Dealing with Dependent Failures in Distributed Systems

Ranjita Bhagwan, Henri Casanova, Alejandro Hevia, Flavio Junqueira, Yanhua Mao, Keith Marzullo, Geoff Voelker, Xianan Zhang

University California San Diego

Dept. of Computer Science & Engineering

Research Interests

Work in the *formal/practical boundary of distributed systems*.

Basic strategy:

- Identify a practical issue
- Develop an abstraction of the problem
- Work at abstract level
- Bring back down to practical issue

Being Abstract About Failures

When designing or analyzing a protocol, it is usually done in the context of a *failure model*

- What are the base components? (processes, networks, processors, controllers, ...)
- How can components deviate from correct behavior? (crash, produce the wrong output, drop input, ...)
- How many of them can fail? (it is highly unlikely that more than this set can fail)

Threshold Failure Models

- Consider only processes.
- Out of n processes, no more than t can be faulty.
  - Easy to reason about.
  - Useful for expressing bounds.
  - In practice, compute t and n offline

Some examples of lower bounds

- $n > 2t$ for crash consensus
- $n > 3t$ for arbitrary consensus
- $n > 4t$ for Byzantine quorum systems
- $n > 3t/2$ for receive-omission primary-backup

Implication of using Threshold Model

Threshold model is an outgrowth of an early, influential project in fly-by-wire (SIFT).

Any lower bounds derived on it are based on an implicit assumption of *independent and identically distributed failures* (*IID*).

- Any subset of *t* or fewer processes can be faulty
- Attained by careful design of the system

Dealing with Dependent Failures

Nonidentically distributed and dependent failures are now the common case.

- Heterogeneous components.
- Clusters with shared components.
- Networks of hosts with shared vulnerabilities under attack by malware.

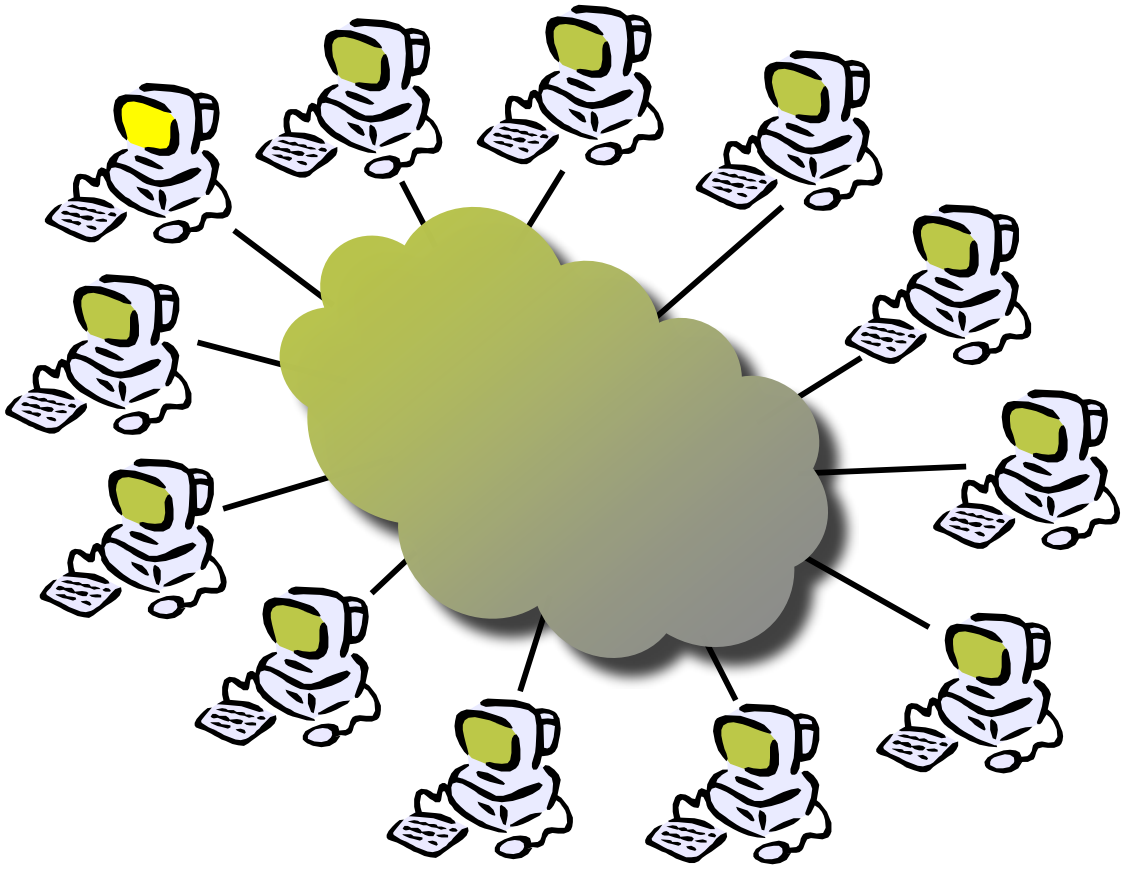
To use a protocol designed with the threshold model, choose *t* large enough to cover all failure situations.

Generalizing thresholds

Definition

- A *core* is a minimal set of processes such that in all executions, at least one of them will be nonfaulty.
  - It isn't known *a priori* which are faulty.
  - With threshold model, any subset of $t + 1$ processes.

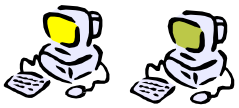
An example

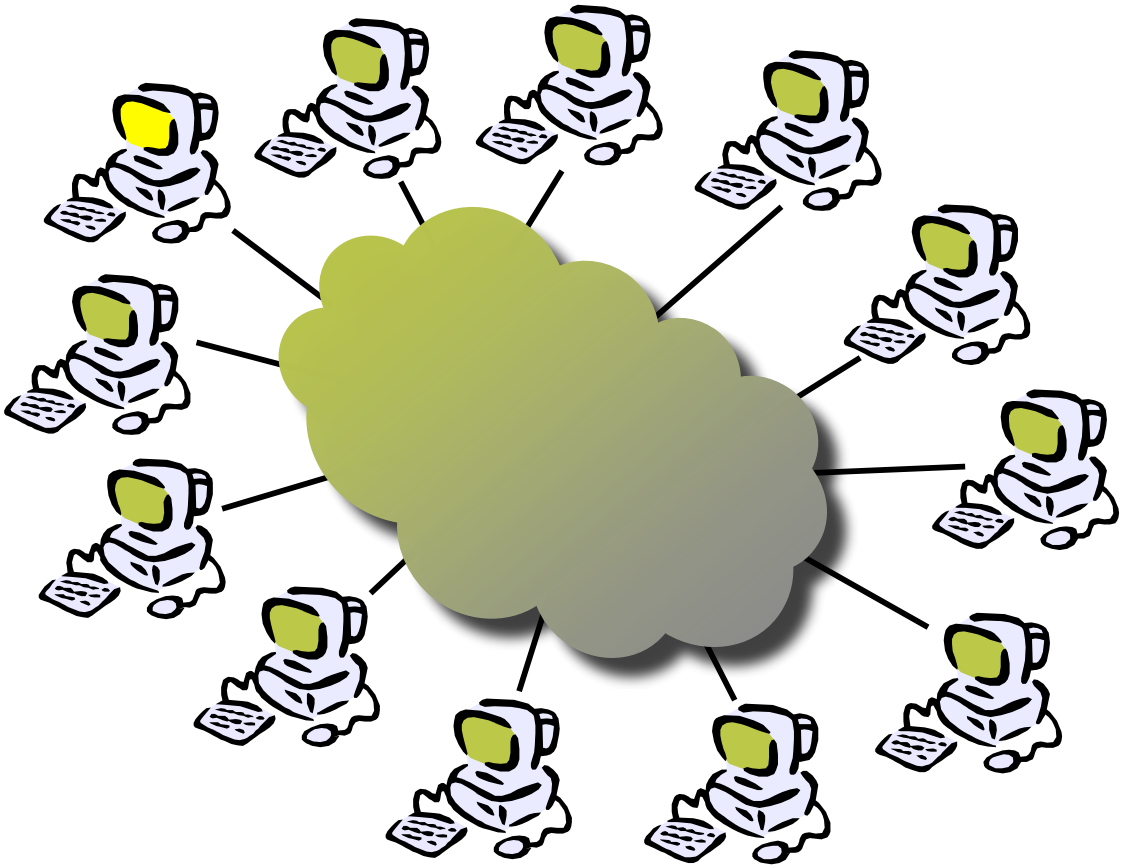


$n = 12$

An example

cores

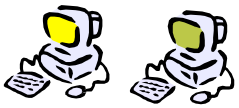


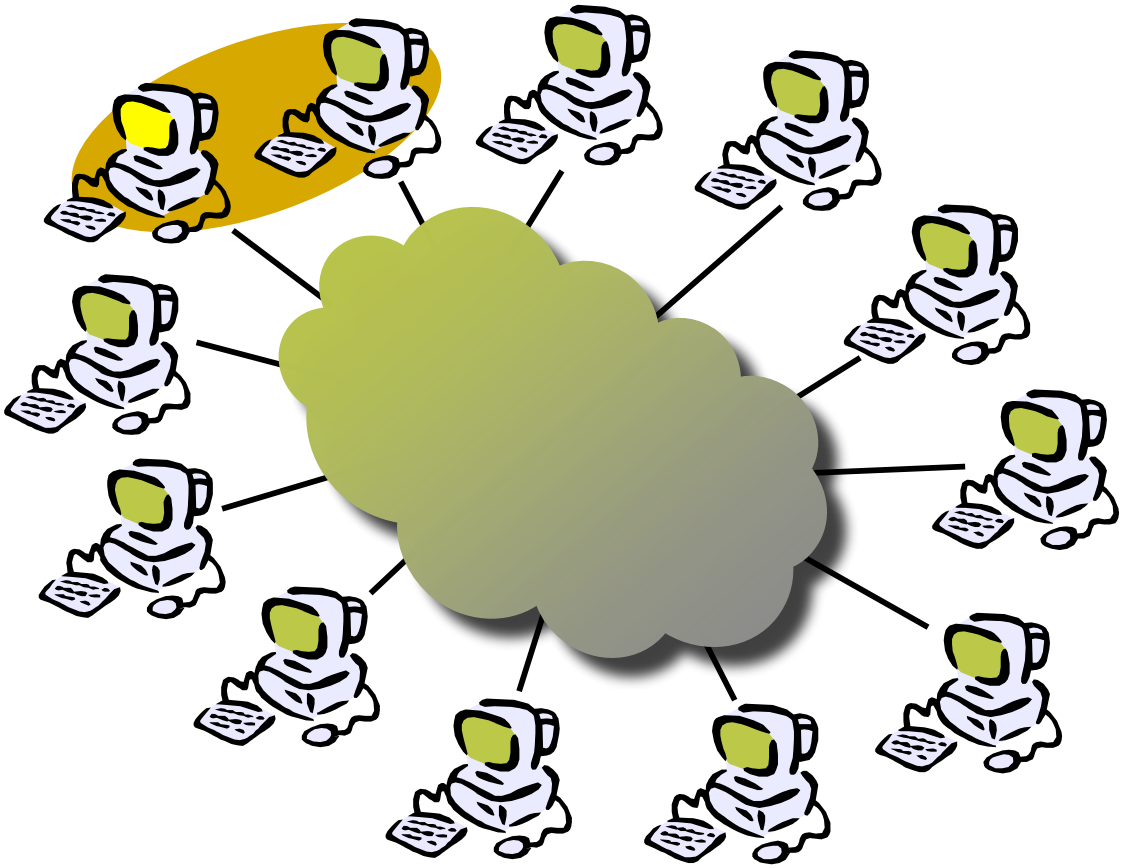


$n = 12$

An example

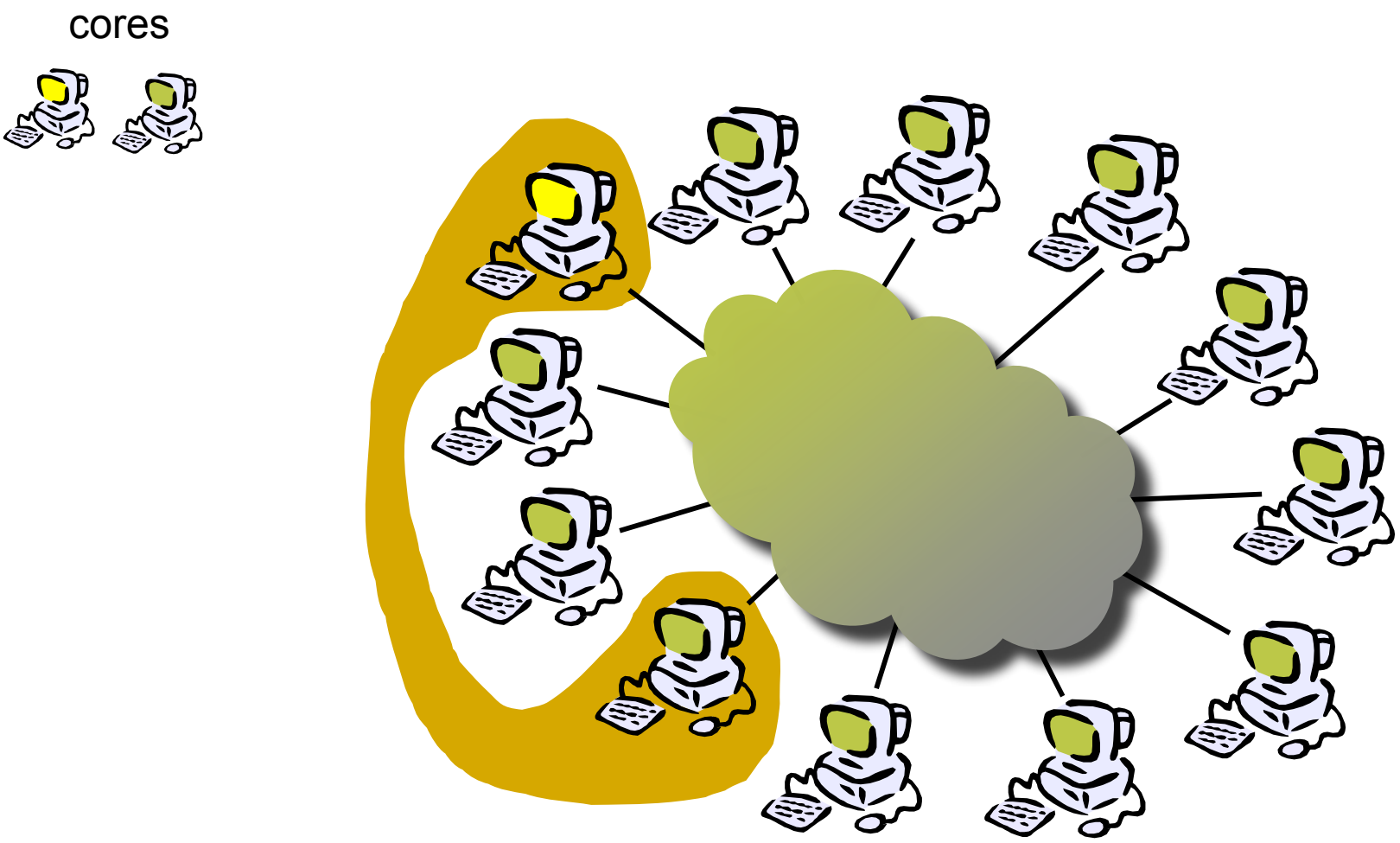
cores





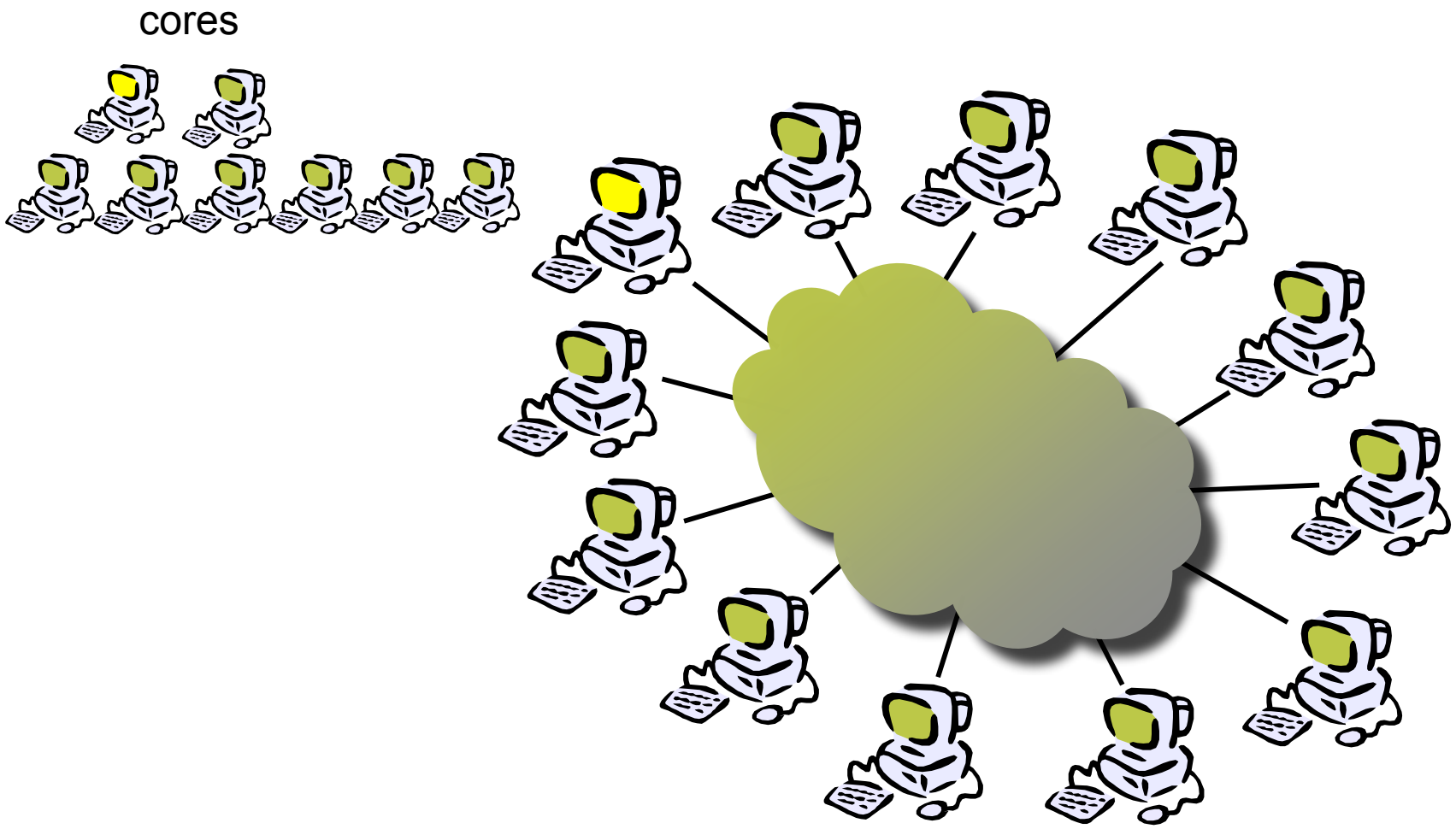
$n = 12$

An example



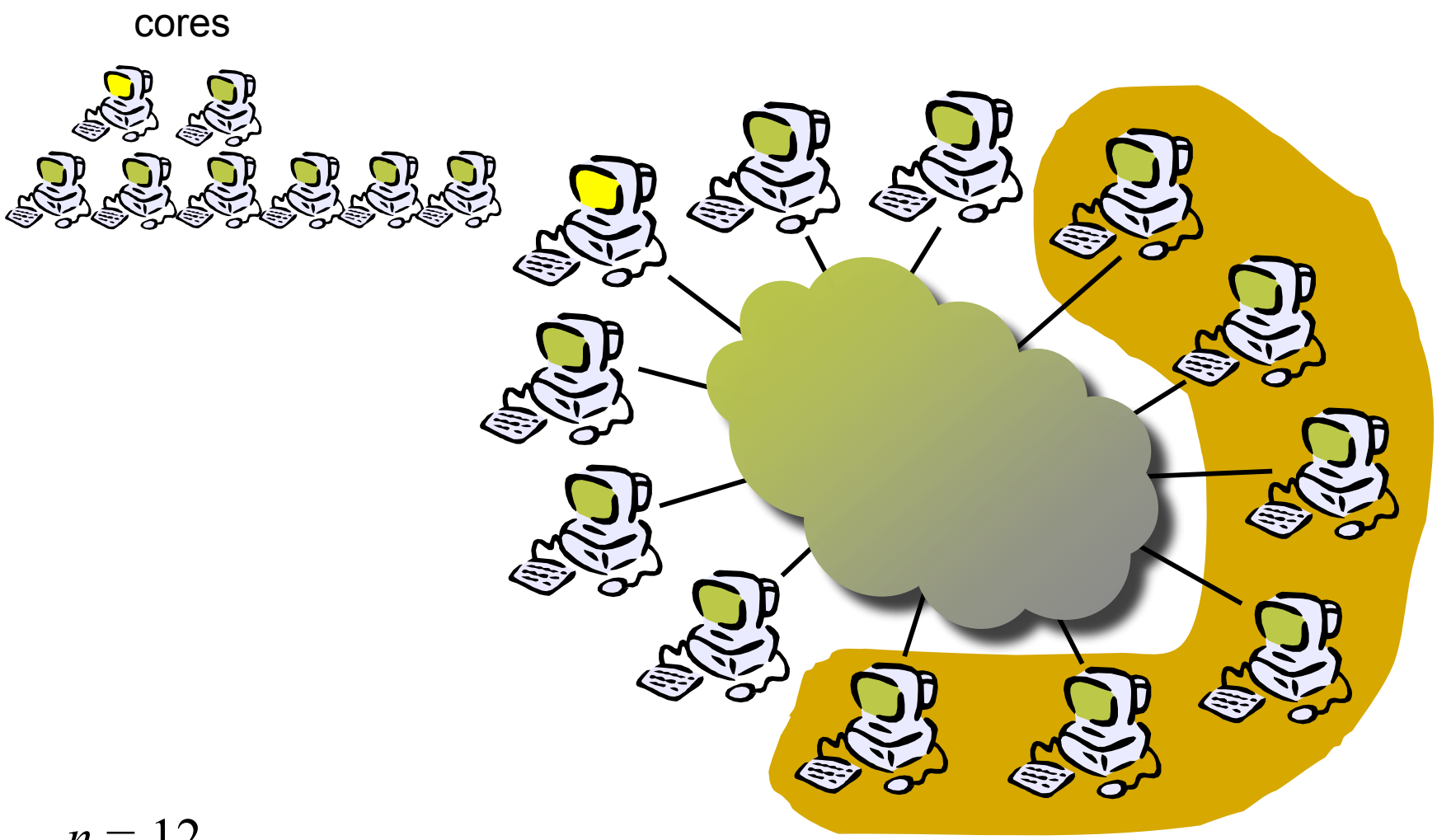
$n = 12$

An example



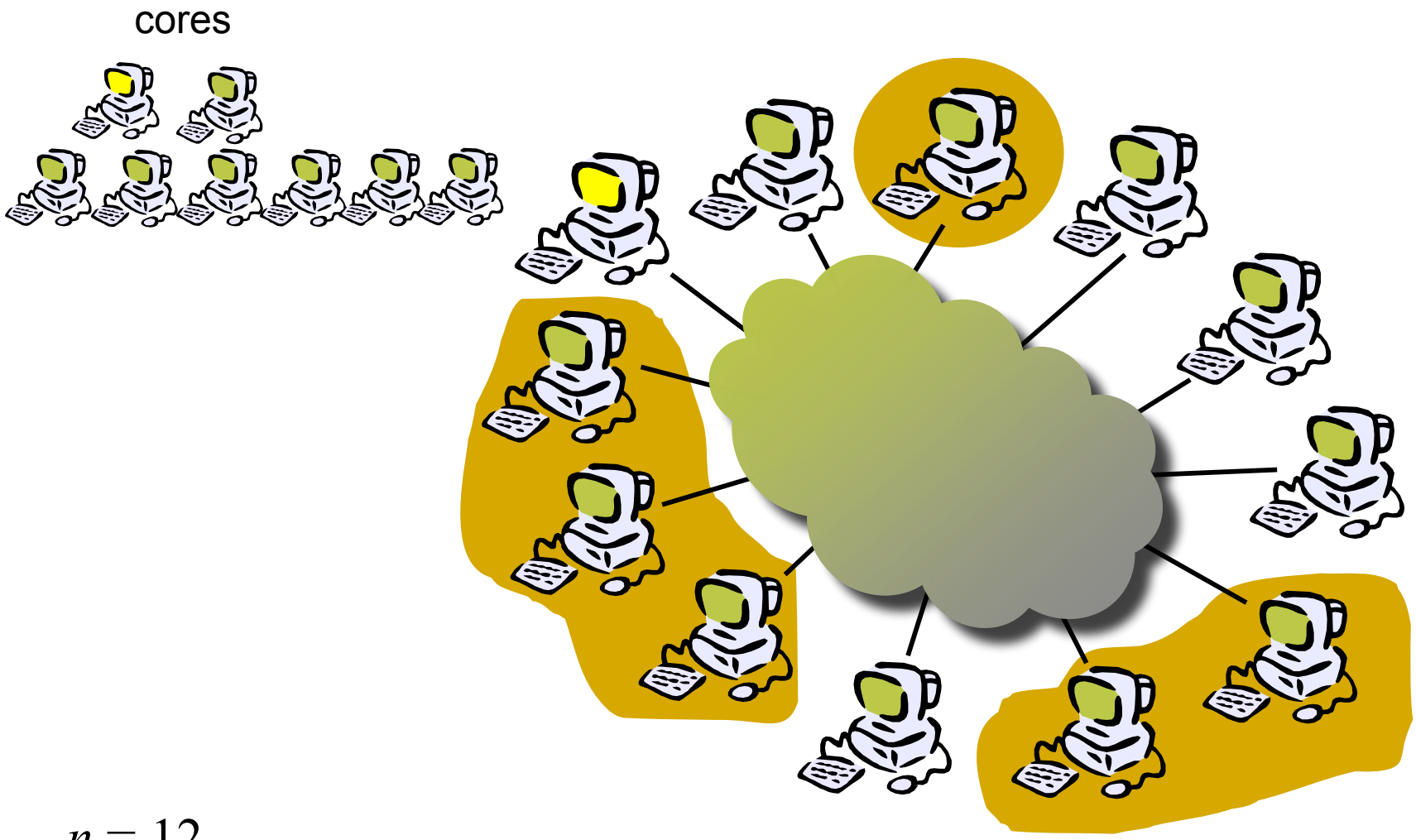
$n = 12$

An example



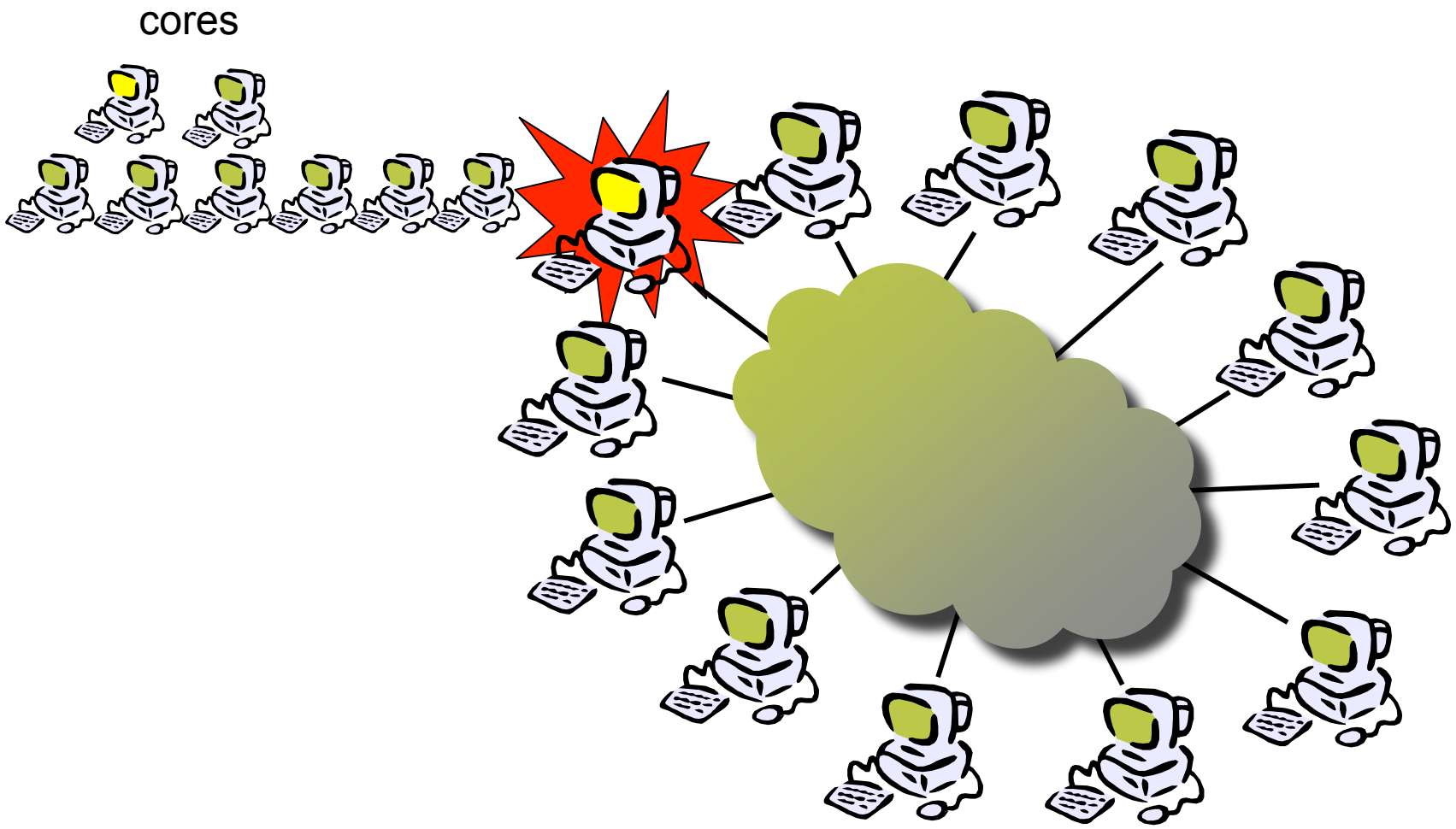
$n = 12$

An example



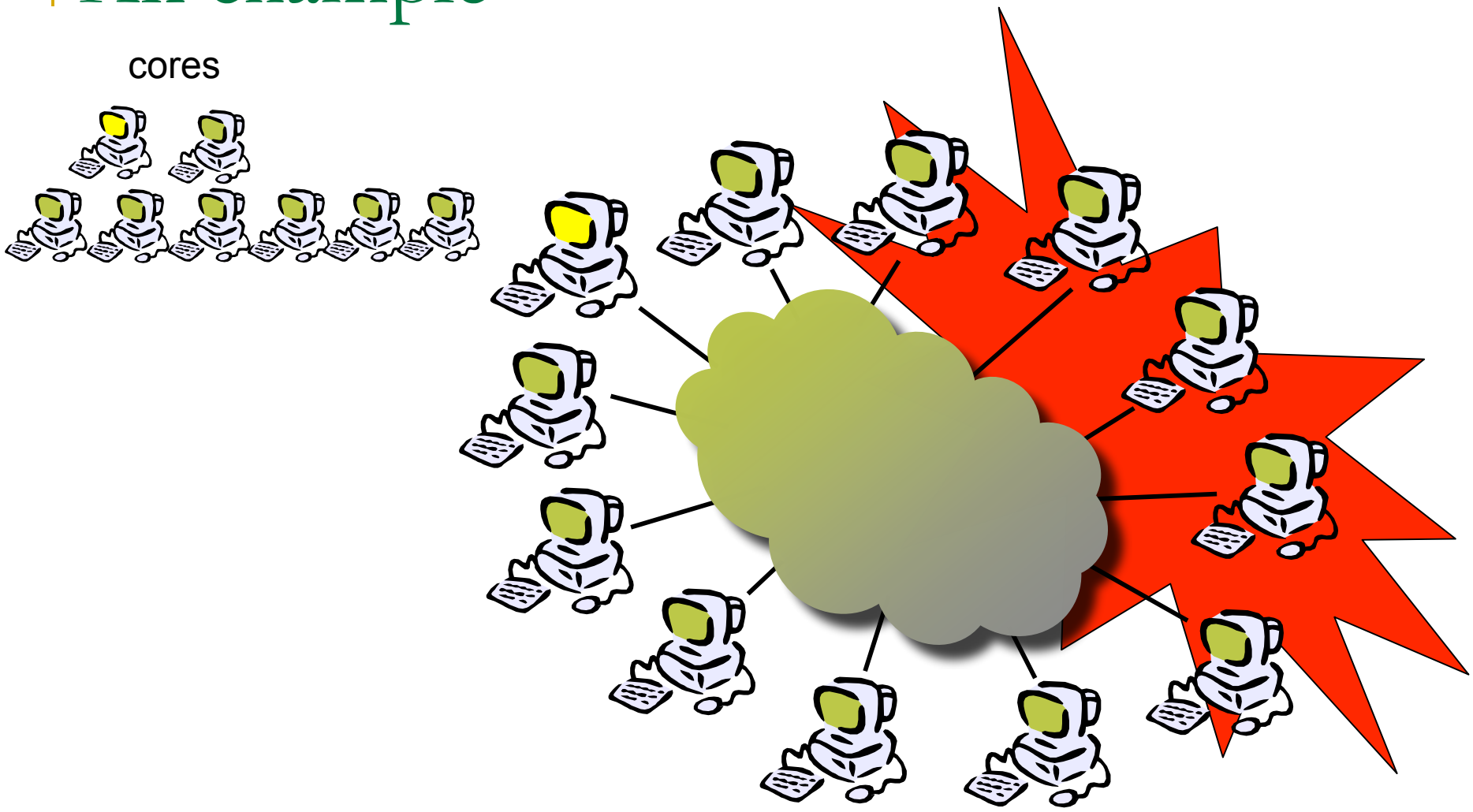
$n = 12$

An example



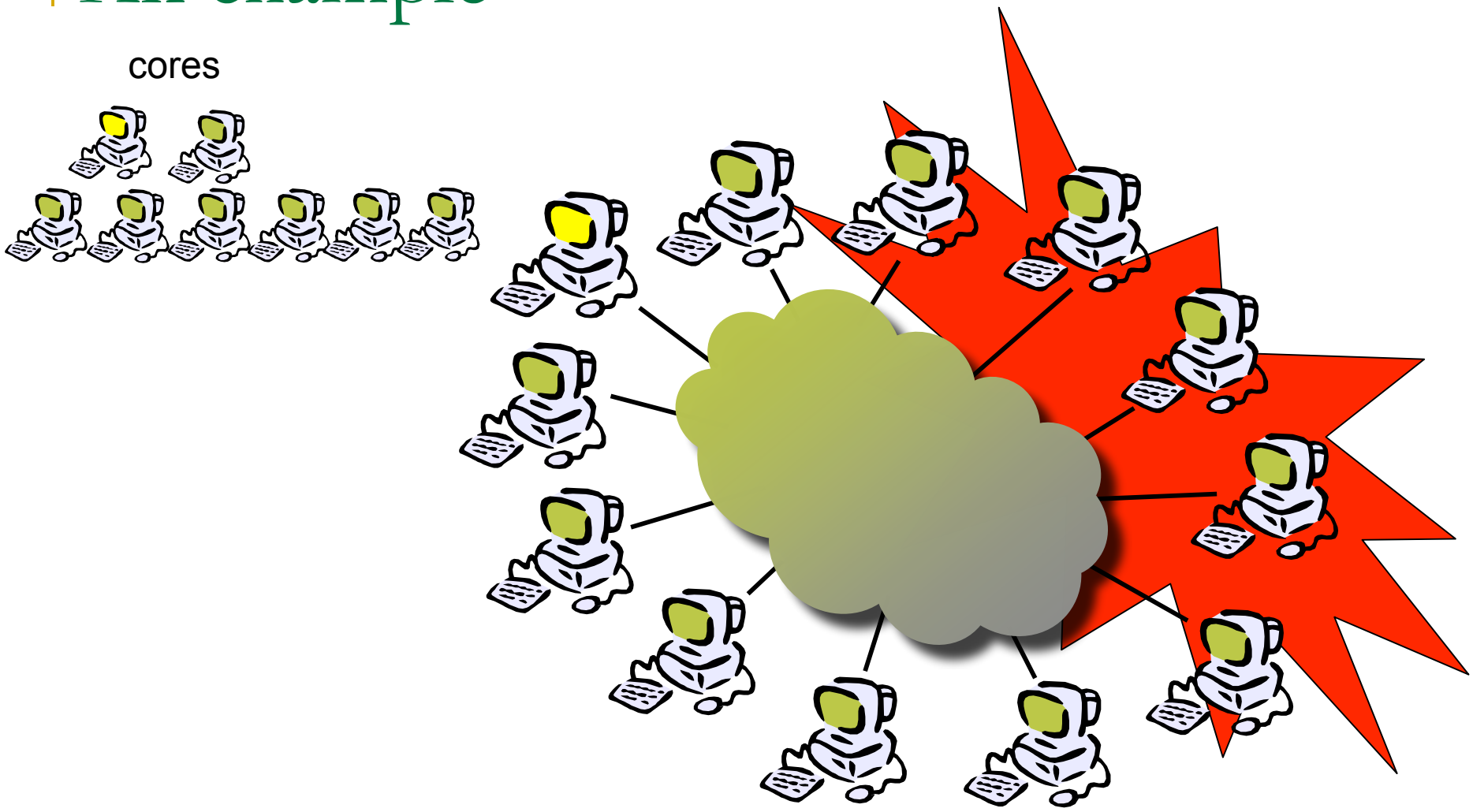
$n = 12$

An example



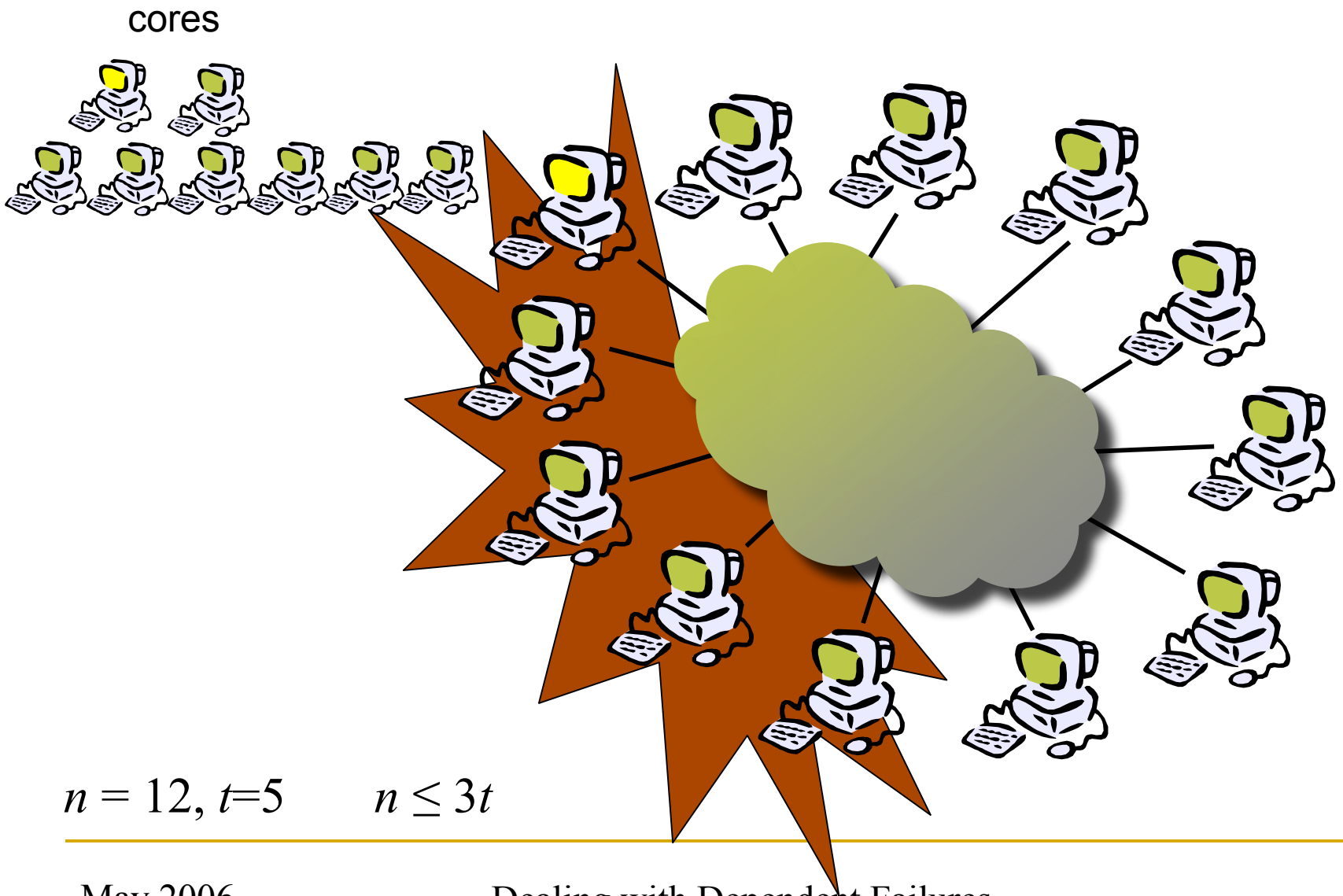
$n = 12$

An example



$n = 12, t=5$ $n \leq 3t$

An example



$n = 12, t=5$ $n \leq 3t$

Protocols for Dependent Failures

Wish a simple abstraction different from the threshold model that can represent non-IID failures.

- Rather than being a source of despair, can exploit knowledge about dependent failures.
- Understand the limitations (eg, lower bounds) when failures are not IID.

Modeling dependent failures

Dual representations

- *core c*: a minimal set of processes such that, in every run, there is some process in *c* that does not fail.
- *survivor set s*: a minimal set of processes such that there is a run in which no processes in *s* fails.
  - Each survivor set has a non-empty intersection with each core.

Can be formally defined either probabilistically or in terms of admissible runs.

... using cores is useful for *lower bound proofs* while using survivor sets is useful for *protocol design*.

Modeling dependent failures

Let R be the set of runs, π be the set of processes and $up(r)$ be the set of processes that do not fail during the run r.

$$C_\pi = \{c \subseteq \pi \mid (\forall\, r \in R\colon c \cap up(r) \neq \emptyset) \wedge (\forall\, c' \subset c\colon \neg\, \forall\, r \in R\colon c' \cap up(r) \neq \emptyset)\}$$

$$S_\pi = \{s \subseteq \pi \mid (\exists\, r \in R\colon s = up(r)) \wedge (\forall\, s' \subset s\colon \neg\, \exists\, r \in R\colon s' = up(r))\}$$

Modeling dependent failures

Under the threshold model:

- any set of $t + 1$ processes is a core;
- any set of $n - t$ processes is a survivor set.

Cores and survivor sets are duals of each other.

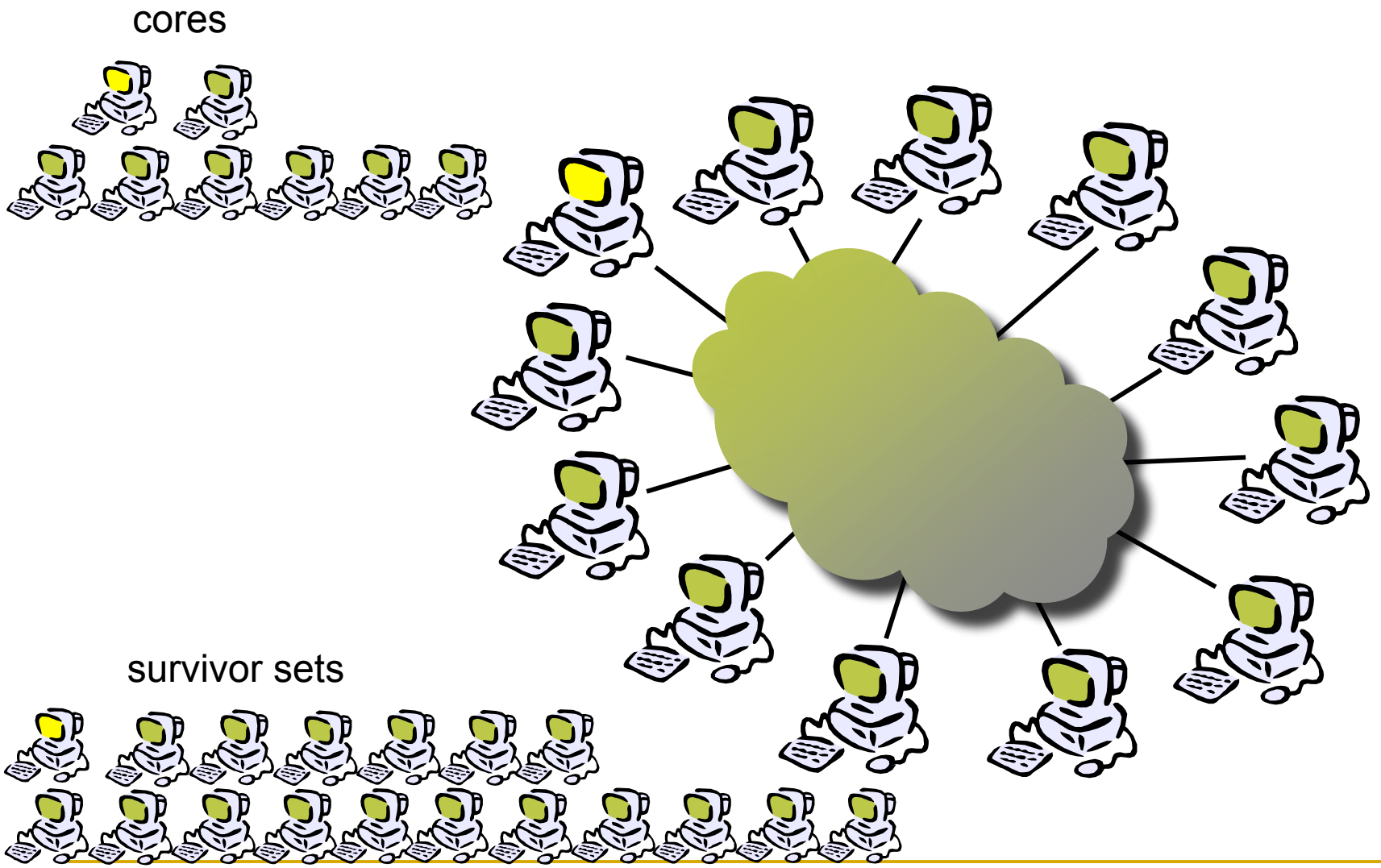
Eg, cores $\{a, b\}$, $\{a, d\}$

$x \equiv$ *process x does not crash*

$(a \vee b) \wedge (a \vee d)$ *vs.* $a \vee (b \wedge d)$

survivor sets $\{a\}$, $\{b, d\}$

An example

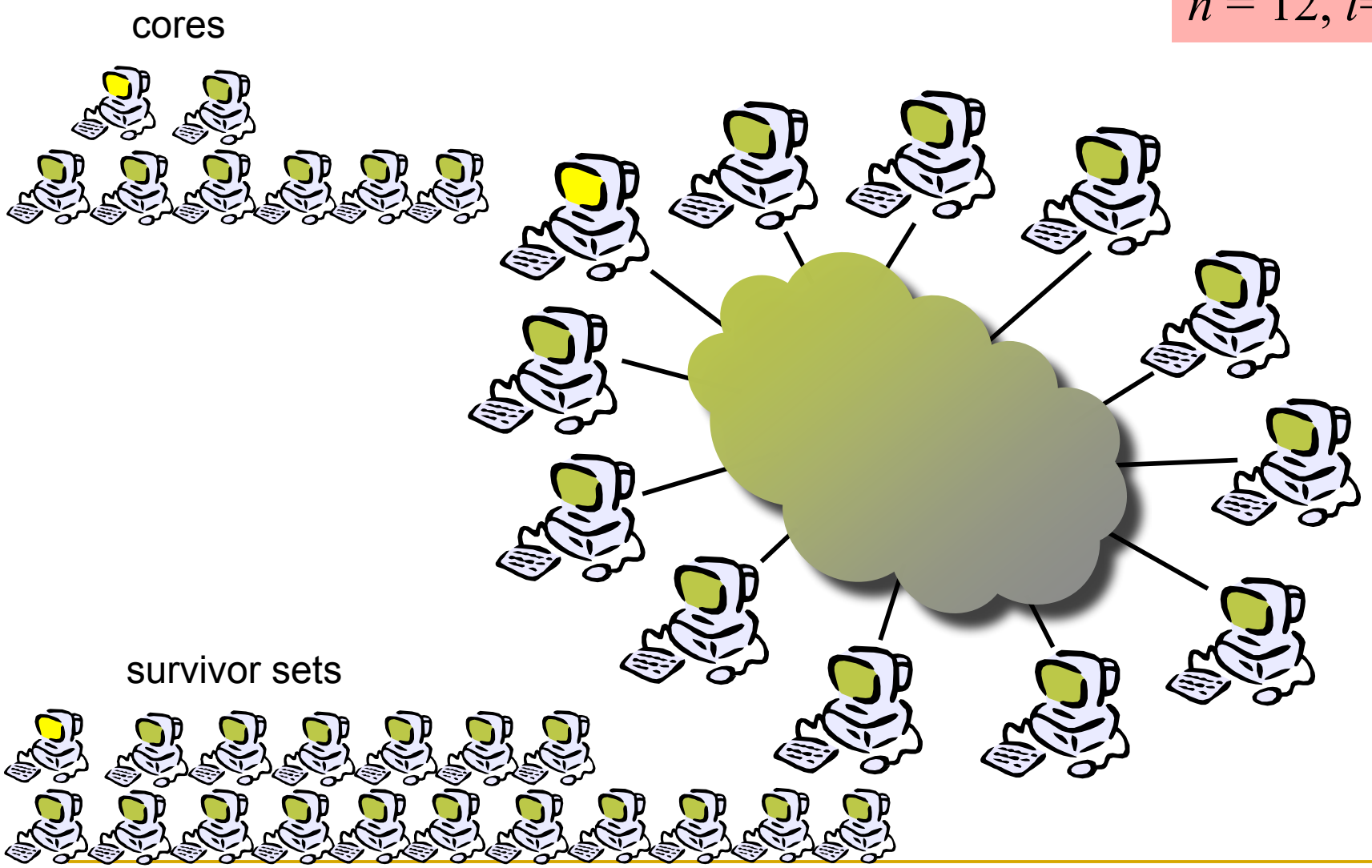


Replication Requirements

- Replication requirements of the form $n > k\,t$ becomes
  - Any partition of the processes into k subsets results in one of the subsets containing a core.
  - The intersection of any k survivor sets is not empty.
    - The intersection of any $k - 1$ survivor sets contains a core.

An Example

$n = 12, t=5$



Fractional Replication Requirements

$n > a\ t\ /\ b$

- There are some surprising consequences
  - ($a = 4, b = 2$) is not the same as ($a = 2, b = 1$)
- The survivor set properties aren't easily expressed in a general way
  - (n, $n - 1$) and (3, 2)

Consensus and Dependent Failures

- We have re-examined consensus protocols in this model.
  - The protocols are often easily derived from existing protocols
    - Synchronous crash consensus: decide in smallest core, with last round delivering result to all.
    - Asynchronous crash consensus: Paxos over quorum system with survivor sets as coterie rather than majority coterie system (which has optimal availability in IID)
  - The lower bound proofs are similar, but occasionally surprising
    - Time to decide in synchronous consensus.
      - Same in IID, but arbitrary slower than crash in general.

Generalizing

- We have translated, by hand, many protocols.
  - For a restricted class of protocols, this can be automated.
  - Can a generalized technique be developed?

Practical Use: Grid Systems

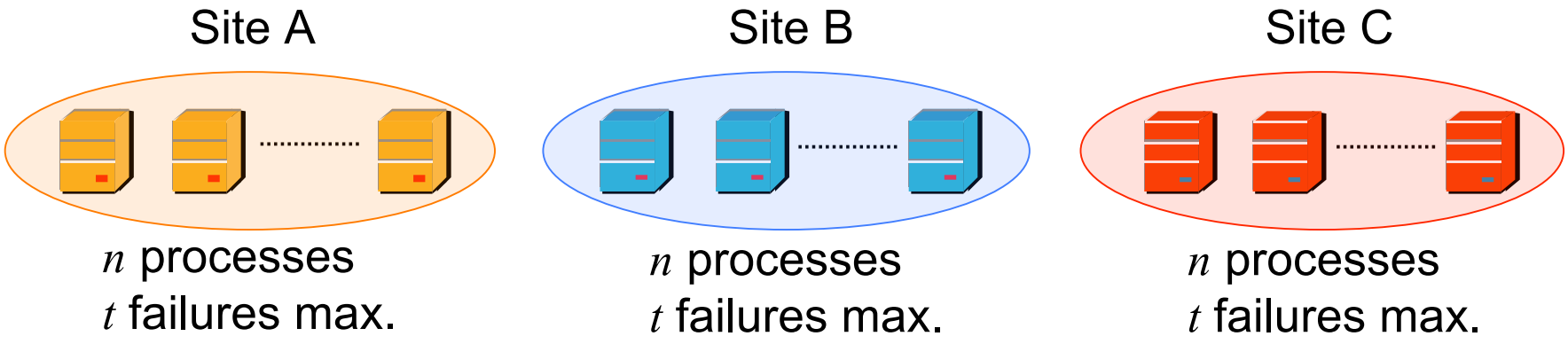
- In grid systems (GriPhyN, BIRN, Geon) the application for non-IID failures reduces to construction of coteries in a wide-area network.
  - Individual machines can fail
  - Whole sites can fail or can be disconnected

  ... in such an environment, simple majority quorums do not have optimal availability.

Multisite Failure Model

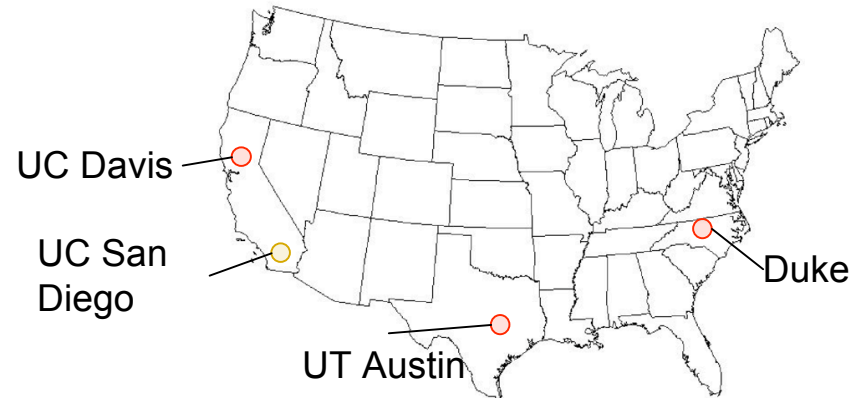
Consider case where failures within sites is IID and failures of sites is IID

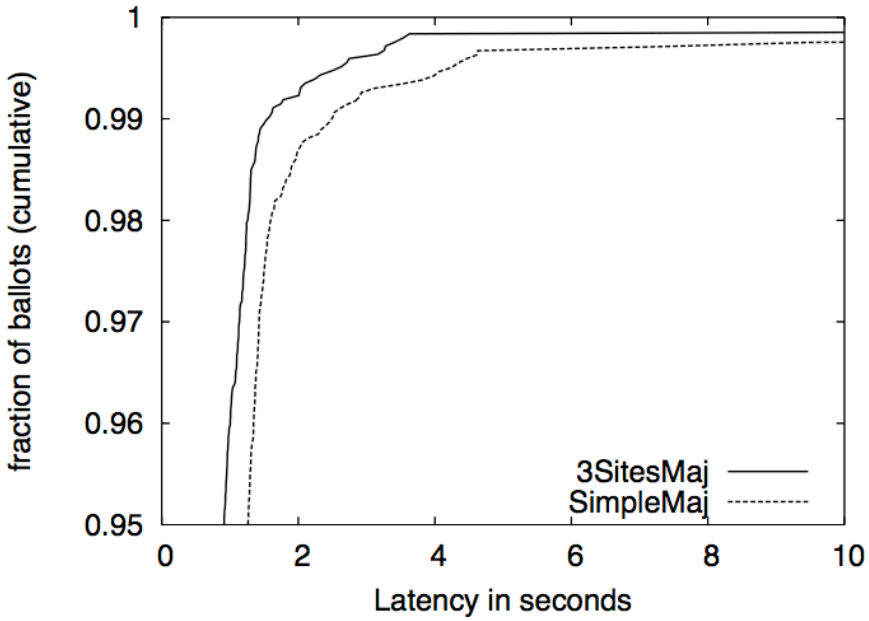
- At most t_s unavailable sites, and at most t unavailable within a site
- For example, $t_s = 1$
  - One site can be unavailable
  - Two sites: at least $n - t$ working properly
  - Survivor sets: $n - t$ processes from two different sites



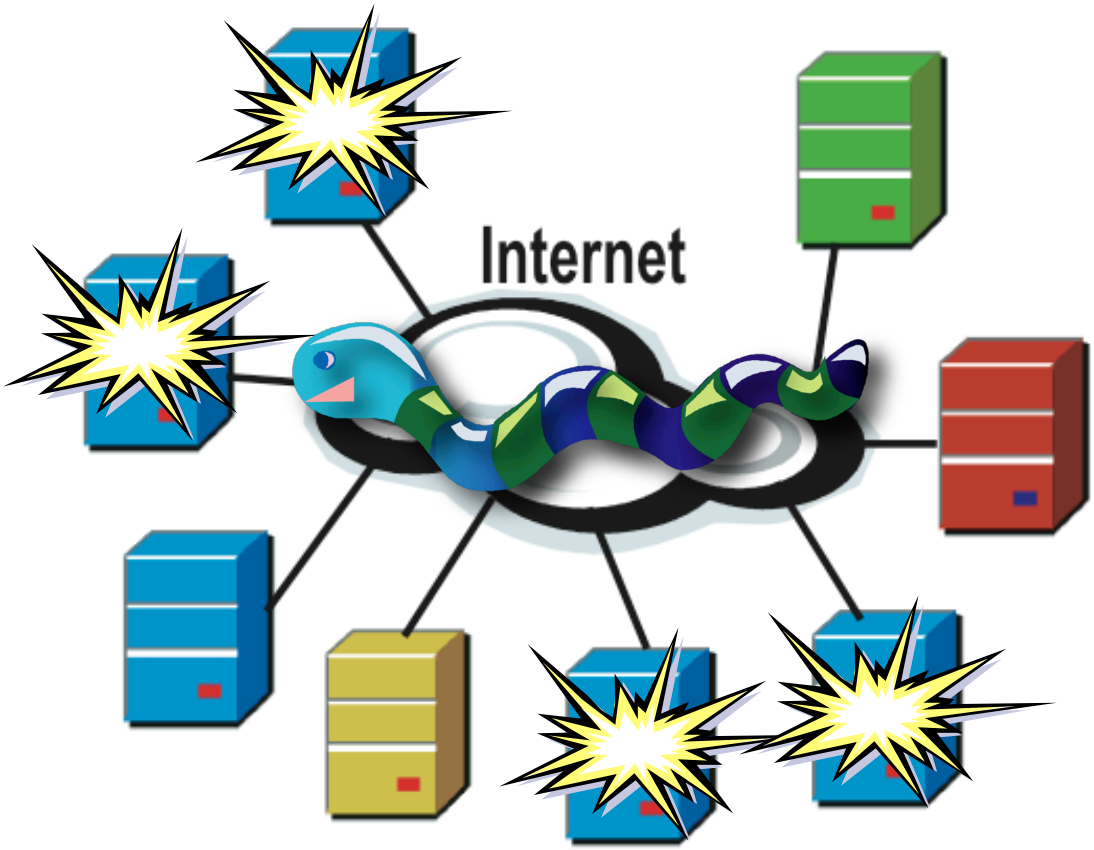
Quorums on PlanetLab

- Toy application: quorums of acceptors (Paxos)
  - Client: issues requests
    - Two access for each request (ballot)
  - Sites used
    - Three sites for the acceptors
    - Three acceptors from each site
    - One UCSD host: client
- Two ways of constructing quorums
  - SimpleMaj
    - Quorum: any five processes
  - 3SitesMaj
    - Quorum: four hosts, majority from each of two sites





Practical Use: Internet Catastrophes



- *Vulnerabilities shared among many hosts* allow an Internet pathogen to quickly spread.
- Can replication be used to preserve function in the face of such an attack?

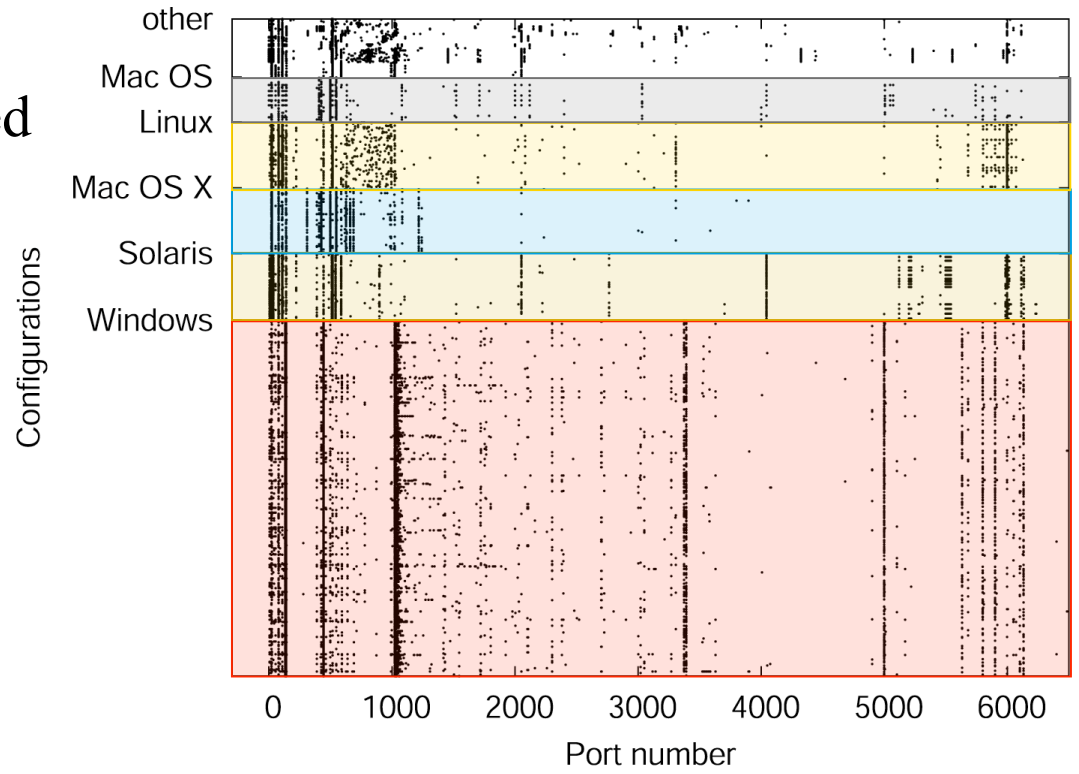
Replicating for Internet Catastrophes

Phoenix: Cooperative Backup

- Informed replication
  - Replica sets based on attributes
  - Different attributes indicate disjoint vulnerabilities
- Attributes
  - Common software systems
- Challenges
  - Is there sufficient diversity in an Internet setting?
  - Can we select small replica sets?

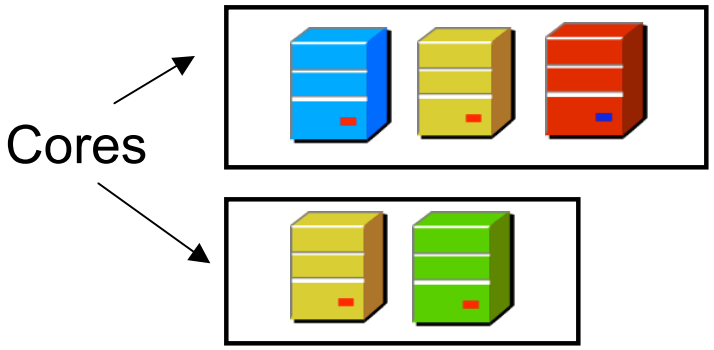
Host diversity

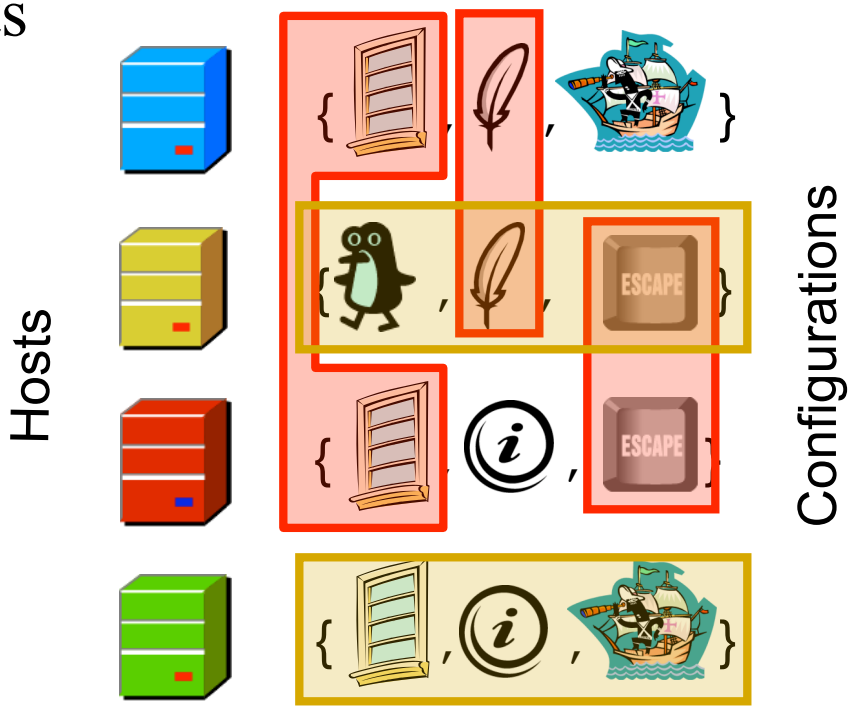
- Study of the UCSD network
- Vulnerability represented by open port
- *nmap* utility
  - Port scans
  - OS fingerprinting
- 2,963 general-purpose hosts (port data + OS)
- Host configuration
  - OS + Applications



Replica sets

- A set *S* of hosts is a core
  - No attribute in all the hosts
  - *S* is minimal



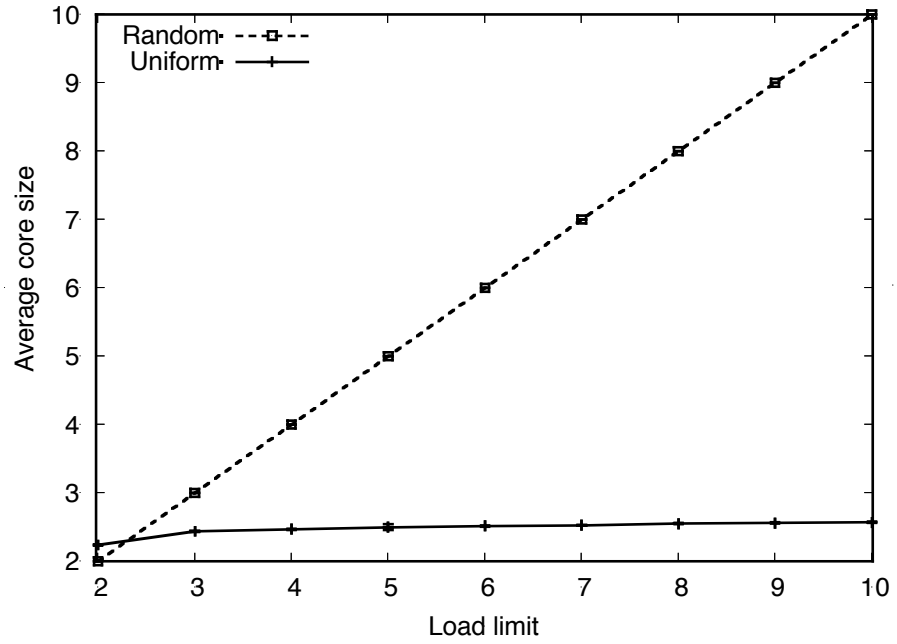


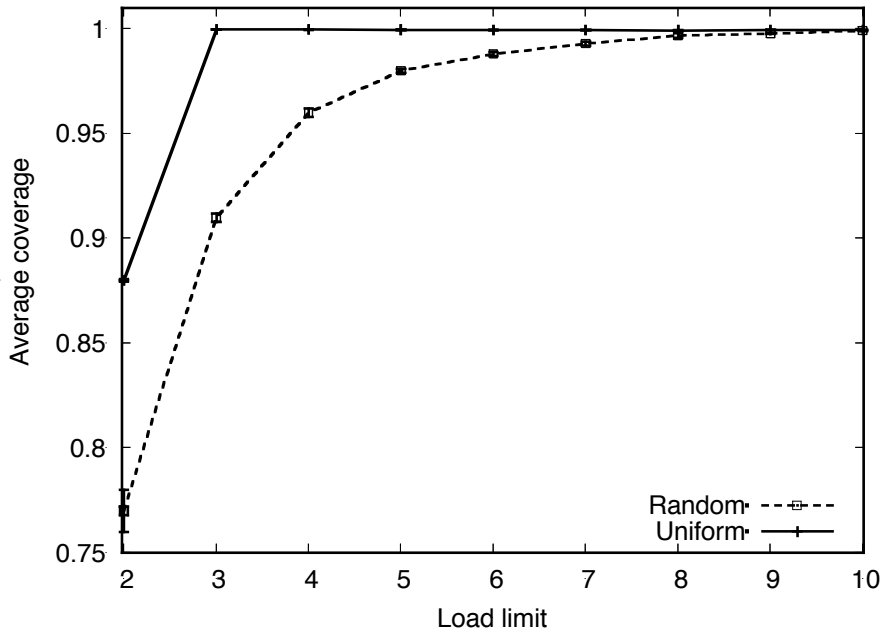
Heuristics for Selecting Cores

- Random selection
- Greedy selection

... with limits on how many cores a given host participates in.

Core size and Coverage



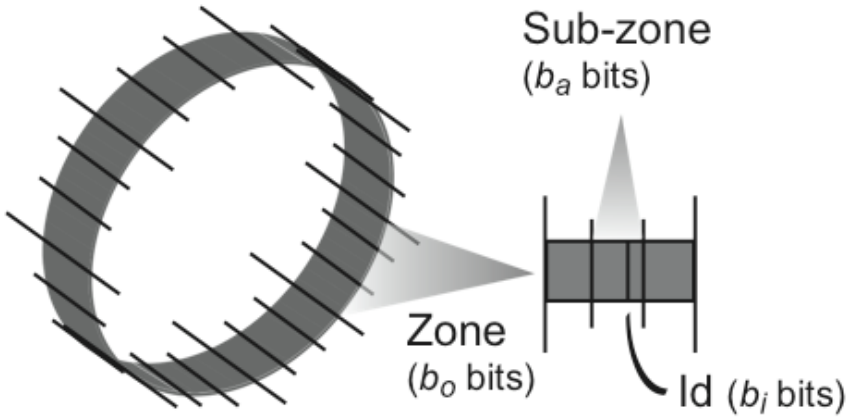


Uniform: most hosts require a single extra replica

Three times more storage for random

Phoenix Recovery System

- Data backup
  - On cores using Uniform
- Implement on a DHT
  - Pastry
  - Macedon framework
- Advertising configurations
  - Container → Zone
  - Sub-container → Sub-zone
- To get a core
  - Request messages
- To recover
  - Periodic announcement messages
  - Core members only



Prototype evaluation (USENIX '05)

- On PlanetLab
- Total number of hosts: 63
  - 62 PlanetLab hosts
  - 1 UCSD host
- Configurations manually set
  - 63 randomly chosen out of the 2,963
- Simulating a catastrophe
  - Failed Windows hosts
  - Recovery time
    - Announcement period: 120s
    - For 35: ~ 100s
    - One site: order of minutes

L	Core size		Coverage	
	Imp.	Sim.	Imp.	Sim.
3	2.12	2.22	1.0	1.0
5	2.10	2.23	1.0	1.0
7	2.10	2.22	1.0	1.0

Wrap up

Designing for non-IID failures is possible and worth doing.

- On the formal side, working on automated methods for transforming protocols from IID to non-IID.
- For grid computing, empirical study to see what failures occur in Geon and how replication can be done to increase availability.
- For Internet catastrophes, empirical study on the nature of actual vulnerabilities in an enterprise network.