Speculating Seriously

Rachid Guerraoui, EPFL

The World is turning IT

IT is turning distributed

Everybody should come to disc/podc

But some don't

Indeed theory
scares practitioners

But wait, there is more

We need be less conservative

We can do so and have fun

i.e., still do theory

This talk

So what do we do exactly?

As distributed computing community, we study agreement, renaming, concurrent objects, gossip, routing, etc

As theoreticians, we study complexity

Complexity in a centralized setting

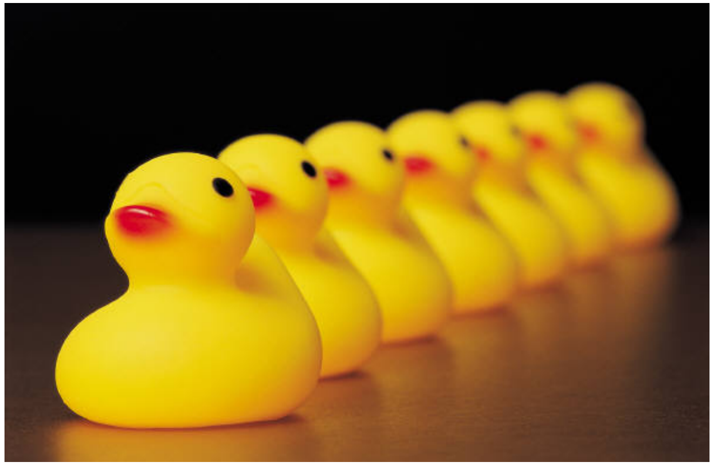
Number of cells/steps on a single tape Turing machine

Complexity in a distributed setting

We count the number of rounds/messages

In a given model…

The game



Processes

Adversary

Satan

Lucifer

Scheduler

Model

Model (set of runs)

Synchronous with t crashes

Model

Asynchronous with arbitrary failures

Centralized: C(P)

Distributed: C(P,M)

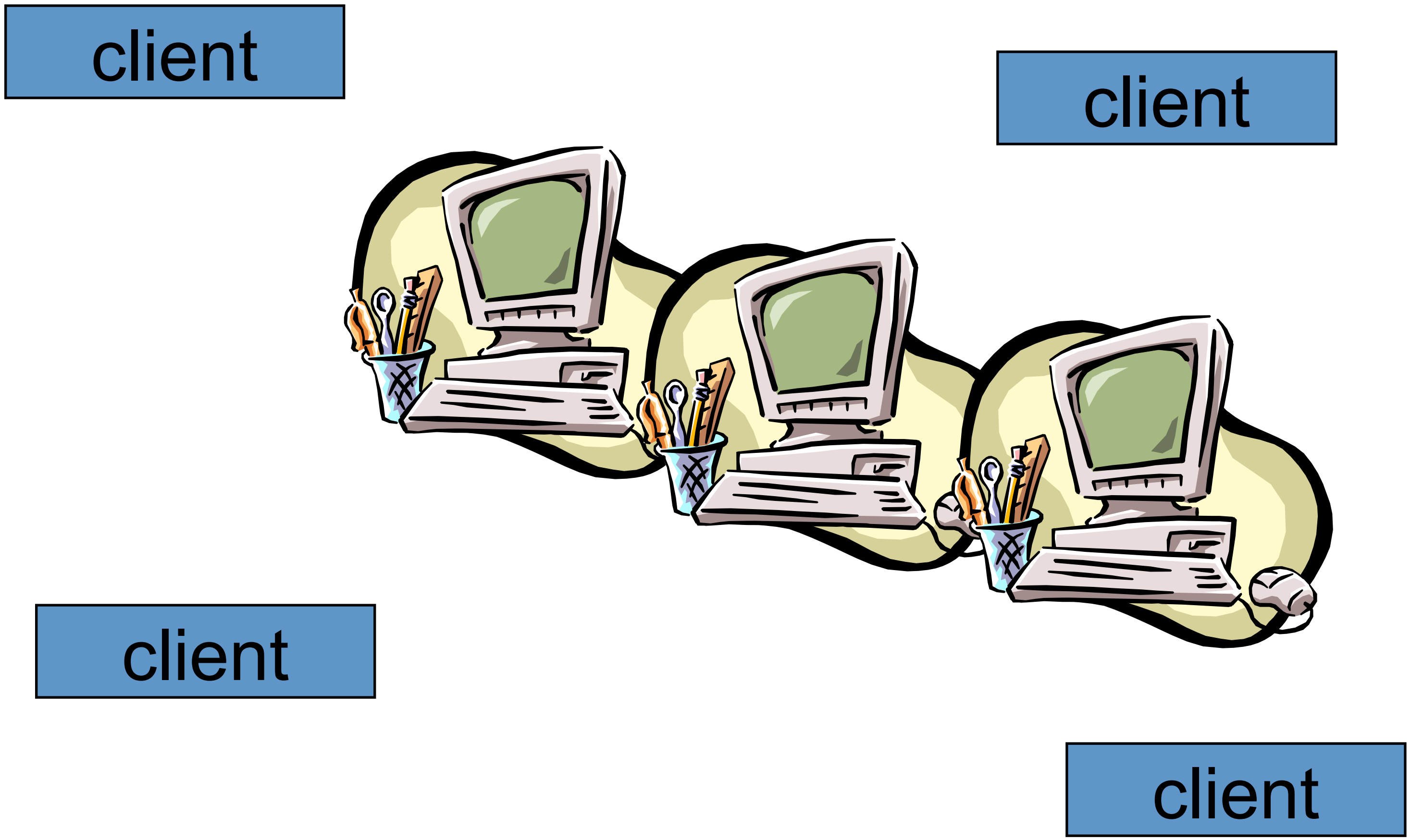
Example:
a highly available state machine

A robust Turing machine

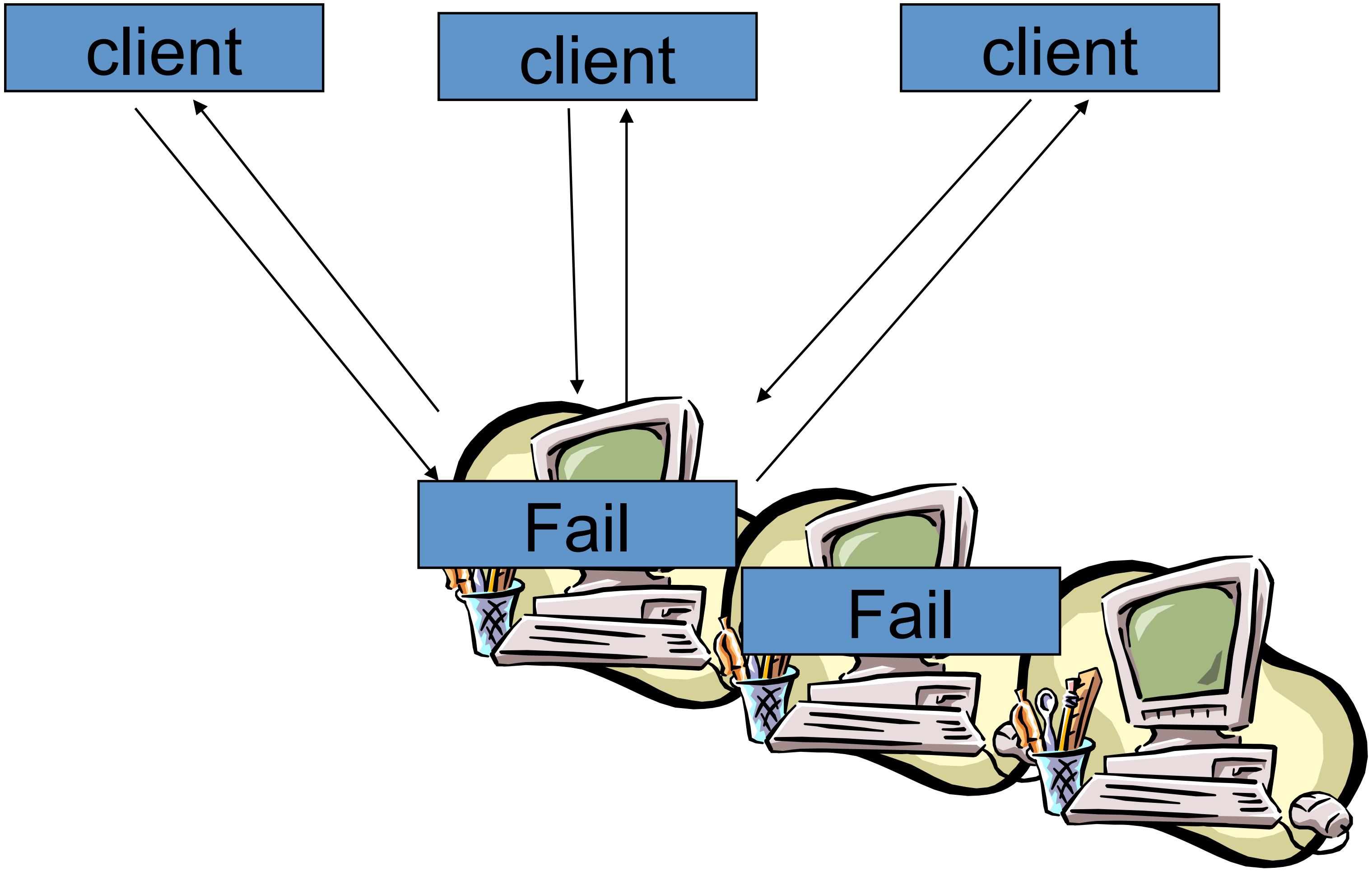
A universal construction

A data center

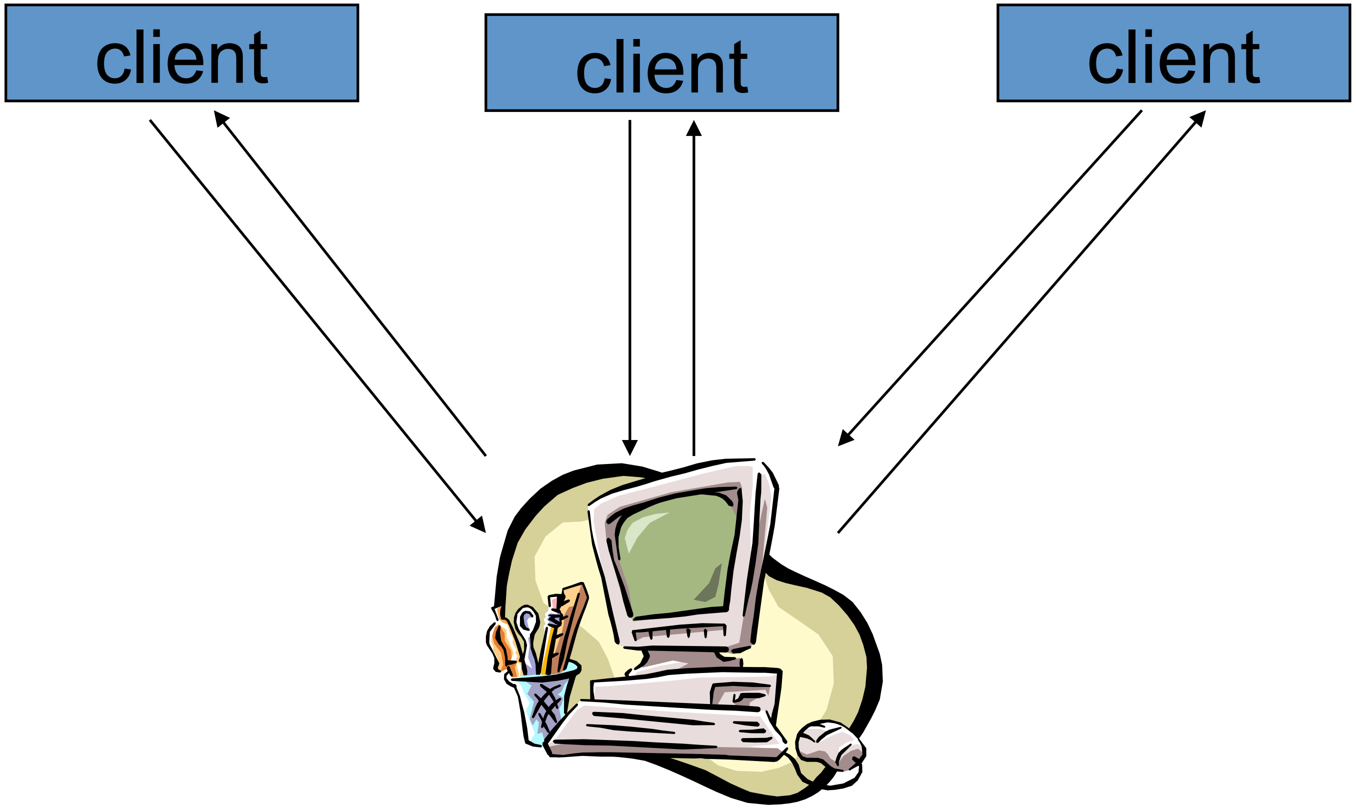
State Machine Replication



The Illusion of a robust server



The single server illusion

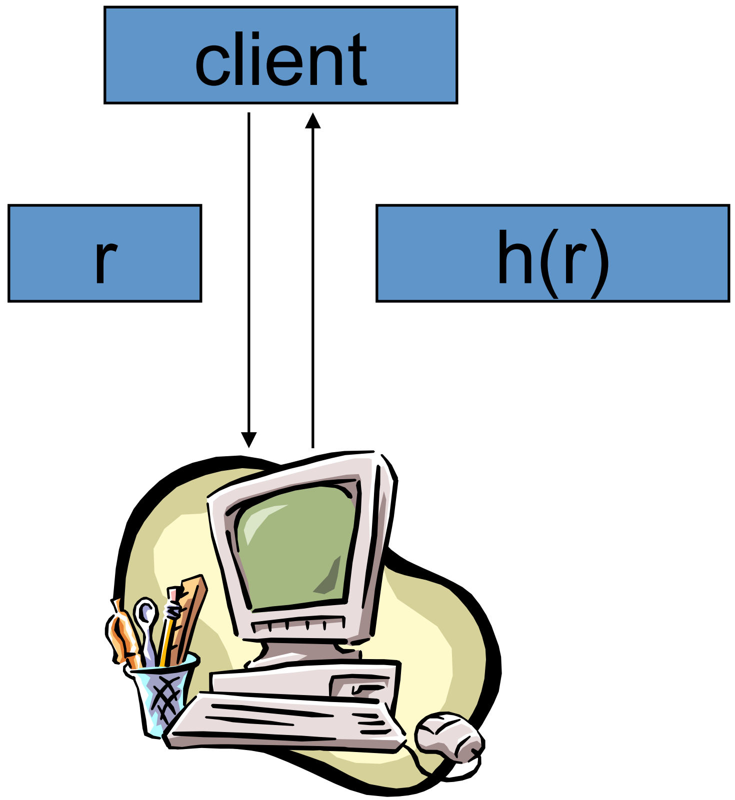


State machine

- The state of the server is modeled by a ***history*** of ***requests h***

- The client ***invokes*** request ***r*** on the server and gets back a ***history h*** of ***requests (reply);*** we say the client ***delivers h***

State machine



The single server illusion

- (**Ordering**) If c1 delivers a history h1 and c2 delivers a history h2, then either h1 is prefix of h2 or vice et versa

- (**Real-time**) If c1 delivers h1 before c2 invokes r, then h1 is prefix of h2(r)

- (**Validity**) In every delivered history, every request appears at most once and only if it was invoked by some client

The robust server illusion

- ***O,R,V (single server illusion)***

+

- (**HA**) If a correct client c invokes a request r, c eventually delivers a history h(r) including r

What complexity?

C(SMR,M) = X

Every SMR algorithm has a run of M where some (correct) client requires X rounds to get a reply

There is a SMR algorithm of which no (correct) client requires more than X rounds to get a reply

Model

Asynchronous with arbitrary failures

What complexity?

Orthodox answer

"Infinity"

What complexity?

Pragmatic answer

"1 round-trip"

Wrong?

The Fish does not think

The Fish doesn't need to think

The Fish knows

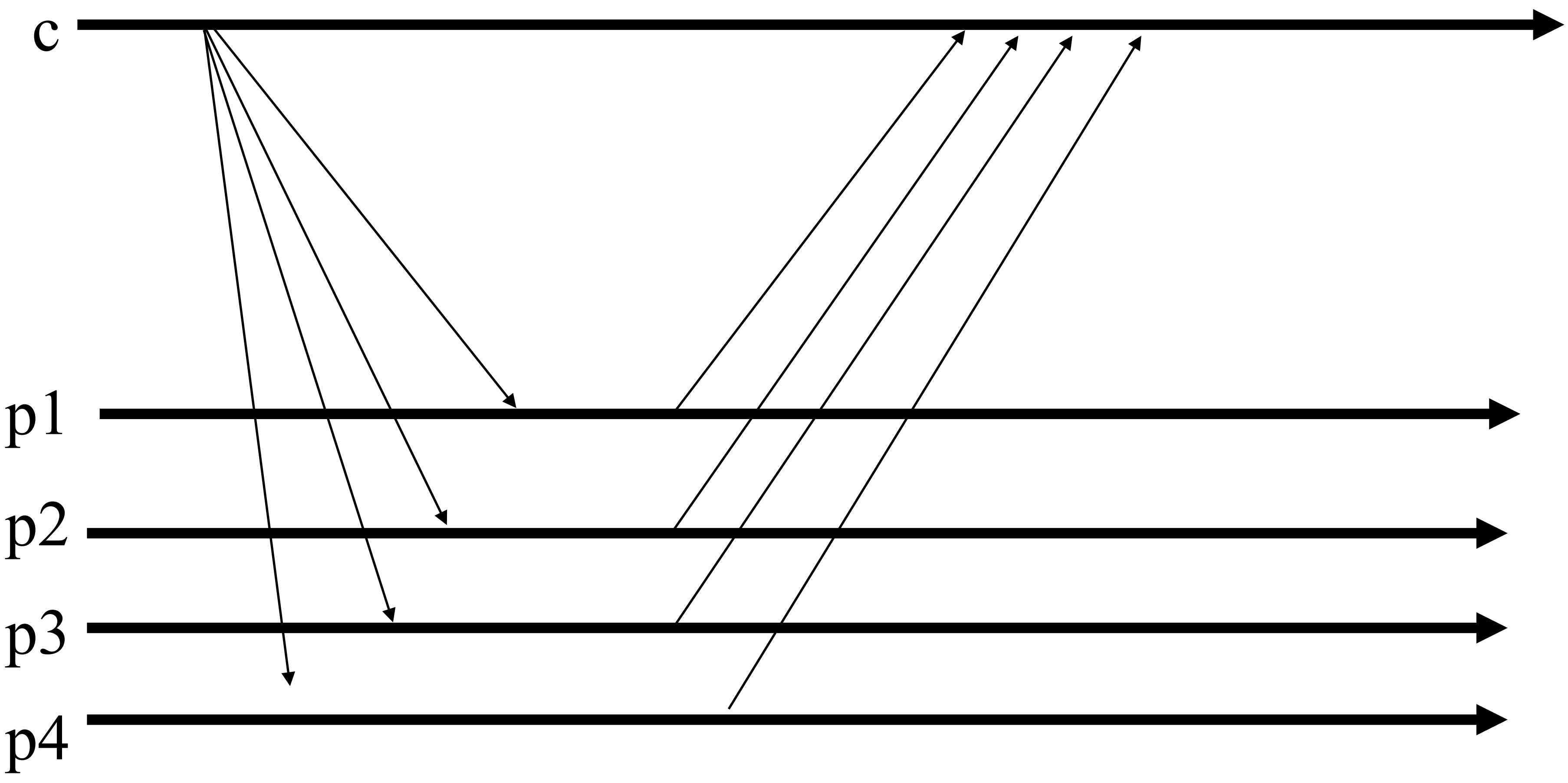
Iggy Pop

Complexity?

There is an algorithm that returns a reply after 1 round-trip

When the system is synchronous, failure-free and contention free

Quorum (GKQV10)



Model?

Synchronous, failure and contention-free

Complexity?

What if there is contention?

"2 round-trips"

Complexity?

What if there are t failures?

“t + 2 rounds”

Complexity?

Does the system need to be synchronous?

“**few rounds of synchrony are enough**”

Complexity?

What if the system is really asynchronous?

“**infinity**”

Orthodox: “**now we are talking**”

What is really going on?

Speculations…

Plan for the worst
Optimize for the common

What is the common?

- Synchrony and no failure
- Synchrony, no failure or contention
- Synchrony, no failure, little contention
- Synchrony, no failure, high contention,
- Synchrony, no failure, little contention, small requests
- ....

SMR Algorithms

- PBFT [OSDI'99, SOSP'01]
- Q/U [SOSP'05]
- HQ [OSDI'06]
- Zyzzyva [SOSP'07]
- Mencius [OSDI'08]

…

Getting each protocol to really work is a Dantean task

- 30.000 lines of non-trivial C code
- Manual proof is a nightmare
- Model checking is impossible

Beyond SMR

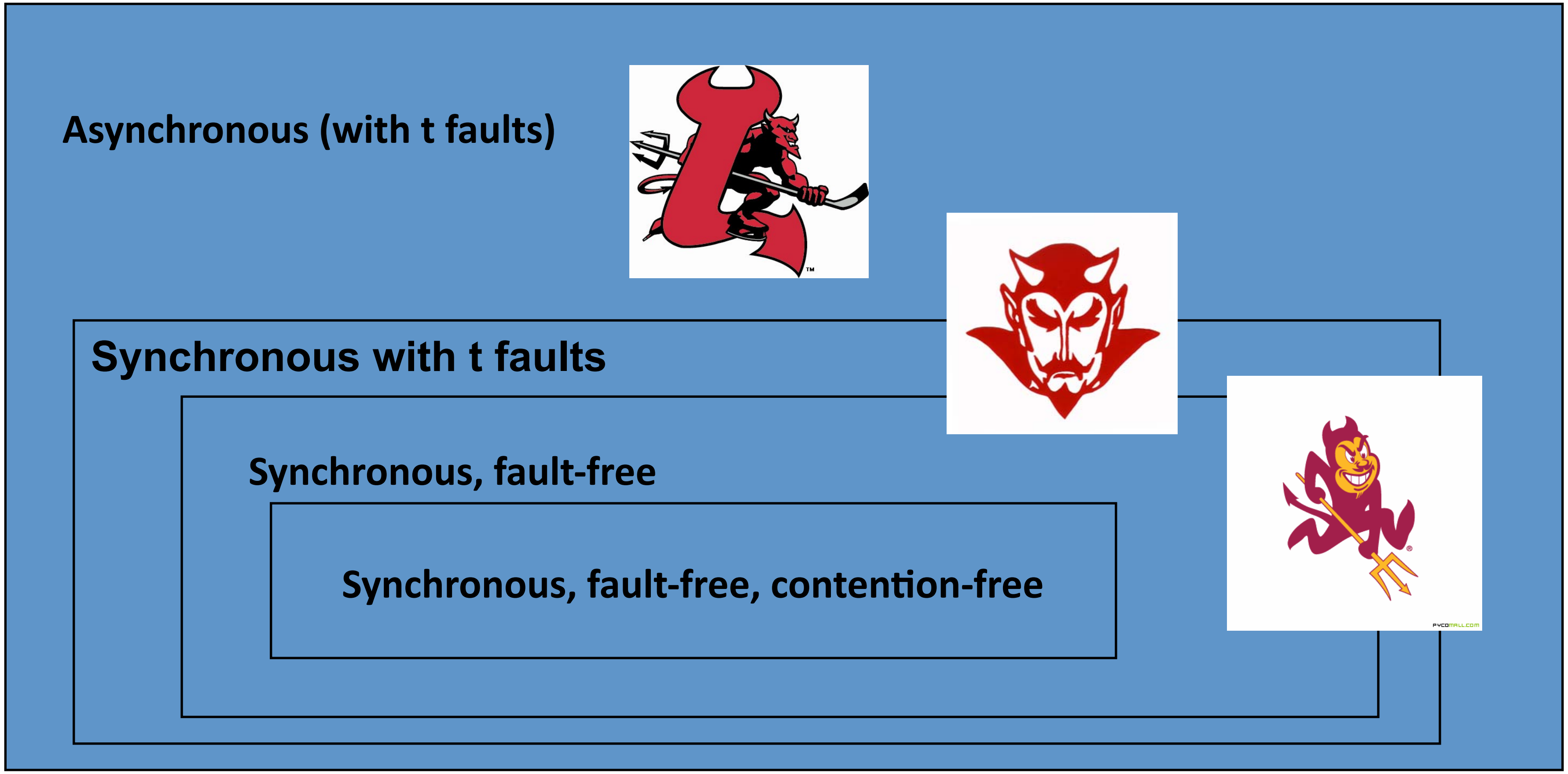
- Concurrent object implementations
- Transactional memory
- Sensor networks

Wanted

Theoretical foundations

What is really going on?

Hierarchical complexity



Complexity in distributed computing

Speculative

C(P,M) vs C(P,M1,M2,….M)

Speculative algorithm

$$C(A/P, M1, M2, \ldots, Mn\text{-}1, M) = (c1, c2, .., cn)$$

$$c1 < c2 < c3, .. < cn1 < cn$$

How to prove speculative lower bounds

$$C(P,M1,M2,\ldots,Mn-1,M) = ?$$

How to write/prove speculative algorithms?

I have a dream

```
Switch(model)
  Case M1: speculation1();
  Case M2: speculation2();
  Case M3: speculation3();

  ….

  Case M: conservative();
```

ABSTRACT

(Abortable state machine replication)

- A ***SMR*** abstraction that can either:
  - ***Commit*** a request (as in SMR)
  - ***Or***
  - ***Abort*** a request and return a (unforgable digest of request) history to invoke another Abstract instance
  - The conditions under which Abstract can ***abort*** define a specific ***instance***

Abstract examples

- Abort is allowed only in case of asynchrony
- Abort is allowed only in case of contention or asynchrony
- Abort is allowed only in case of asynchrony and high-contention
- Abort is allowed only in case of asynchrony or k failures
- Abort is allowed only after committing k requests
- Abort is never allowed

Abstract properties

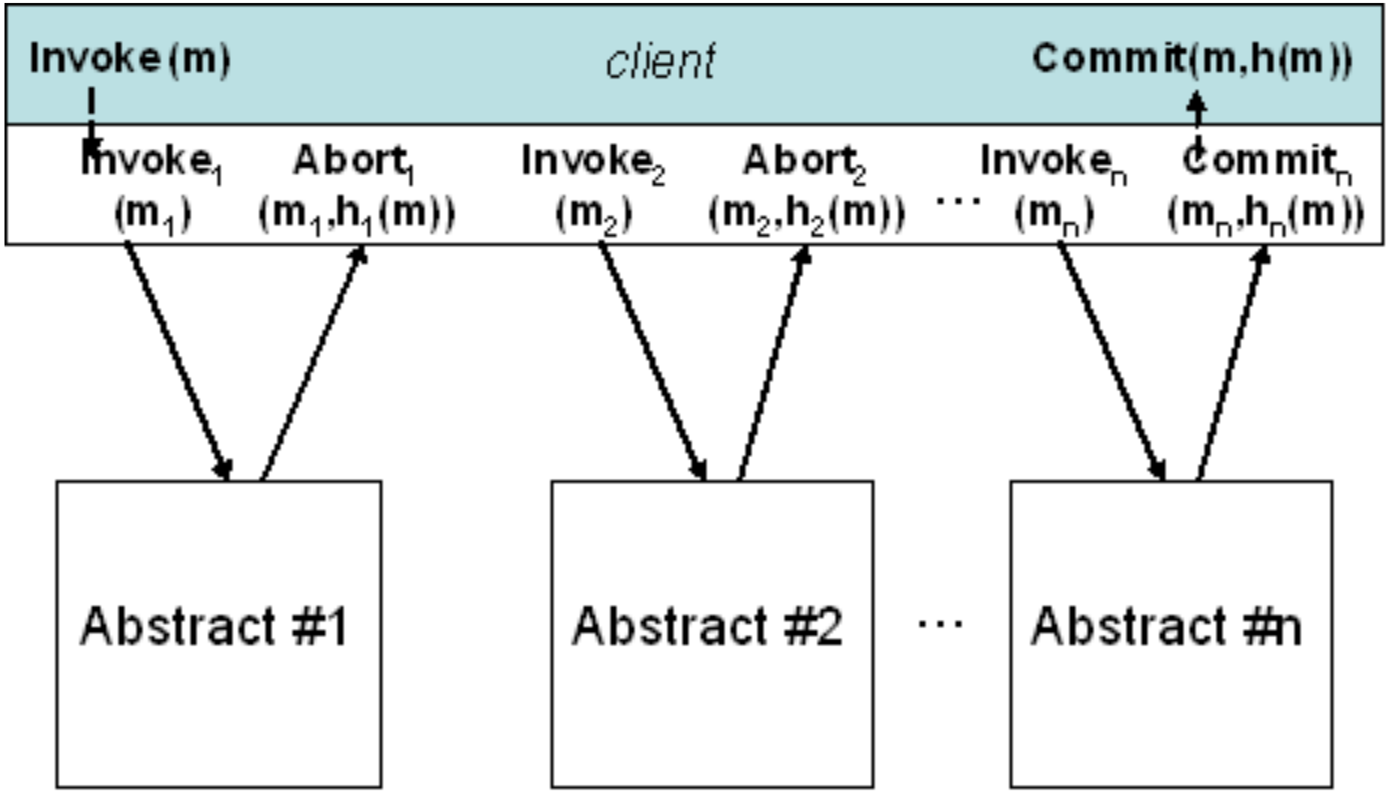
- ***O-C***: If histories h(r1) and h(r2) are ***committed***, then one is the prefix of the other

O-A: If history h(r1) is ***committed*** and history h(r2) is ***aborted***, then h(r1) is prefix of h(r2)

Abstract initialization

- ***Init requests*** are made of a request and a history
- ***Initialization property***: any common prefix of init histories is a ***prefix*** of any committed or aborted history

Abstract compositions



m_{i+1} =<INIT, m, h_i(m)> (i>1)

$m_1 = m$

Aliph

- Uses 4 instances
  - **Quorum**: 30% reduced latency when no time-out or contention,8% of PBFT code
  - **ZyzyLight:** 100% improved throughput when no time-out and little contention, 15% PBFT code
  - **Chain**: achieves up to 400% improvedpeak throughput when no time-out and high contention
  - **mPBFT**: commits at least m requests

We need be less conservative

We can do so and have fun

i.e., still do theory

This talk

Beyond SMR

- Concurrent object implementations
- Transactional memory
- Sensor networks

Thank you for your attention

Example : AQuorum

- < 4000 lines of code
- Decentralized approach (« quorum »)
- Outperforms all BFT protocols we know of in terms of latency
- Model checked in +Cal

Cost of switching

- 54ms with a history log of 32 requests
- 147 ms with a history log of 100 requests

- Request and reply of 8 bytes

- NB. In this case, the best-case latency is 1ms