Why Is Random Testing Effective for Partition Tolerance Bugs?

Rupak Majumdar, Filip Niksic

Max Planck Institute for Software Systems
(MPI-SWS)

Despite Many Formal Approaches…

Despite Many Formal Approaches…

…practitioners **test** their code

Despite Many Formal Approaches…

…practitioners **test** their code

…by providing **random** inputs.

Despite Many Formal Approaches…

…practitioners **test** their code

…by providing **random** inputs.

And despite our best judgement,

Despite Many Formal Approaches…

…practitioners **test** their code

…by providing **random** inputs.

And despite our best judgement,

…testing is **surprisingly effective in finding bugs**.

Despite Many Formal Approaches…

…practitioners **test** their code

…by providing **random** inputs.

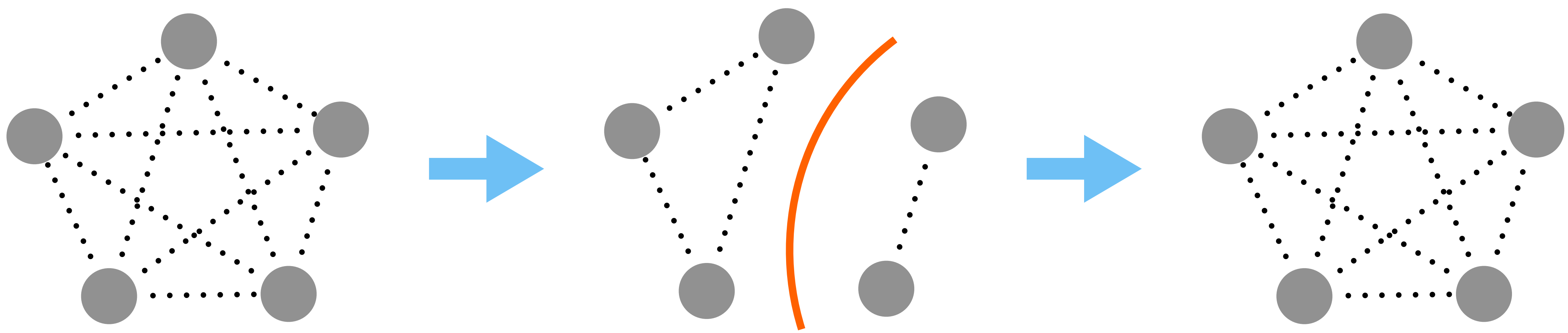
And despite our best judgement,

…testing is **surprisingly effective in finding bugs**.

We explore this unexpected effectiveness
in testing distributed systems under partition faults.

Jepsen: Call Me Maybe

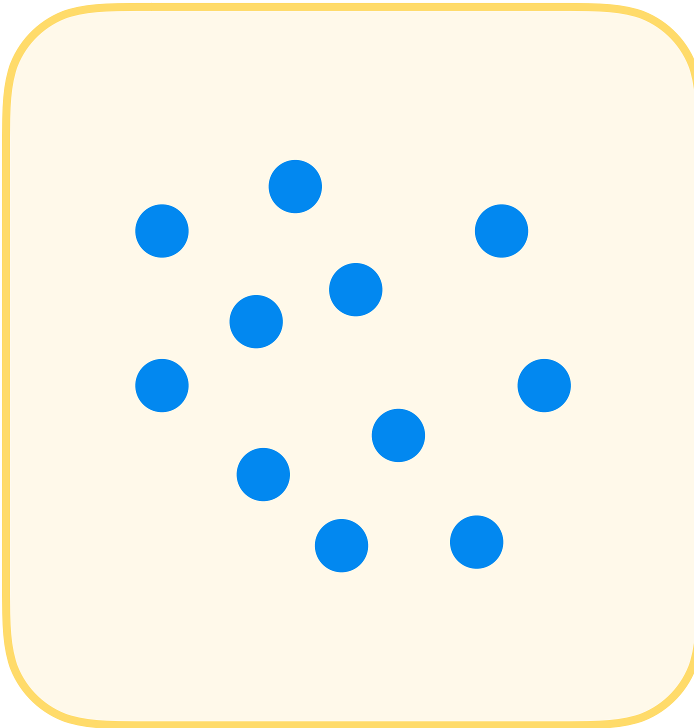
A framework for black-box testing of distributed systems by randomly inserting network partition faults

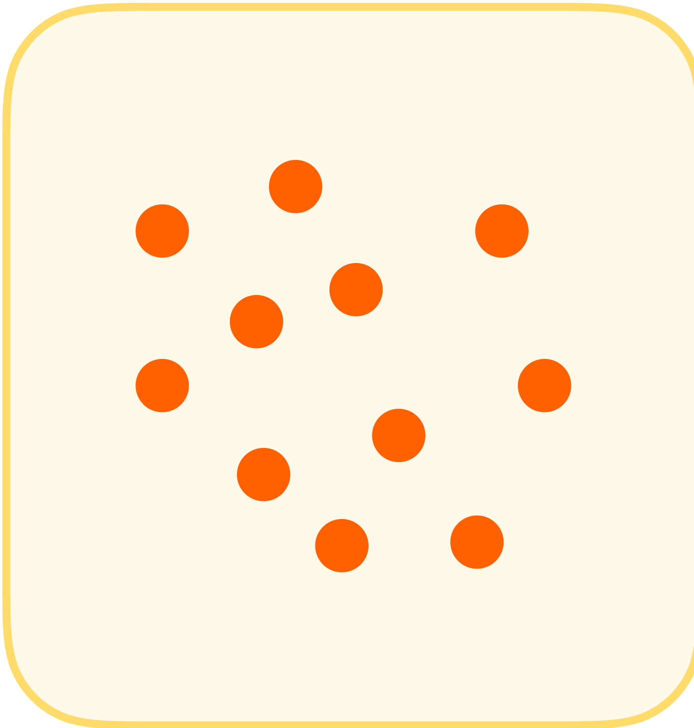


Analyses on http://jepsen.io/: etcd, Postgres, Redis, Riak, MongoDB, Cassandra, Kafka, RabbitMQ, Consul, Elasticsearch, Aerospike, Zookeeper, Chronos…

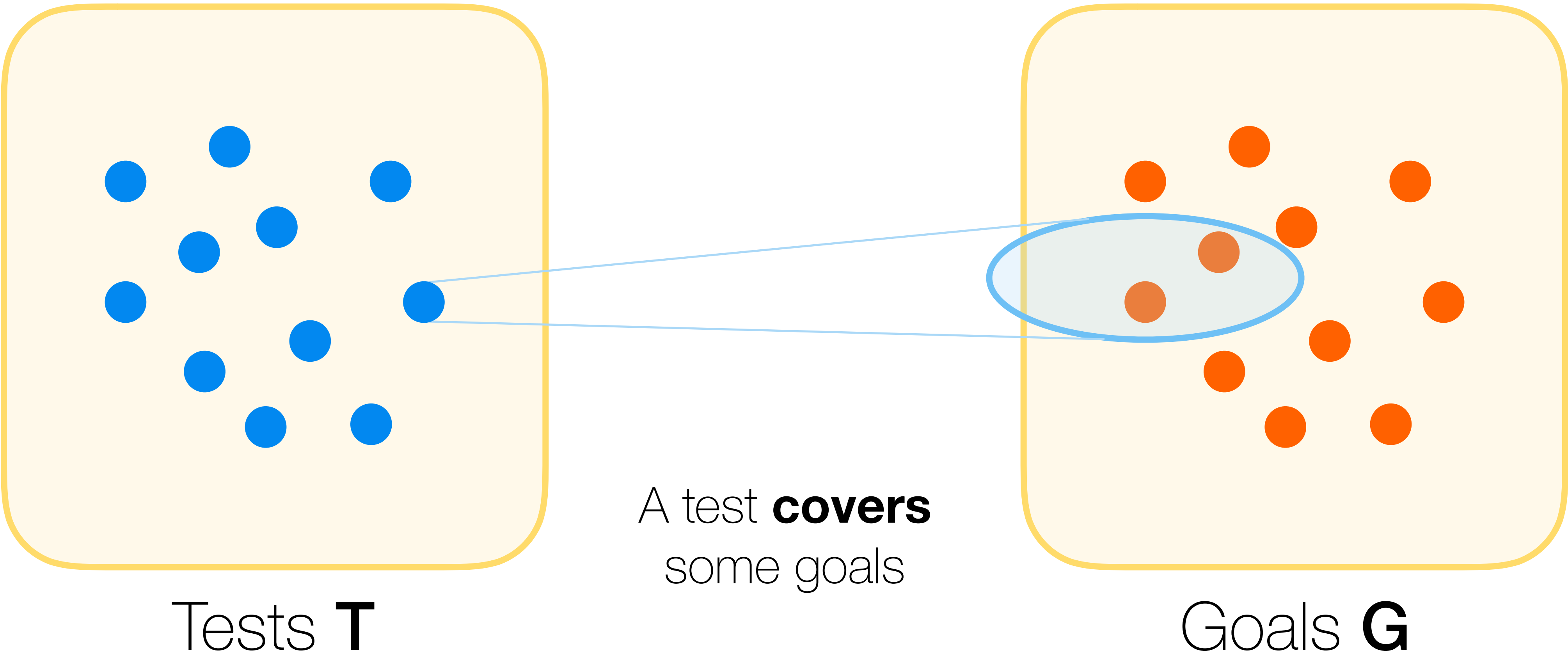
1. General Random Testing Framework
2. Randomly Testing Distributed Systems
3. Wider Context: Combinatorial Testing

Tests and Goal Coverage

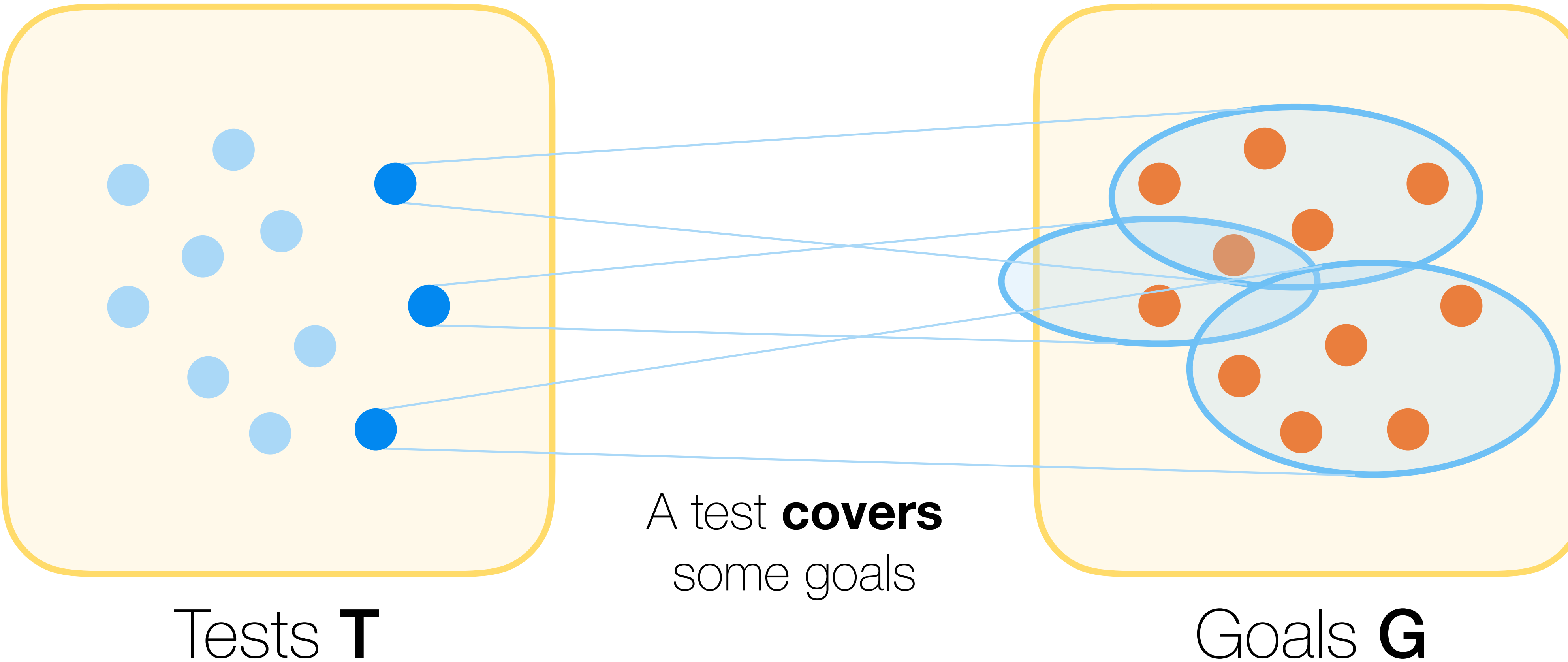




Tests and Goal Coverage

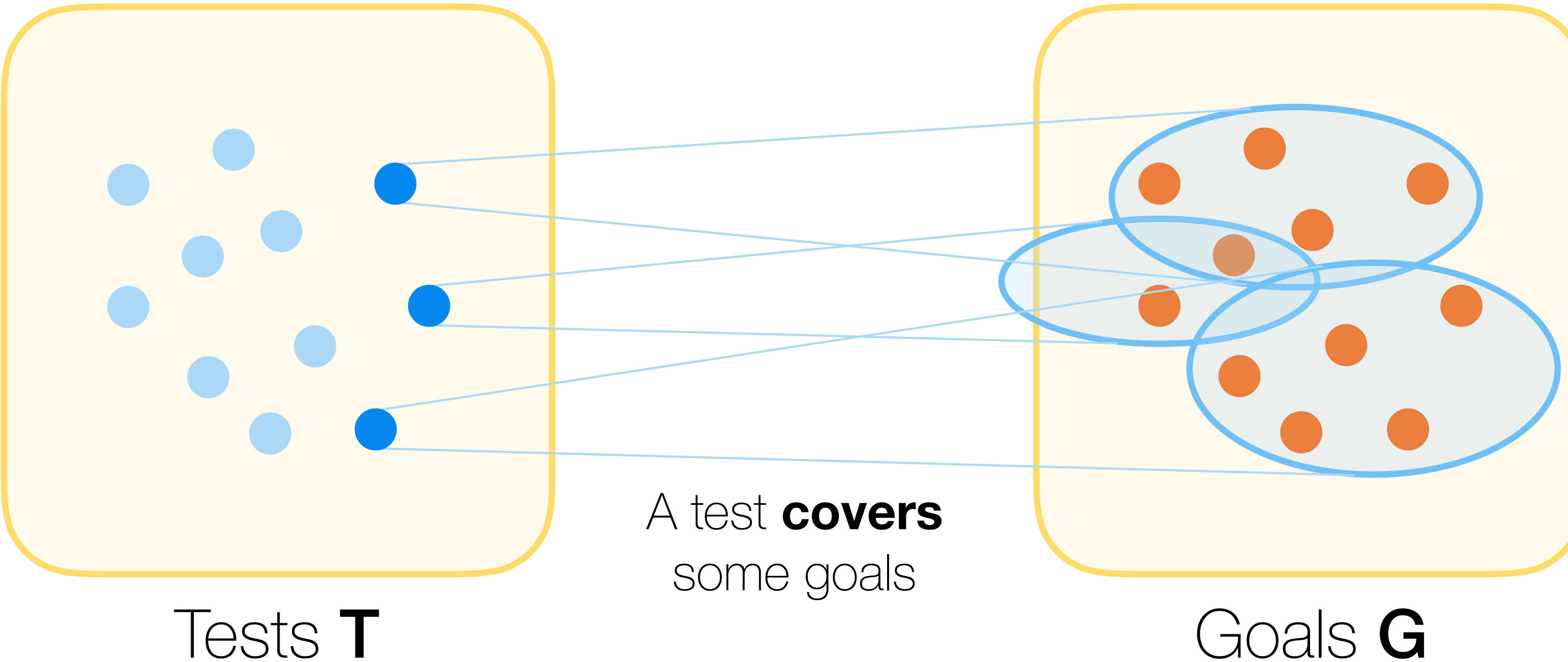


Tests and Goal Coverage



Covering family = Set of tests that **cover all goals**

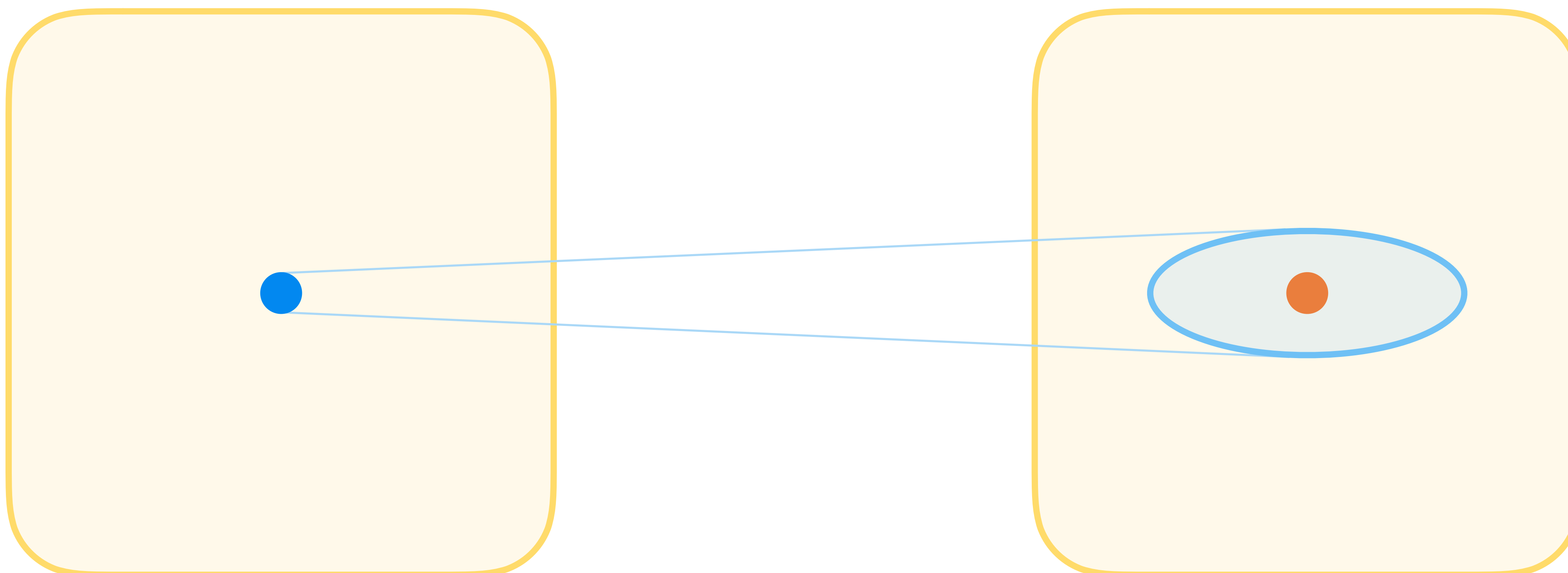
Tests and Goal Coverage



Covering family = Set of tests that **cover all goals**

"Small" covering families = Efficient testing

Random Testing



Pick a random test from **T** Fix a goal from **G**

Suppose **P[**● covers ●**] ≥ p**

Characterize covering families with respect to **p** and **|G|**

Probabilistic Method

Let **G** be the set of goals and **P[**random ● covers ●**]** $\geq$ **p**

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

Probabilistic Method

Let **G** be the set of goals and **P[**random ● covers ●**] ≥ p**

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

Proof.

P[random ● does *not* cover ●] ≤ 1 - p

Probabilistic Method

Let **G** be the set of goals and **P[**random ● covers ●**] ≥ p**

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

Proof.

P[random ● does *not* cover ●] ≤ 1 - p

P[K independent ● do *not* cover ●] ≤ $(1 - p)^K$

Probabilistic Method

Let **G** be the set of goals and **P[**random ● covers ●**] ≥ p**

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

Proof.

P[random ● does *not* cover ●] $\leq 1 - p$

P[K independent ● do *not* cover ●] $\leq (1 - p)^K$

P[K independent ● are *not* a covering family] $\leq |G|\,(1 - p)^K$

Probabilistic Method

Let **G** be the set of goals and **P[**random ● covers ●**] ≥ p**

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

Proof.

P[random ● does *not* cover ●] ≤ 1 - p

P[K independent ● do *not* cover ●] ≤ $(1 - p)^K$

P[K independent ● are *not* a covering family] ≤ $|G| (1 - p)^K$

For $\mathbf{K = p^{-1} \log|G|}$, this probability is **strictly less than 1**.

Therefore, there must exist **K** tests that are a covering family!

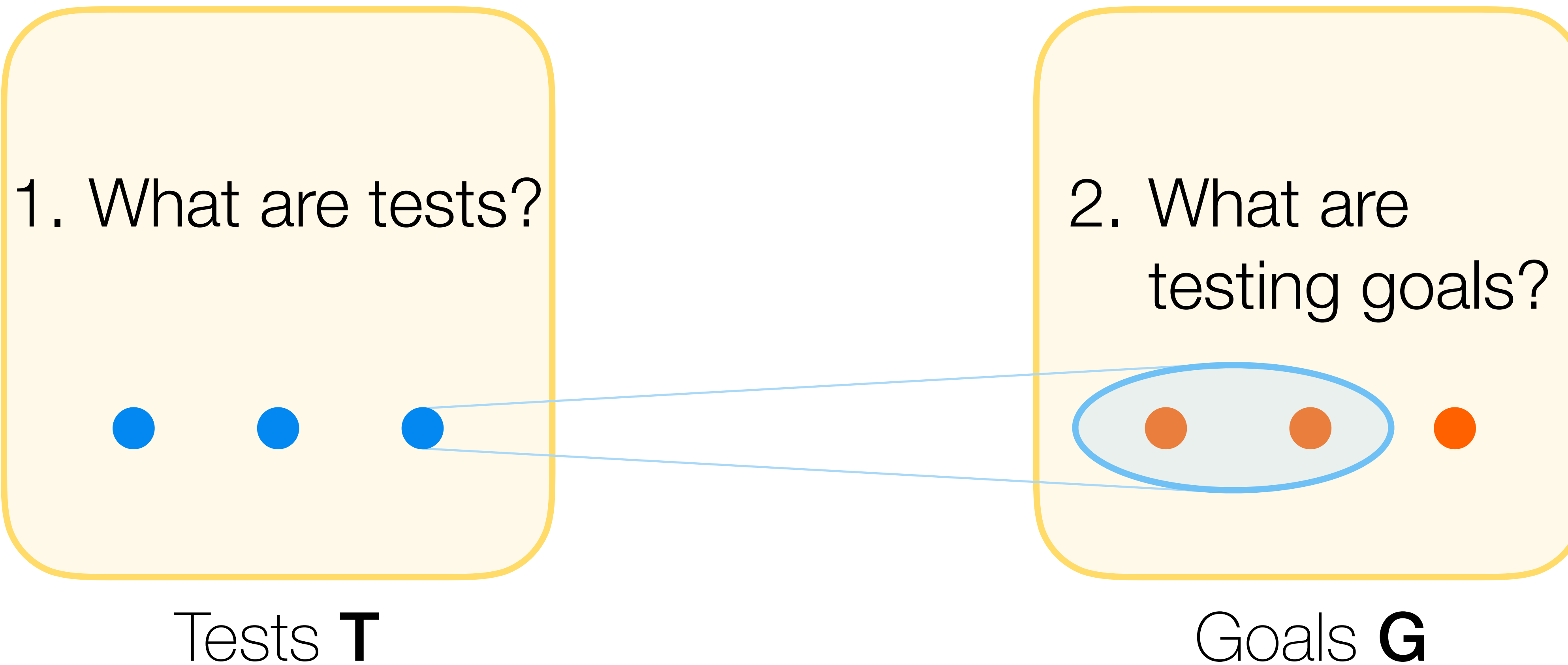
Probabilistic Method

Let $\mathbf{G}$ be the set of goals and $\mathbf{P[}$random ● covers ●$\mathbf{]} \geq \mathbf{p}$

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

Theorem. For $\boldsymbol{\epsilon > 0}$, a random family of $\mathbf{p^{-1} \log|G| + p^{-1} \log \epsilon^{-1}}$ tests is a covering family with probability at least $\mathbf{1 - \epsilon}$.

Random Testing Framework



3. What is the notion of coverage?
4. Can we bound **P[**random ● covers ●**]**?

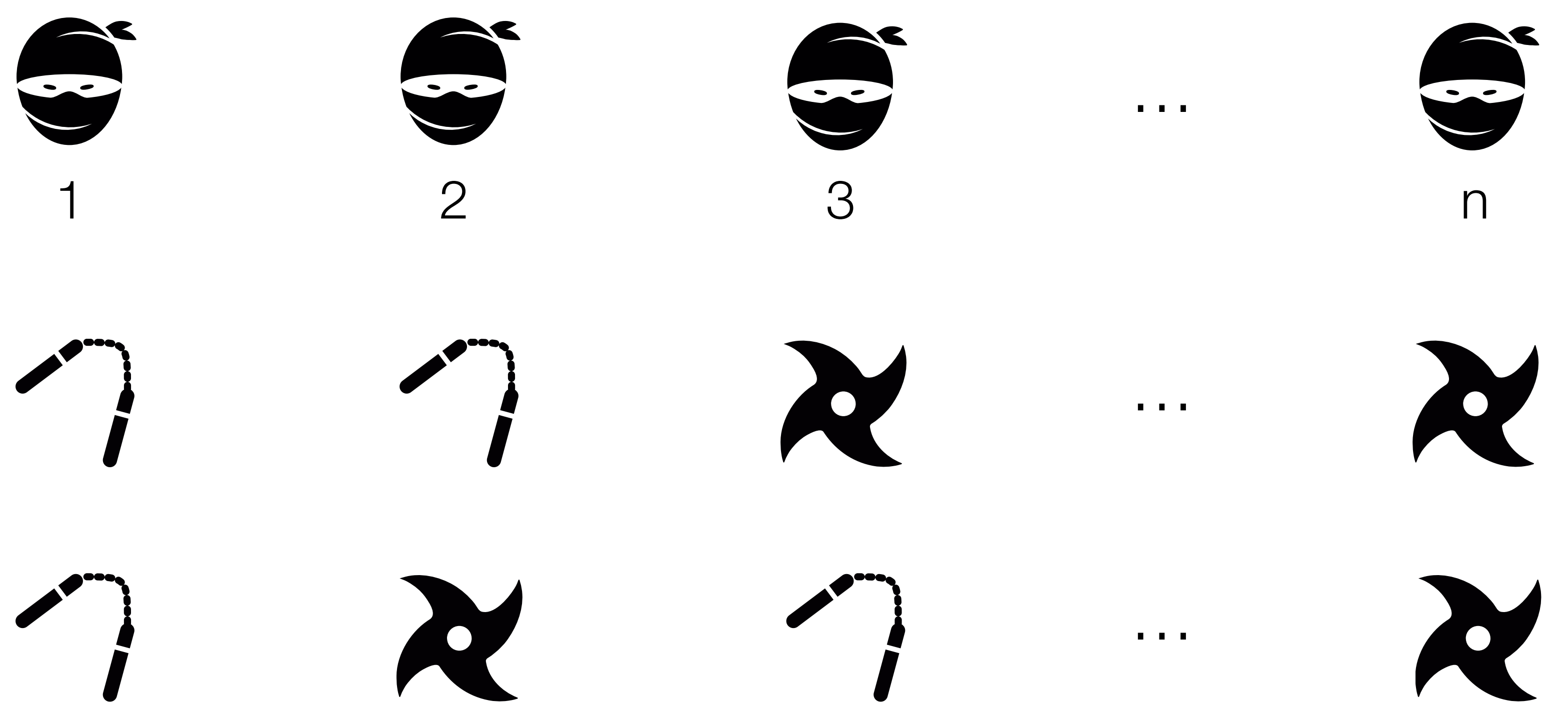
1. General Random Testing Framework
2. Randomly Testing Distributed Systems
3. Wider Context: Combinatorial Testing

Ninjas in Training

In a dojo in Kaiserslautern, **n** ninjas are 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?

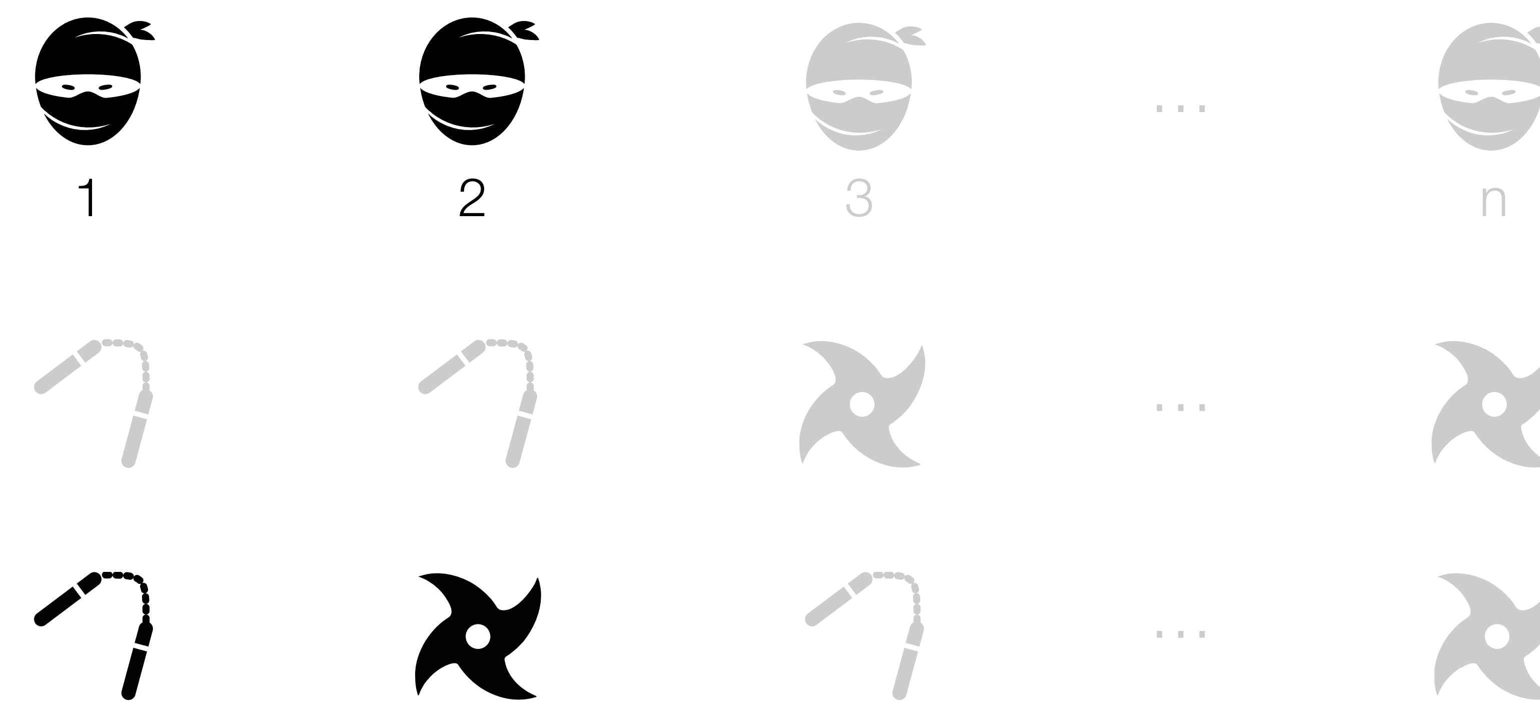


Ninjas in Training

In a dojo in Kaiserslautern, **n** ninjas are 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?



Ninjas in Training

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

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

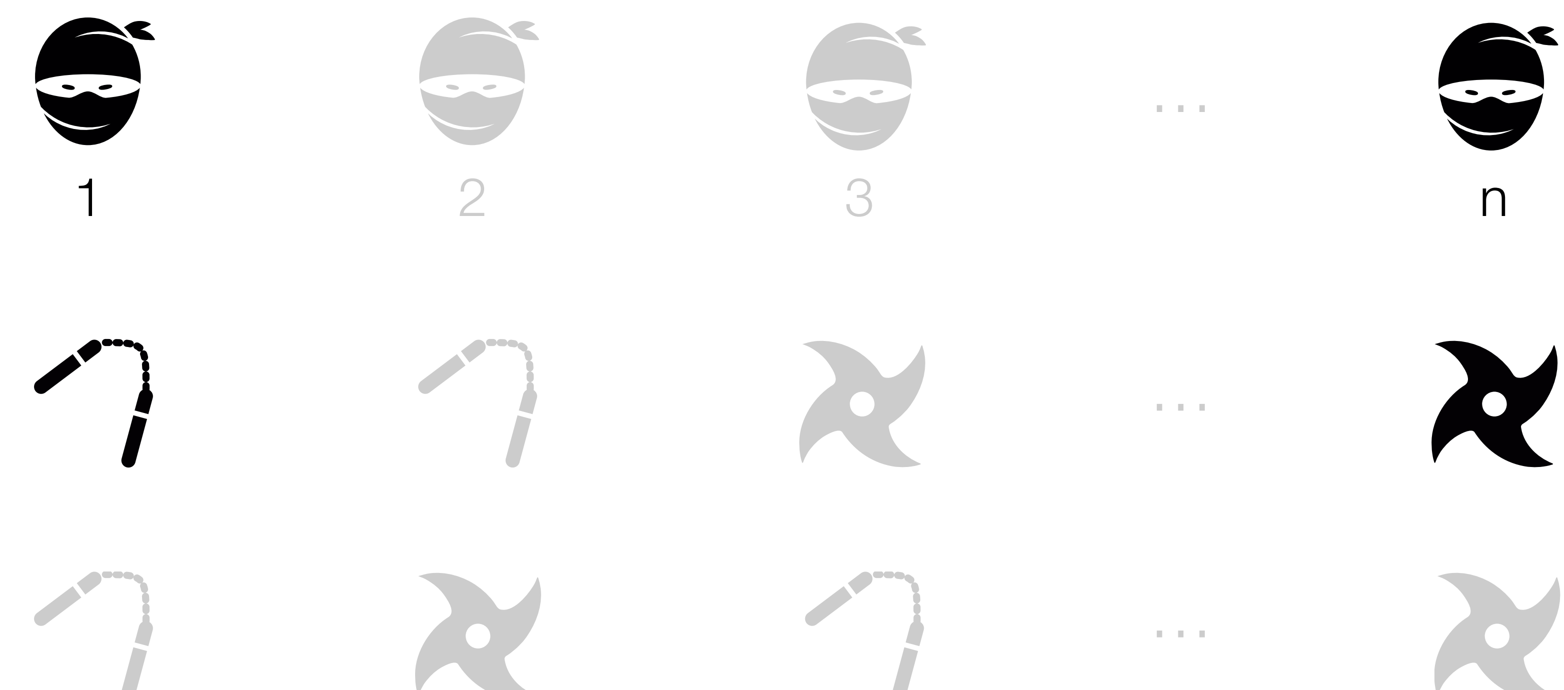
How many rounds make the training complete?

Ninjas in Training

In a dojo in Kaiserslautern, **n** ninjas are 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?

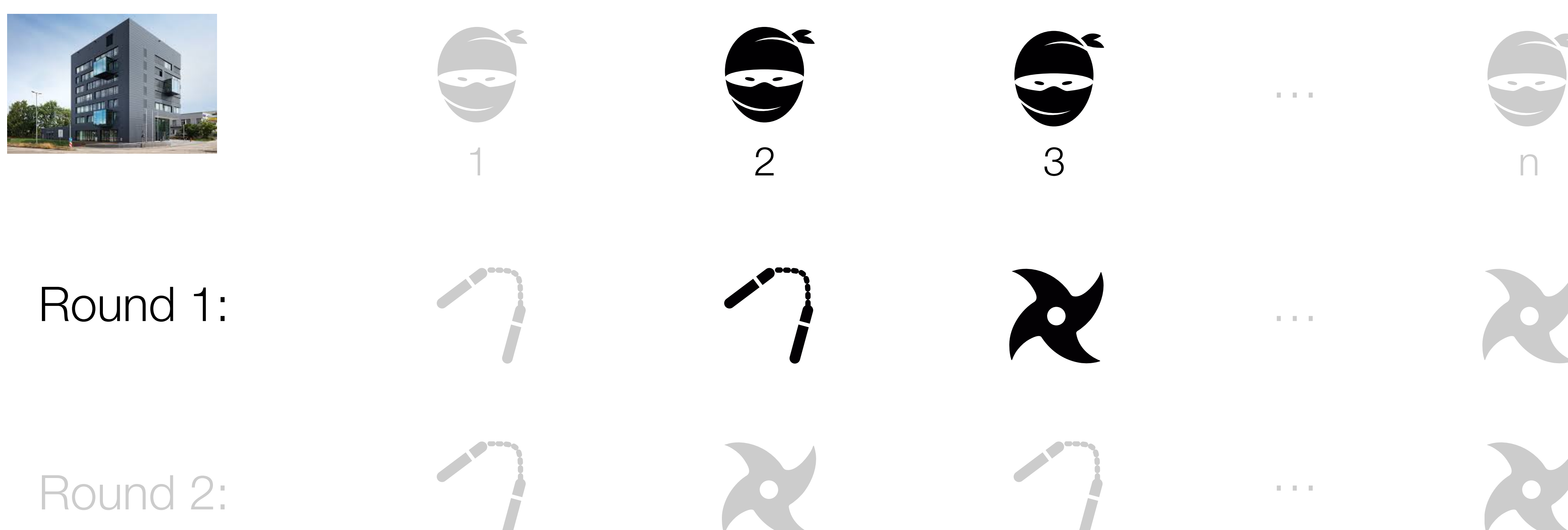


Ninjas in Training

In a dojo in Kaiserslautern, **n** ninjas are 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?

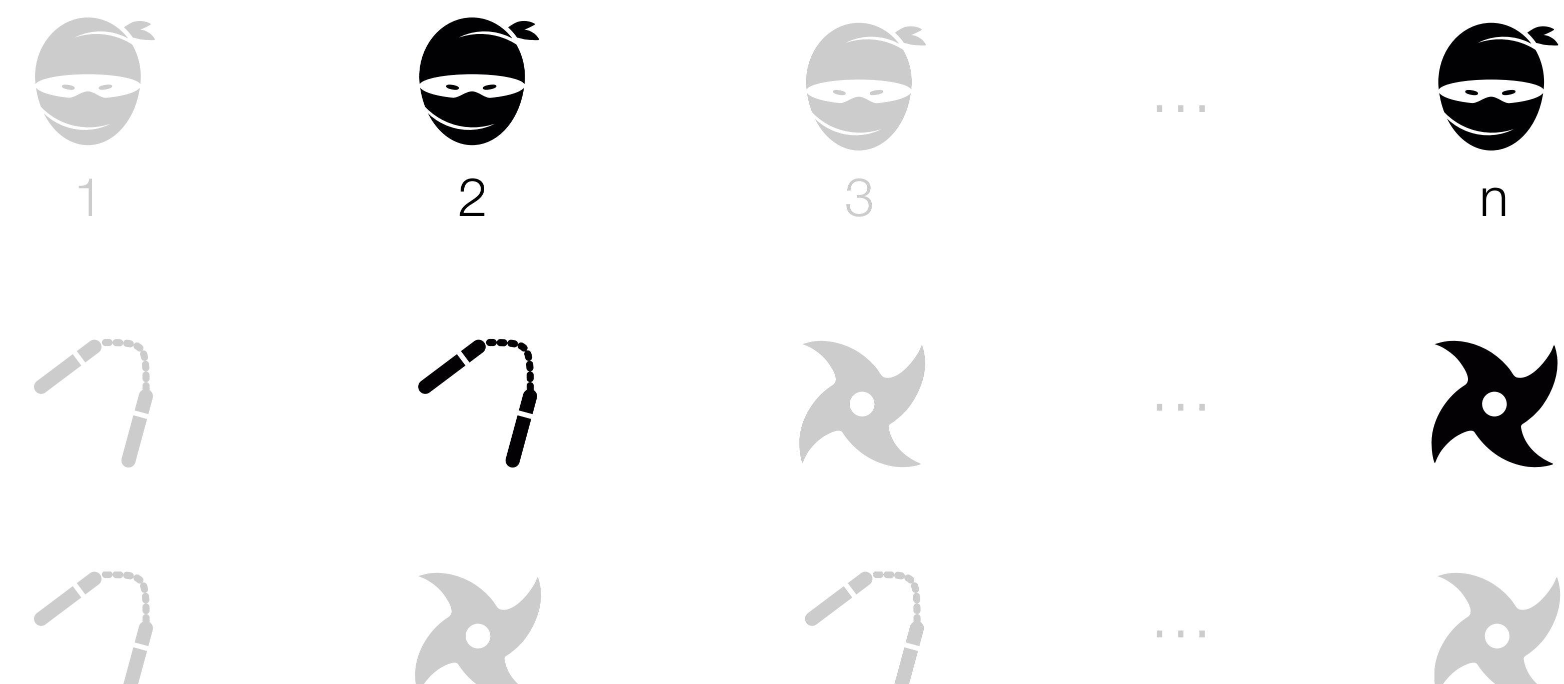


Ninjas in Training

In a dojo in Kaiserslautern, **n** ninjas are 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?

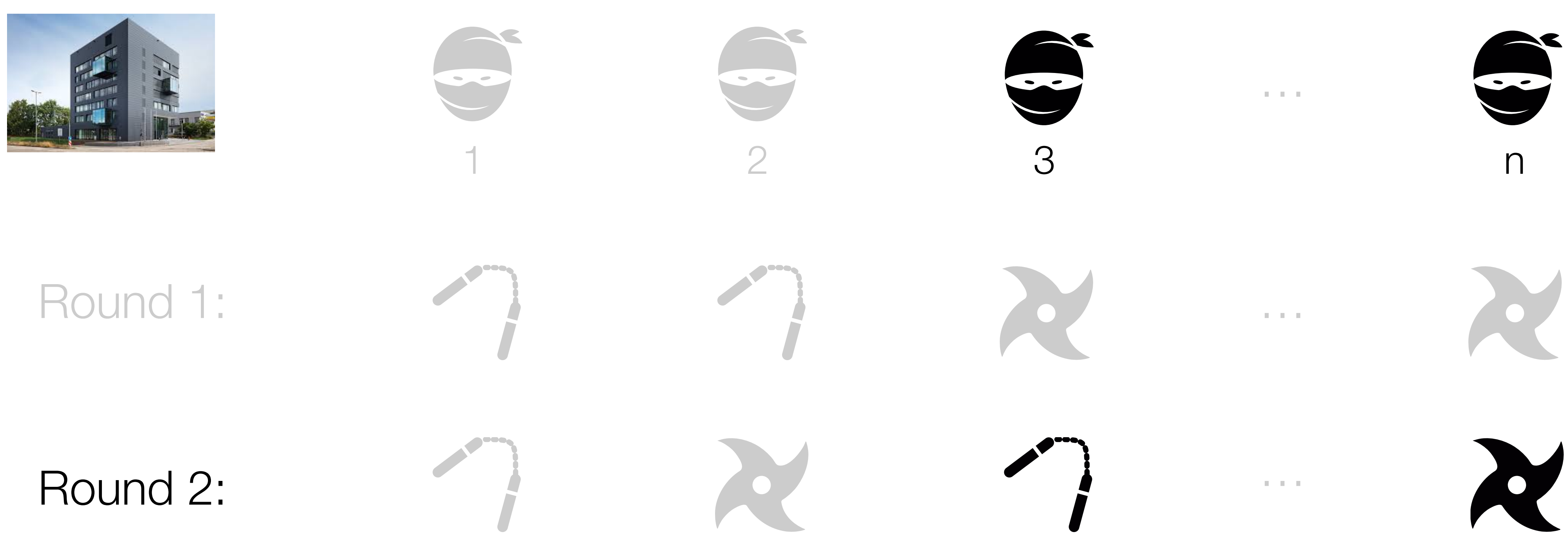


Ninjas in Training

In a dojo in Kaiserslautern, **n** ninjas are 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?



Ninjas in Training

In a dojo in Kaiserslautern, **n** ninjas are 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: $\mathbf{O(n^2)}$

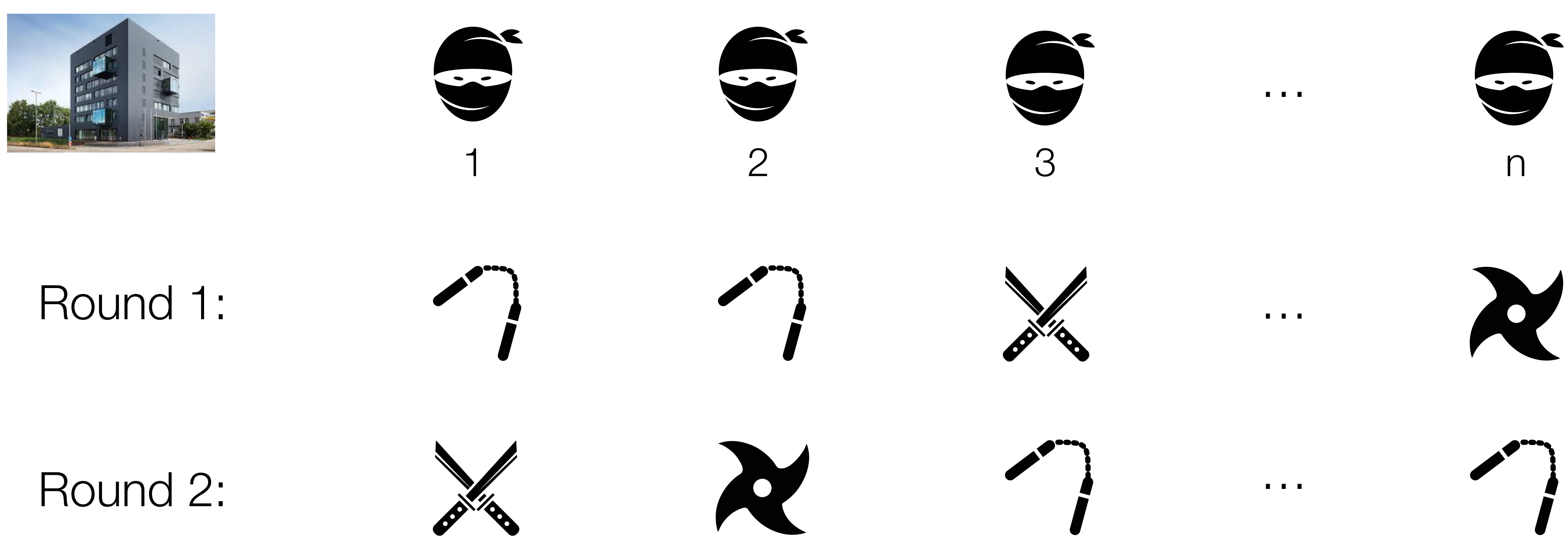
- Can you do it in **log n** rounds?

Ninjas in Training

More generally, **n** ninjas are training in **k** teams.

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

How many rounds make the training complete?

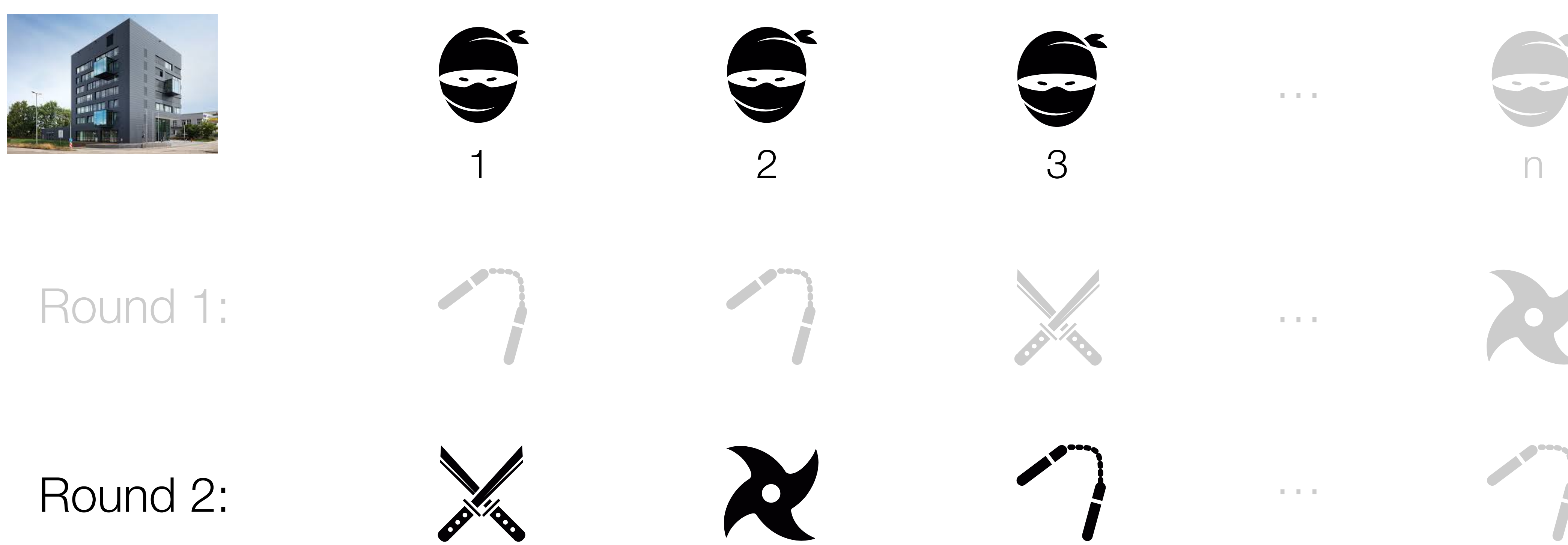


Ninjas in Training

More generally, **n** ninjas are training in **k** teams.

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

How many rounds make the training complete?

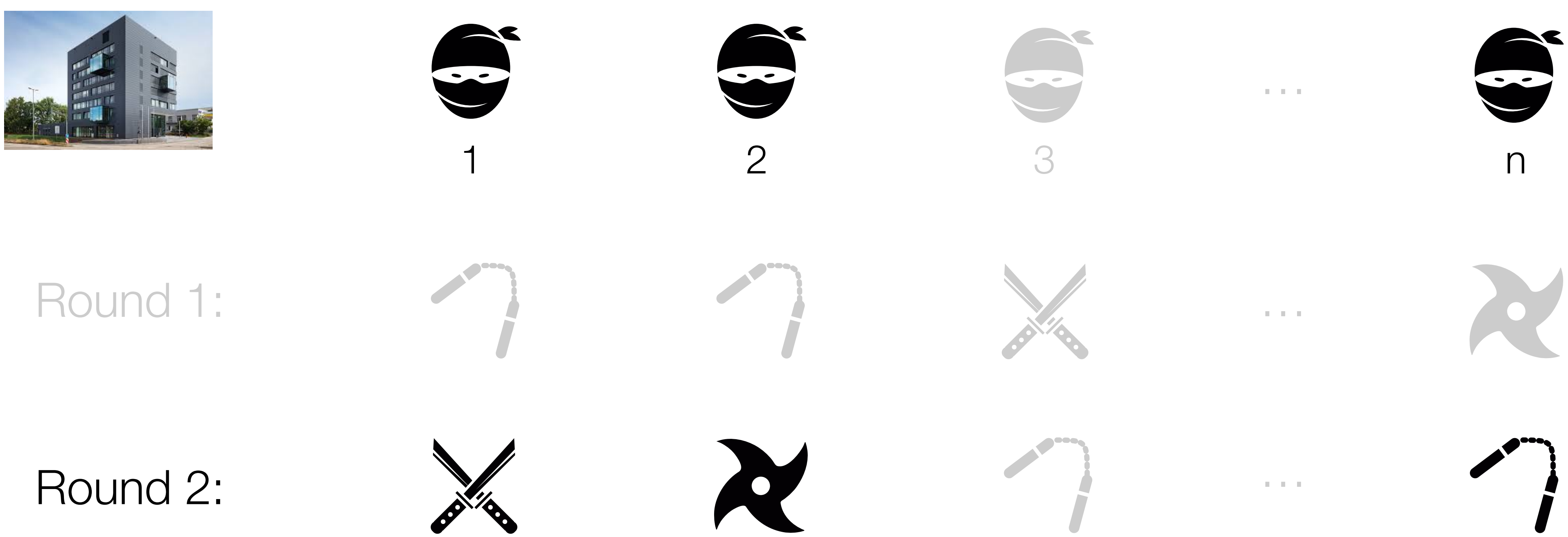


Ninjas in Training

More generally, **n** ninjas are training in **k** teams.

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

How many rounds make the training complete?

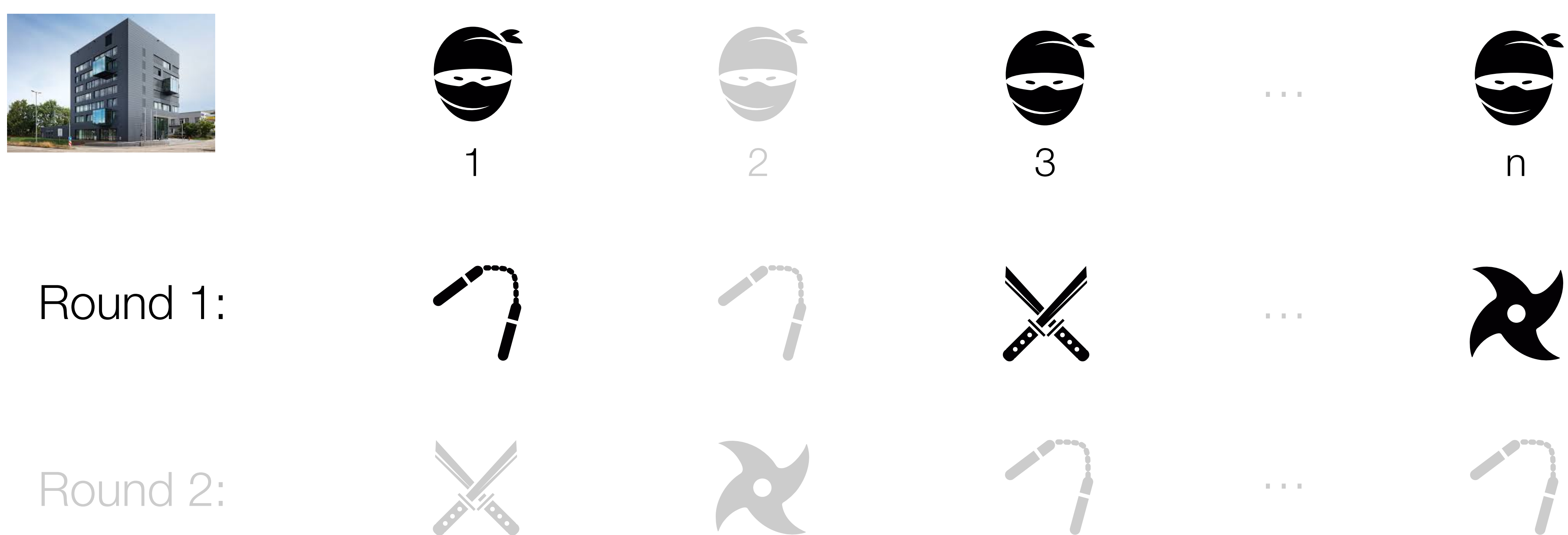


Ninjas in Training

More generally, **n** ninjas are training in **k** teams.

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

How many rounds make the training complete?

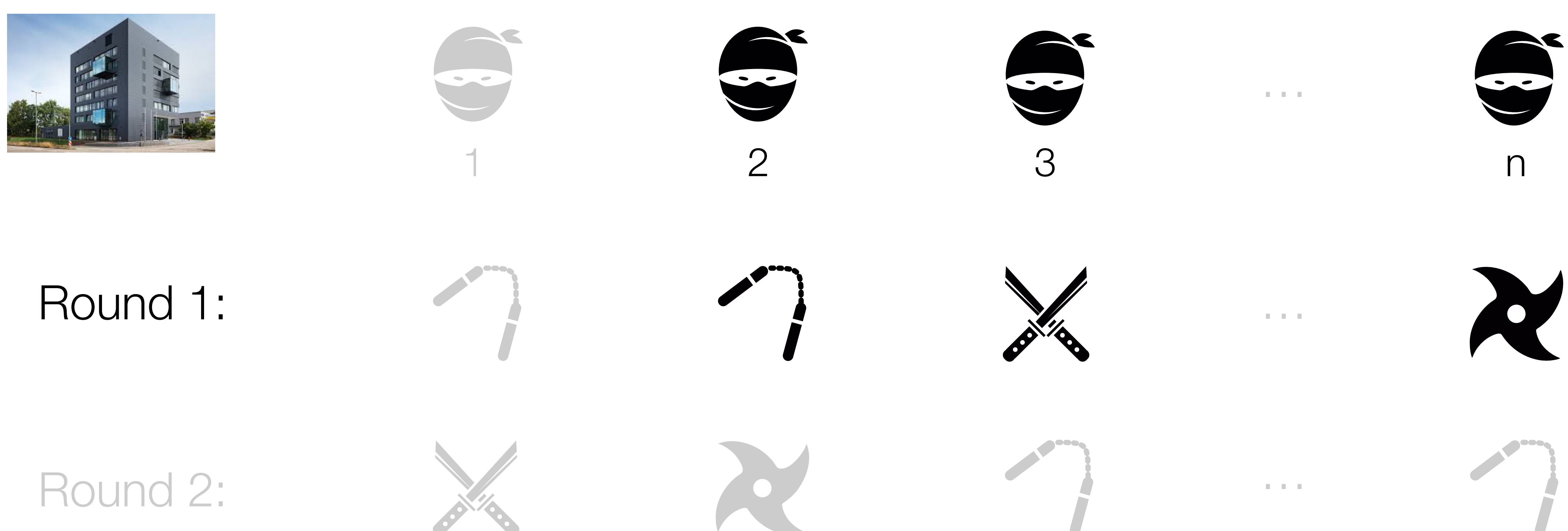


Ninjas in Training

More generally, **n** ninjas are training in **k** teams.

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

How many rounds make the training complete?



Ninjas in Training

More generally, **n** ninjas are training in **k** teams.

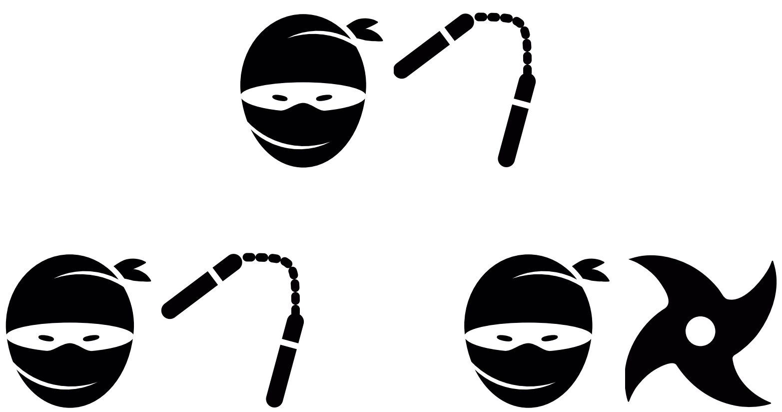
Training is **complete** if for every choice of **k** ninjas, there is a round where they are each in different team.

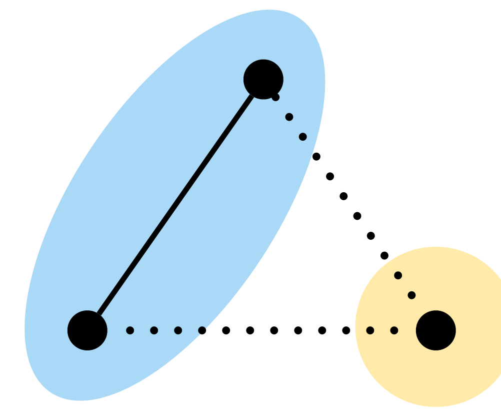
How many rounds make the training complete?

- Naïve: $\mathbf{O(n^k)}$

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

From Training Ninjas to Distributed Systems with Partition Faults





ninjas	nodes in a network
teams	blocks in a partition
rounds	partitions
complete training	**covering family**

Splitting Coverage

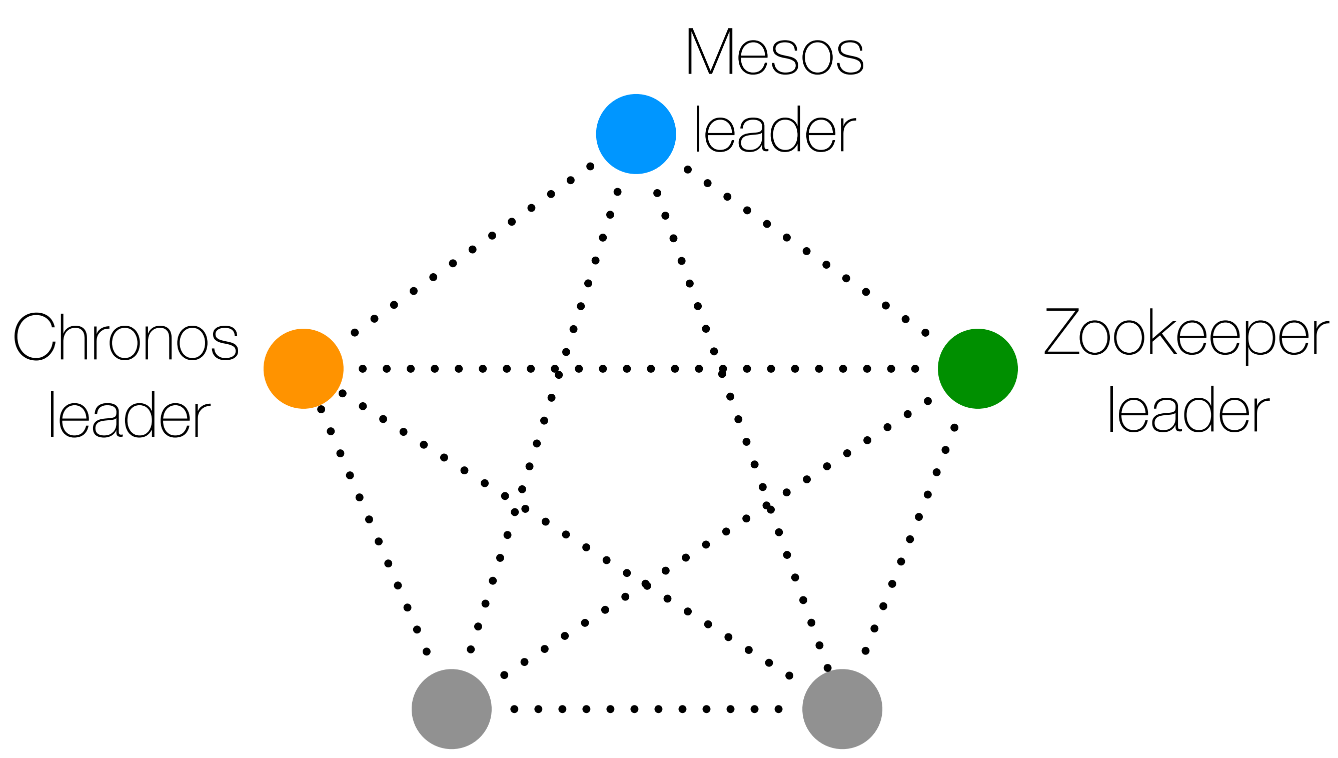
Given **n** nodes and $\mathbf{k \leq n}$:

- Tests are partitions of nodes into **k** blocks: $\mathbf{P = \{B_1, \ldots, B_k\}}$
- Testing goals are sets of **k** nodes: $\mathbf{S = \{x_1, \ldots, x_k\}}$
- **P** covers **S** if **P** splits **S**: $\mathbf{x_1} \in \mathbf{B_1, \ldots, x_k} \in \mathbf{B_k}$

Covering families are called **k-splitting families** here

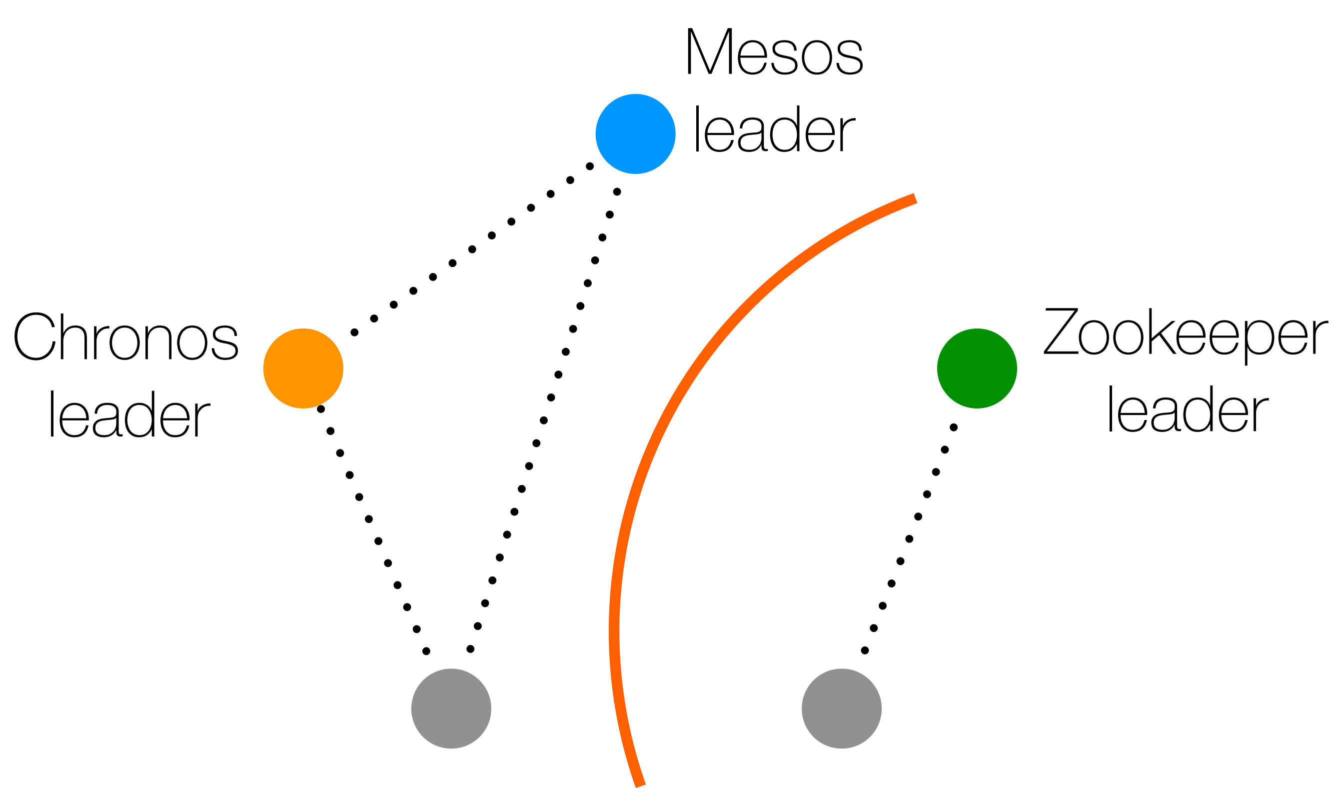
A Bug in Chronos

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



A Bug in Chronos

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



Filip

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

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

Splitting Coverage

Given **n** nodes and **k ≤ n**:

- Number of partitions with **k** blocks: $\left\{ {n \atop k} \right\} \approx \frac{k^n}{k!}$
- Number of sets of **k** nodes: $\binom{n}{k} \approx \frac{n^k}{k!}$
- Splitting a set with a random partition: $p = \frac{k^{n-k}}{\left\{ {n \atop k} \right\}} \approx \frac{k!}{k^k}$

By the general theorem, there exists a **k**-splitting family of size $\mathbf{k^{k+1} (k!)^{-1} \log n}$

Effectiveness of Jepsen

Theorem. For $\boldsymbol{\epsilon > 0}$, a random family of partitions of size $\mathbf{k^{k+1}\,(k!)^{-1}\,\log n + k^k\,(k!)^{-1}\,\log \epsilon^{-1}}$ is a **k**-splitting family with probability at least $\mathbf{1 - \epsilon}$.

For Chronos, with $\mathbf{n = 5}$, $\mathbf{k = 2}$, $\boldsymbol{\epsilon} \mathbf{= 0.2}$: a family of **10** randomly chosen partitions is splitting with probability **80%**

Other Coverage Notions

k,l-Separation

- Tests: Bipartitions
- Goals: Two disjoint sets of **k** and **l** nodes
- Coverage notion: The two sets included in different blocks
- Size of covering families: **O(f(k,l) log n)**

Minority isolation

- Tests: Bipartitions
- Goals: Nodes
- Coverage notion: The node is in the smaller block
- Covering families: **O(log n)**

Other Coverage Notions

k,l-Separation

- Tests: Bipartitions

- **k**-Splitting, **k**,**l**-separation, and minority isolation explain most bugs found by Jepsen

- Goals: Nodes
- Coverage notion: The node is in the smaller block
- Covering families: **O(log n)**

Other Coverage Notions

k,l-Separation

- Tests: Bipartitions

- **k**-Splitting, **k**,**l**-separation, and minority isolation explain most bugs found by Jepsen
- With high probability, **O(log n)** random partitions simultaneously provide full coverage for all these notions

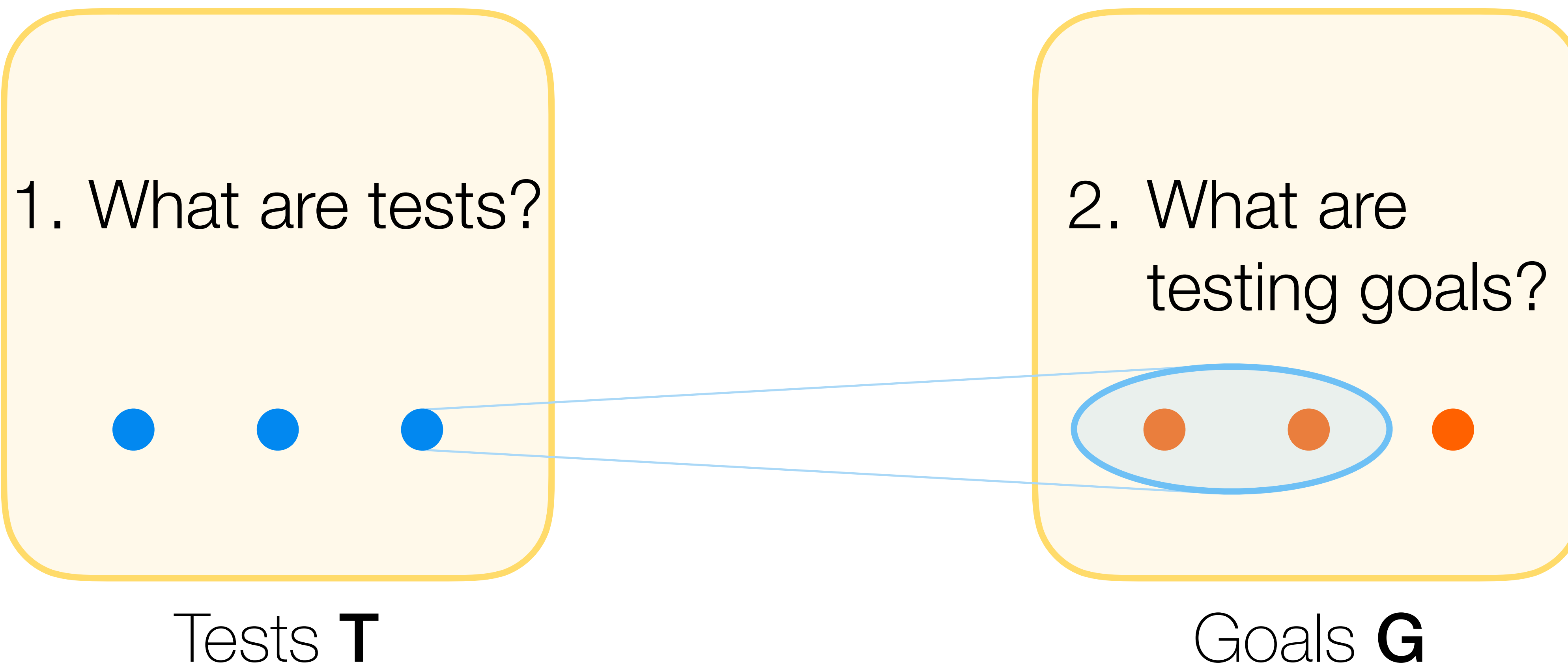
- Goals: Nodes
- Coverage notion: The node is in the smaller block
- Covering families: **O(log n)**

1. General Random Testing Framework

2. Randomly Testing Distributed Systems

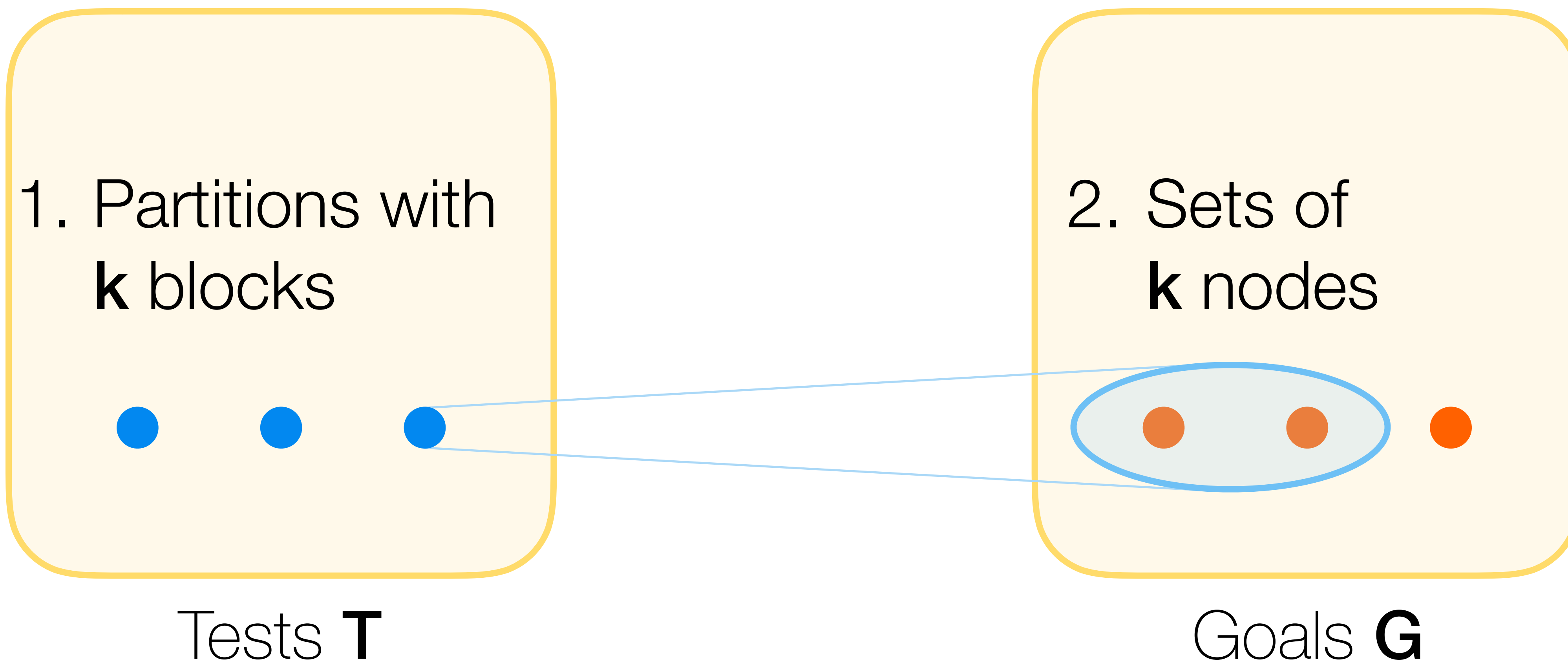
3. Wider Context: Combinatorial Testing

General Testing Framework



3. What is the notion of coverage?
4. How to construct covering families?

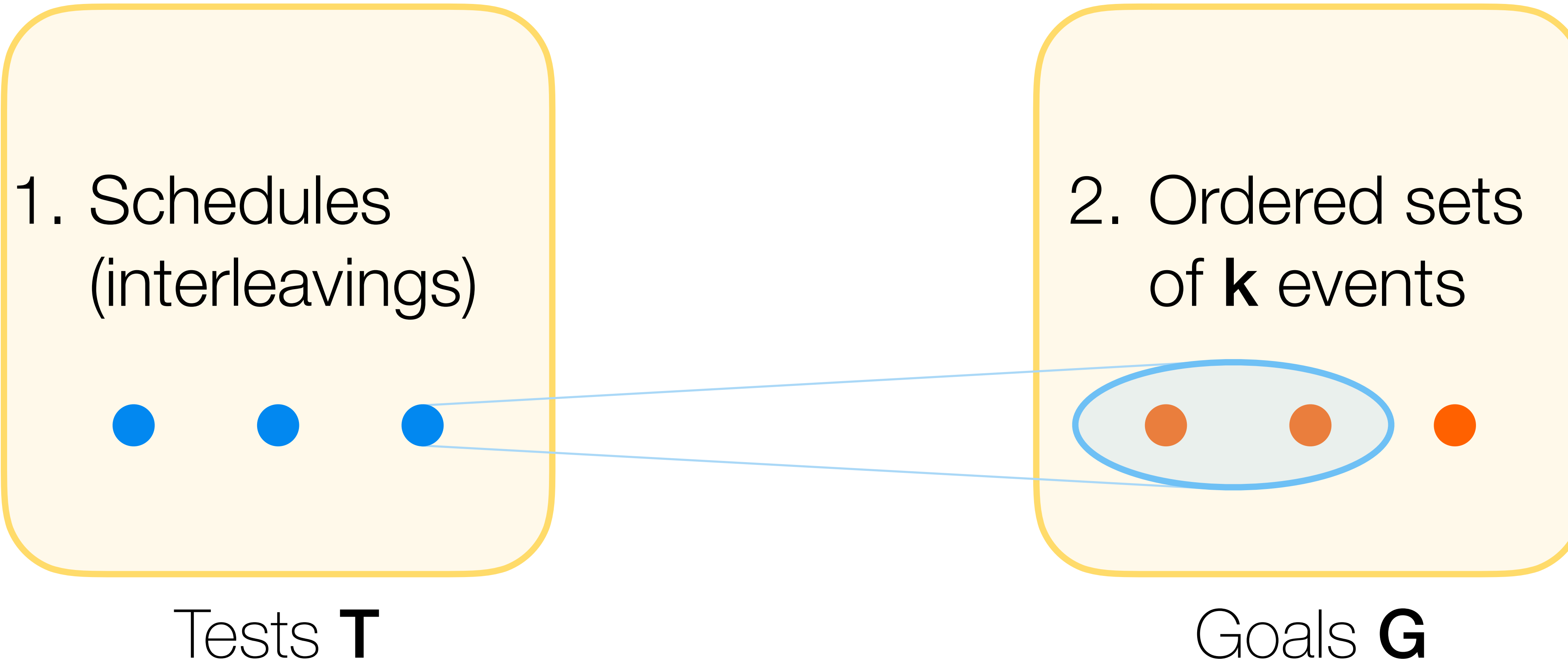
Distributed Systems with Network Partitions



3. **k**-splitting coverage
4. Random families of size **O(log n)** are **k**-splitting w.h.p.

Concurrent Programs

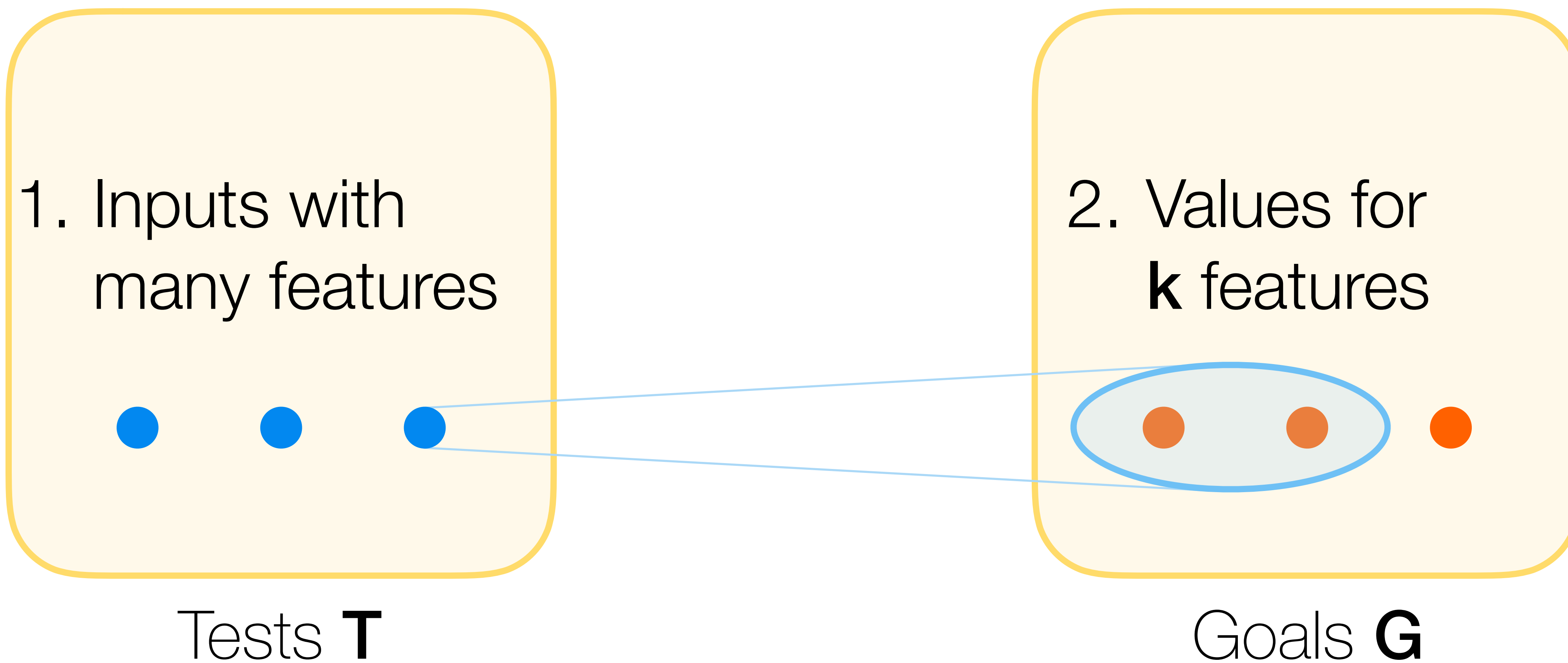
Program = Partially ordered set of events



3. **k**-hitting coverage: Schedule “hits” events $\mathbf{e_1 < \ldots < e_k}$
4. Hitting families of size **O(log n)**, **O(log n)$^{k-1}$**, **O(n^{k-1})**

Chistikov, Majumdar, Niksic. *Hitting families of schedules for asynchronous programs.* CAV 2016
Burckhardt et al. *A randomized scheduler with probabilistic guarantees of finding bugs*. ASPLOS 2010

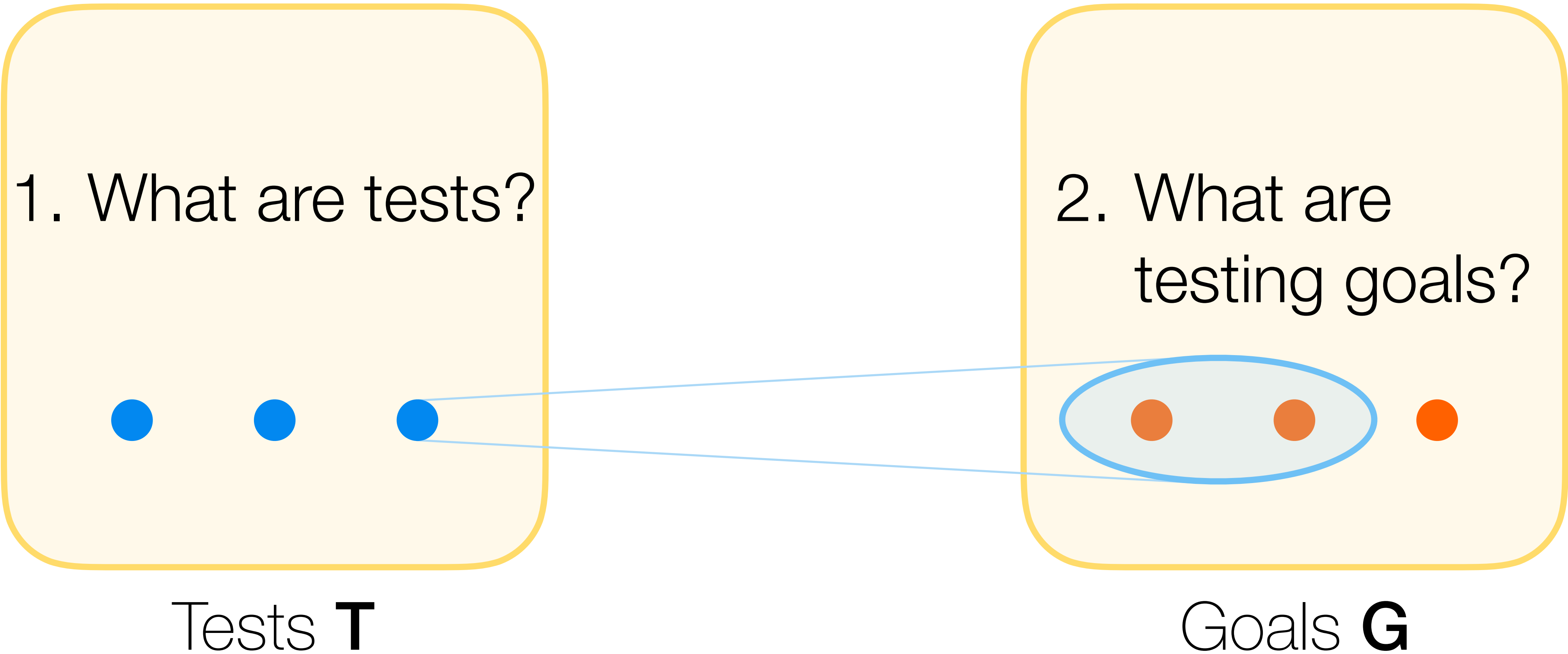
Combinatorial Testing



3. Input coincides with the chosen values on the **k** features
4. Various constructions of covering arrays

Kuhn, Kacker, Lei. *Combinatorial Testing*. Encyclopedia of Software Engineering. 2010

General Testing Framework



3. What is the notion of coverage?
4. How to construct covering families?

General Testing Framework

Where else can we apply this approach?

3. What is the notion of coverage?
4. How to construct covering families?