Introduction

Companion slides for
Distributed Computing
Through Combinatorial Topology
Maurice Herlihy & Dmitry Kozlov & Sergio Rajsbaum
In the Beginning …

a computer was just a Turing machine …
Computing is co-ordination and communication
No, distributed computations are static mathematical objects!

Operational versus combinatorial approaches ...

Distributed computations unfold in time!
Road Map

Distributed Computing

Two Classic Distributed Problems

The Muddy Children

Coordinated Attack
Road Map

Distributed Computing

Two Classic Distributed Problems

The Muddy Children

Coordinated Attack
There are Many Models
There are Many Models

Communication?
There are Many Models

Communication?

Failures?
There are Many Models

Communication?

Failures?

Timing?

Combinatorial Topology
Communication

Message-Passing

http://commons.wikimedia.org/wiki/File:Pennyblack-pd.jpg
Communication

Message-Passing

Read-Write Memory
Communication

Message-Passing

Read-Write Memory

Black-Box Memory
Message-Passing

Prof. James Moriarty
Brown University
Providence RI 02912

Mr. S. Holmes
221B Baker Street
London NW1 6XE
Read-Write Memory

http://commons.wikimedia.org/wiki/File:RosettaStoneDetail.jpg

Read-Write Models

write & read individual locations

write & take memory snapshot

Group writes together then takes snapshots together
Layered Read-Write Memory
Black Box Memory

Compare-and-swap

Fetch-and-add
Failures

Crash failures: processes halt

How many?

Which ones?
Wait-Free Failure Model

All but one may crash
$t$-Resilient Failure Model

At most $t$ may crash

Here, $t = 1$. 
Correlated Failures

Processes on same server may crash
Adversaries
Adversaries

Determine which sets of processes can halt.
Adversaries

Determine which sets of processes can halt.

“worst-case” scenario
Timing Models
Timing Models

Processes share a clock
Processes share a clock

Synchronous
Timing Models

Processes share a clock

Synchronous

Processes do not share a clock
Timing Models

Processes share a clock

Synchronous

Processes do not share a clock

Asynchronous
Timing Models

Processes share a clock

Synchronous

Processes do not share a clock

Asynchronous

Processes have approximately-synchronized clocks
Timing Models

- Processes share a clock: **Synchronous**
- Processes do not share a clock: **Asynchronous**
- Processes have approximately-synchronized clocks: **Semi-synchronous**
Synchronous
Synchronous Failures

detection easy
Asynchronous
Asynchronous Failures

detection impossible
Semi-Synchronous
Semi-Synchronous Failures
Computation Model Space

Which combinations make sense?
Multicores

Asynchronous
Wait-free
Shared Memory

http://www.bit-tech.net/hardware/cpus/2011/01/03/intel-sandy-bridge-review/
Distributed Computing

Asynchronous Message-passing

Internet

Sensor network

http://commons.wikimedia.org/wiki/File:Internet_hosts.PNG

http://commons.wikimedia.org/wiki/File:XBee_Series_2_with_Whip_Antenna.jpg
Parallel Computing

Synchronous Message-passing (or shared memory)

GPU

http://commons.wikimedia.org/wiki/File:Geforce2_mx400_gpu.JPG
Local Views

Each process has a 3-bit local view
Multiple Local Views

local views differ by 1 bit
Multiple Local Views

local views differ by 1 bit

but no process knows which one
Multiple Local Views

each view is represented by a \textit{labeled vertex}
Global States

compatible views represented by an edge
Distributed Computing through Combinatorial Topology

All possible global states
(where blue has 111)
Communication

Each process sends local view to the other.
Each process sends local view to the other but at most one message may be lost!
One Communication Round

110

111

110

111

111

?
One Communication Round

110

111

110

111

111

??

1 Lost

Distributed Computing though Combinatorial Topology
One Communication Round

110
?

111
110

110
111

111
?

1 Lost

none Lost
One Communication Round
Distributed Computing though Combinatorial Topology

All possible global states after one round unreliable communication
Informally ….

Unreliable communication does not change “topology” of global states
Reliable Communication?

110
?

111
110

110

111
?

Distributed Computing though Combinatorial Topology
Reliable Communication?

\[ \begin{array}{cc}
111 & 110 \\
110 & 111 \\
\end{array} \]
Distributed Computing though Combinatorial Topology
Tasks
Tasks

Possible set of *input* values
Tasks

Possible set of *input* values

Finite computation
Tasks

Possible set of *input* values

Finite computation

Possible set of *output* values
Road Map

Distributed Computing

Two Classic Distributed Problems

The Muddy Children

Coordinated Attack
Muddy Children

11:00
Muddy Children

Distributed Computing through Combinatorial Topology
Muddy Children

At least one of you is dirty!

Combinatorial Topology
Muddy Children

You may not communicate!

Combinatorial Topology
When you realize you are dirty, confess on the hour!

Muddy Children
Muddy Children

(silence …)
Muddy Children

2:00

Me!

Me!

Me!
Operational Explanation
Operational Explanation

Others are clean, so I must be dirty.
Others are clean, so I must be dirty.

Me!
Operational Explanation
Operational Explanation

He was quiet, so I must be dirty.

He was quiet, so I must be dirty.
Combinatorial Explanation
Combinatorial Explanation

Distributed Computing through Combinatorial Topology
Each process has its own input
Combinatorial Topology

Each process has its own input

Distributed Computing through
Combinatorial Topology
Combinatorial Explanation
Distributed Computing through Combinatorial Topology
Distributed Computing through Combinatorial Topology

all dirty
Distributed Computing though Combinatorial Topology

all dirty

1:01
Distributed Computing though Combinatorial Topology

all dirty
Road Map

Distributed Computing

Two Classic Distributed Problems

The Muddy Children

Coordinated Attack
Coordinated Attack

Red army wins
If both sides attack together
The Two Generals

Red generals send messengers across the valley
The Two Generals

Messengers don’t always make it

Distributed Computing through Combinatorial Topology
Your Mission

Design a protocol to ensure that Alice and Bob attack simultaneously
Theorem

There is no protocol that ensures that the Red armies attack simultaneously
Operational Proof

Suppose Bob receives a message at 1:00 saying “attack at Dawn”.

Distributed Computing though Combinatorial Topology
Operational Proof

Suppose Bob receives a message at 1:00 saying “attack at Dawn”. Are we done?
Suppose Bob receives a message at 1:00 saying “attack at Dawn”. Are we done? No, because Alice doesn’t know if Bob got that message …
Operational Proof

So Bob sends an acknowledgment to Alice

...
Operational Proof

So Bob sends an acknowledgment to Alice...

Are we done?
Operational Proof

So Bob sends an acknowledgment to Alice ...

Are we done?

No, because Bob doesn’t know if Alice got that message …
Operational Proof

Bob sends an acknowledgment to Alice.

This goes on forever ... so no protocol is possible!

Are we done?

No, because Bob doesn't know if Alice got that message ...
Noon

Attack at dawn!

Attack at noon!

Bob is

Alice is

Distributed Computing though
Combinatorial Topology
Distributed Computing though Combinatorial Topology
These edges go here

delivered

lost

delivered

protocol graph

output graph

dawn!

noon!
This graph is *connected*
This graph is *not connected*
Map not allowed to “tear” protocol complex
protocol graph
decision
map
output graph
Operational Reasoning

http://commons.wikimedia.org/wiki/File:Professor_Lucifer_Butts.gif
Combinatorial Reasoning
Combinatorial Reasoning

Model-independent properties …
Combinatorial Reasoning

Model-independent properties …

… restricted model-dependent reasoning
This work is licensed under a Creative Commons Attribution-ShareAlike 2.5 License.

- You are free:
  - to Share — to copy, distribute and transmit the work
  - to Remix — to adapt the work
- Under the following conditions:
  - Attribution. You must attribute the work to “Distributed Computing Through Combinatorial Topology” (but not in any way that suggests that the authors endorse you or your use of the work).
  - Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.
- For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to
  - http://creativecommons.org/licenses/by-sa/3.0/.
- Any of the above conditions can be waived if you get permission from the copyright holder.
- Nothing in this license impairs or restricts the author's moral rights.