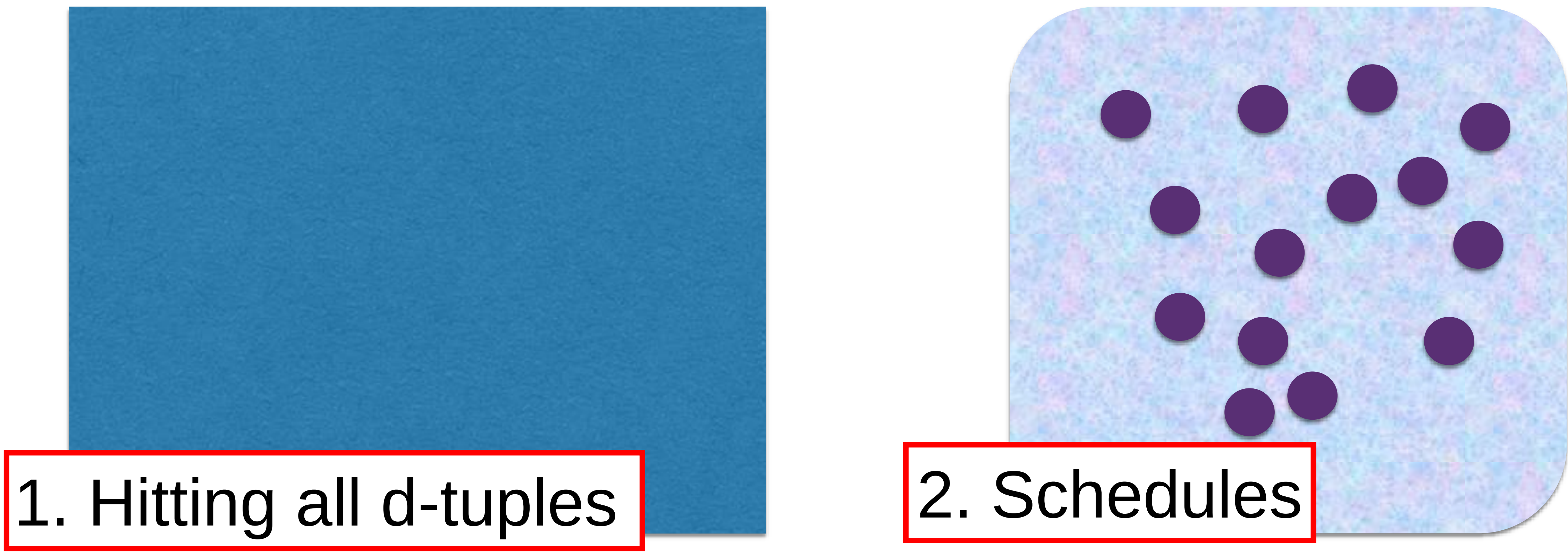
3. Can we bound Pr[Test covers a goal]?
$\Rightarrow$ If so, “small” test sets exist

Summary



3. Pr[Test covers a goal] ≥ exp(k) log n

Summary

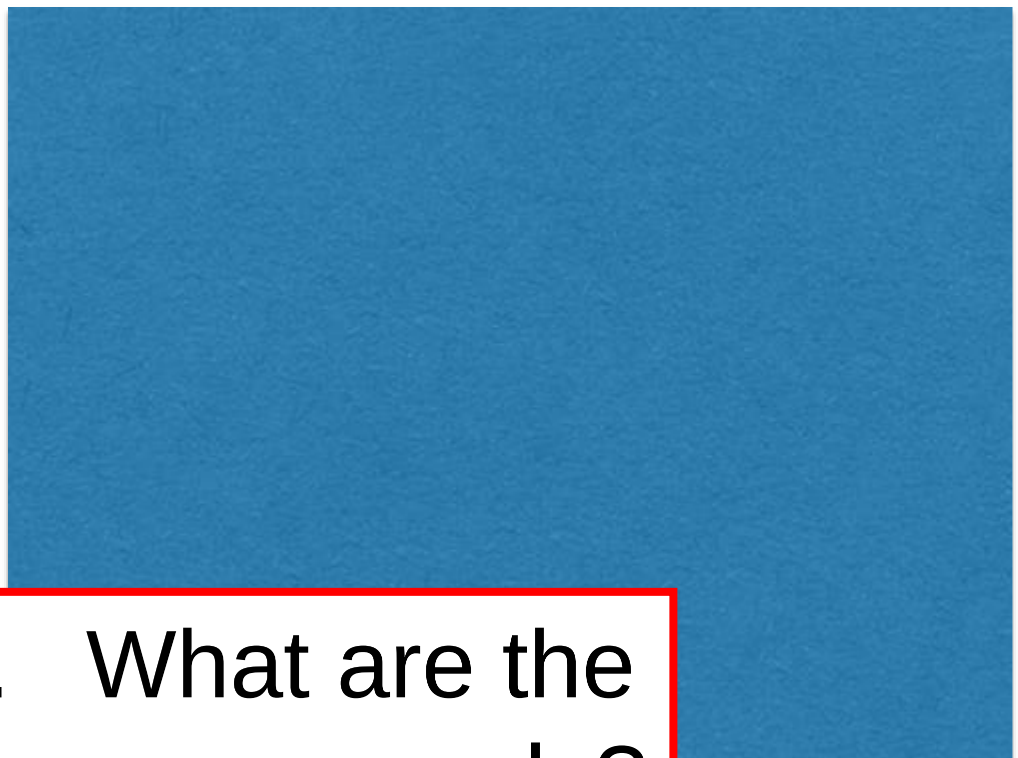


Coverage Goals

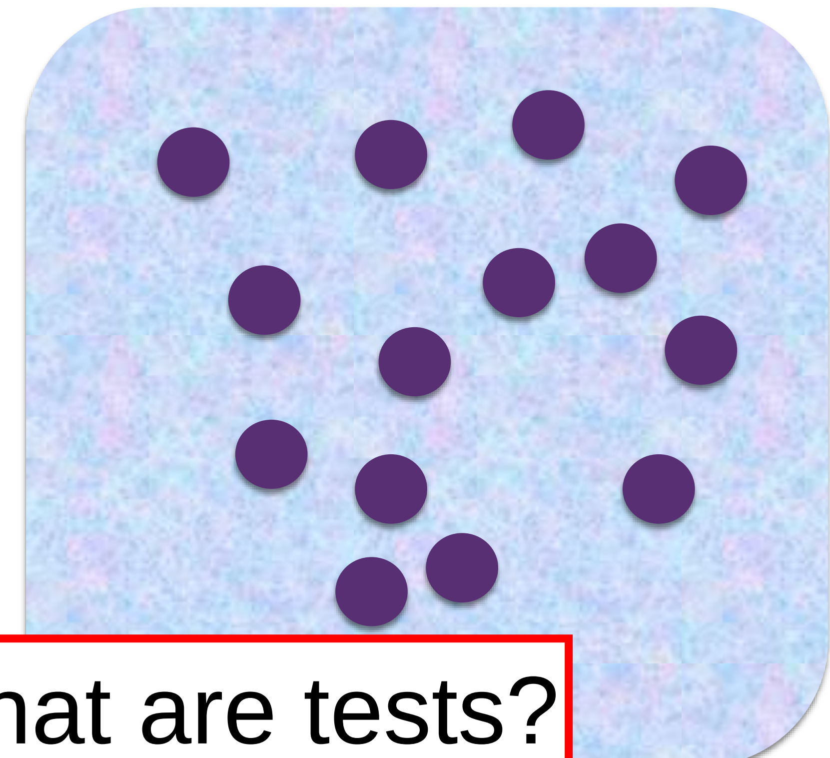
Tests

3. Pr[Test covers a goal] ≥ exp(d) log n

Summary



1. What are the coverage goals?



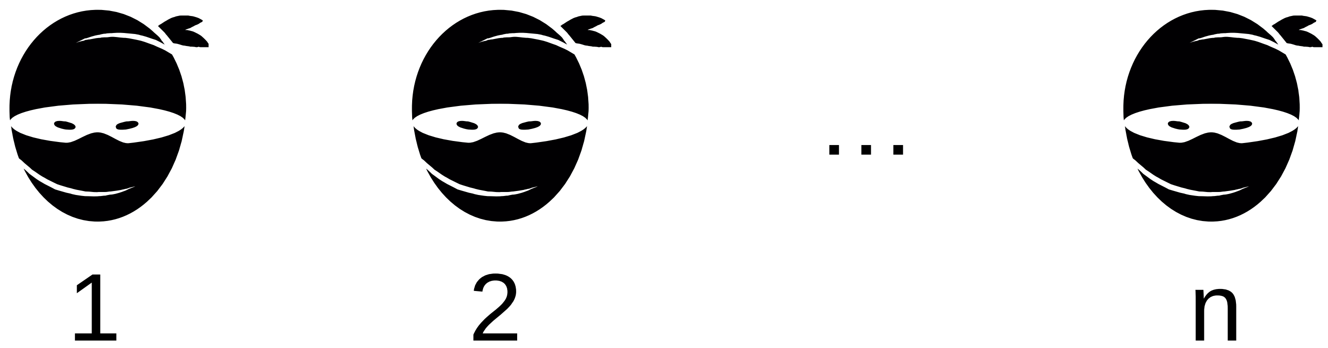
2. What are tests?

Coverage Goals

Tests

3. Can we bound Pr[Test covers a goal]?
⇒ If so, "small" test sets exist

Thank You



Watch out for Filip Niksic's PhD Thesis!

Icons made by Freepik at www.flaticon.com