Reactive Systems: The Why and the How

Dean Wampler, Ph.D.
Typesafe

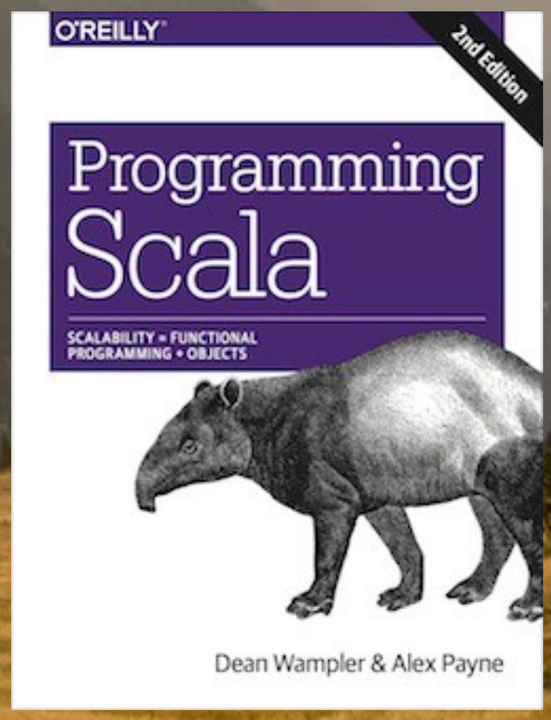
Typesafe

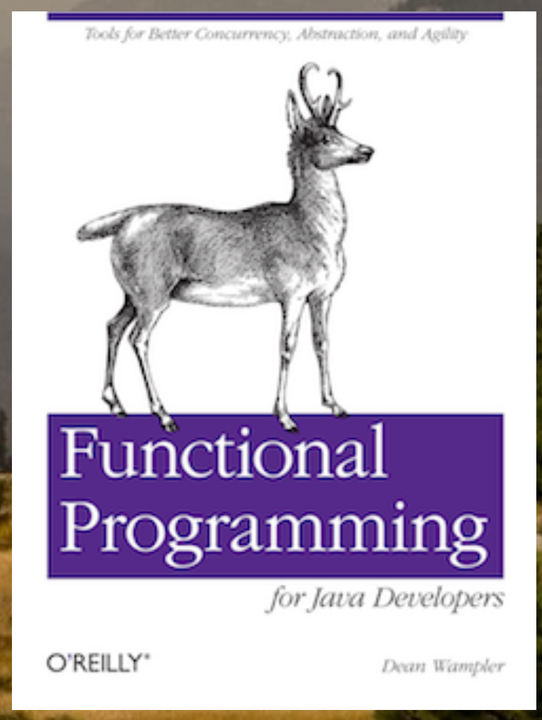
O'REILLY®
Software Architecture CONFERENCE
MARCH 16–19, 2015
BOSTON, MA

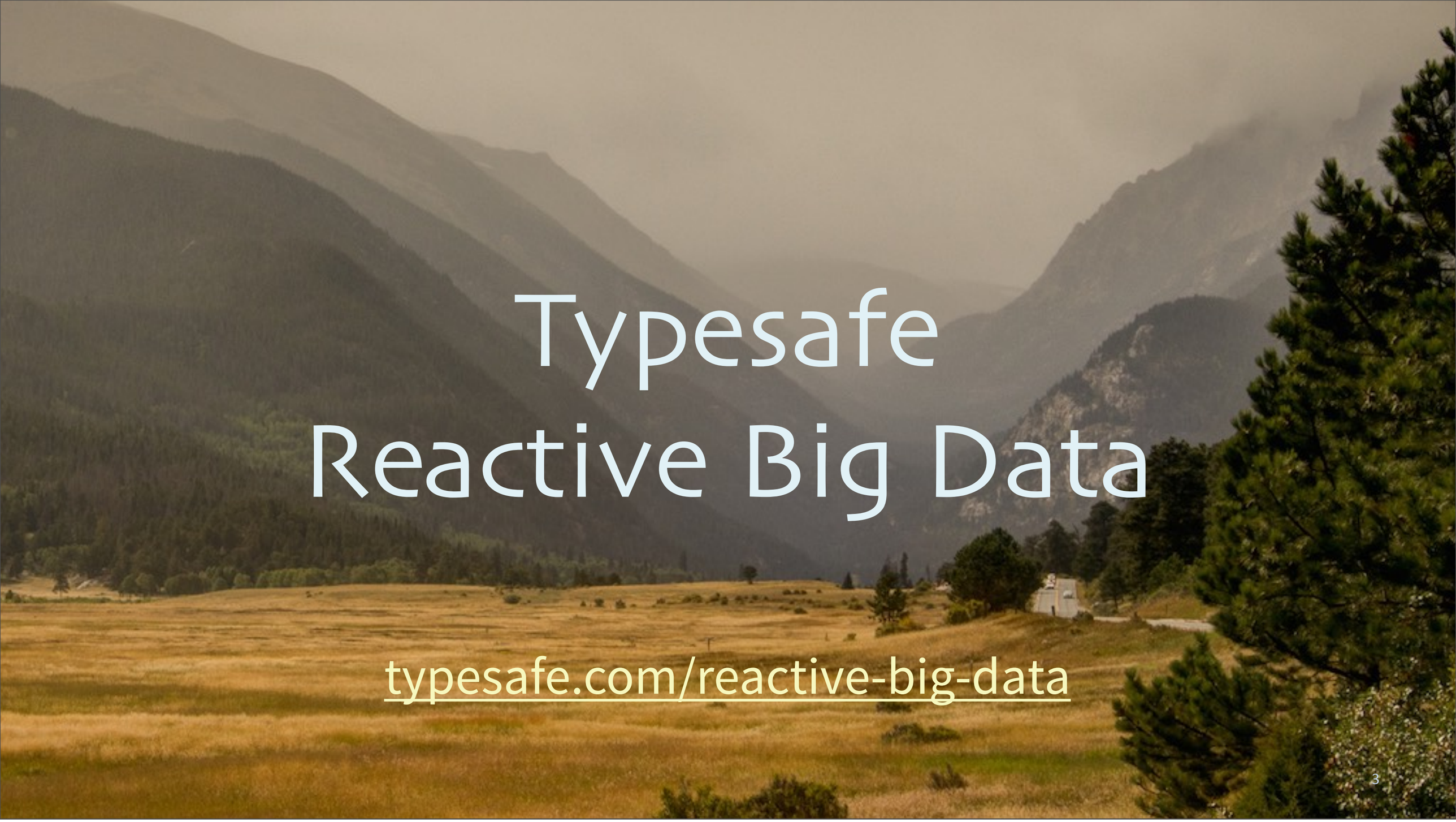
Tuesday, March 17, 15

Photos from Colorado, Sept. 2014.

dean.wampler@typesafe.com
polyglotprogramming.com/talks
@deanwampler







Tuesday, March 17, 15

This is my role. We’re now rolling out commercial support for Spark in non-Hadoop environments. We have other projects in the works. Talk to me if you’re interested in what we’re doing.

Later today:

Error Handling in Reactive Systems

3:30pm–5:00pm Thursday, 03/19/2015
Reactive and its variants
Location: 304
Tags: reactive

Dean Wampler (Typesafe)

The Reactive Manifesto's "Resilient" trait requires a system to stay responsive when failures happen. I'll discuss how real-world systems do this, starting with in-process techniques in Go, Clojure's core.async, and Reactive Extensions. Next, I'll discuss how some tools prevent common failures in the first place. I'll finish with the Actor model's strategic use of supervisor hierarchies. Read more.

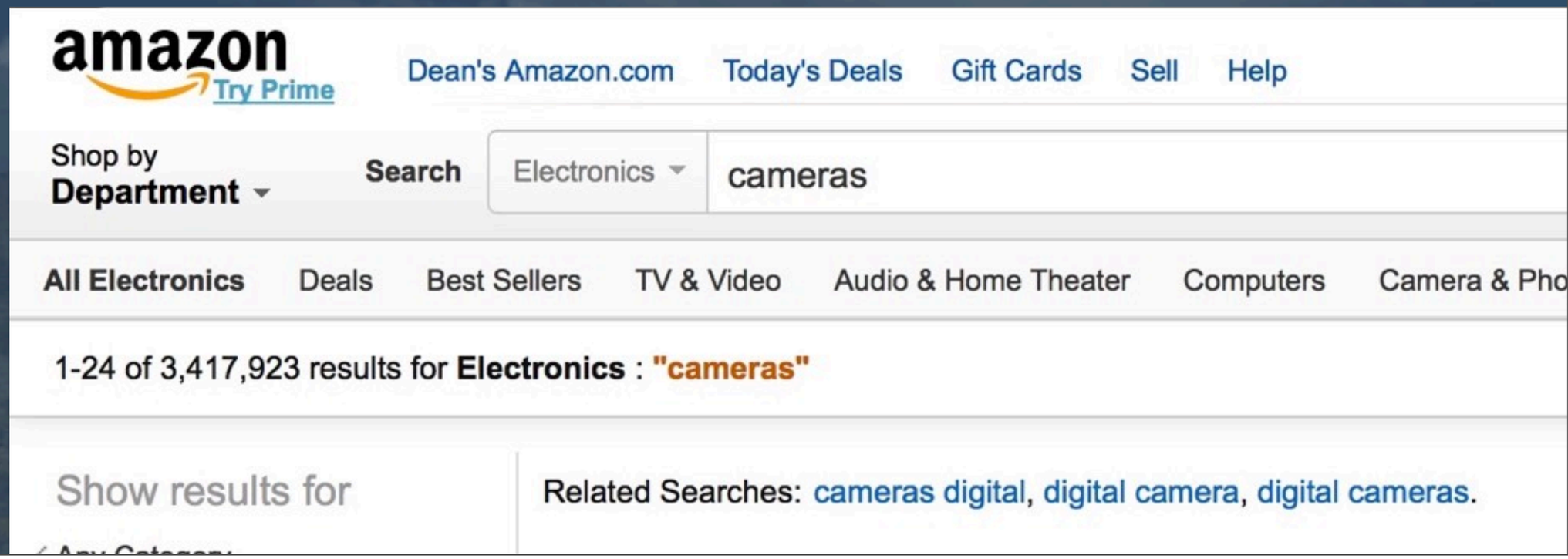
Tuesday, March 17, 15

This talk is a general overview of Reactive concepts. I’m going to dive into the “Resilient” trait in more detail later today.

Tuesday, March 17, 15

Let’s motivate the notion of “reactive” systems by exploring some common scenarios we see today.

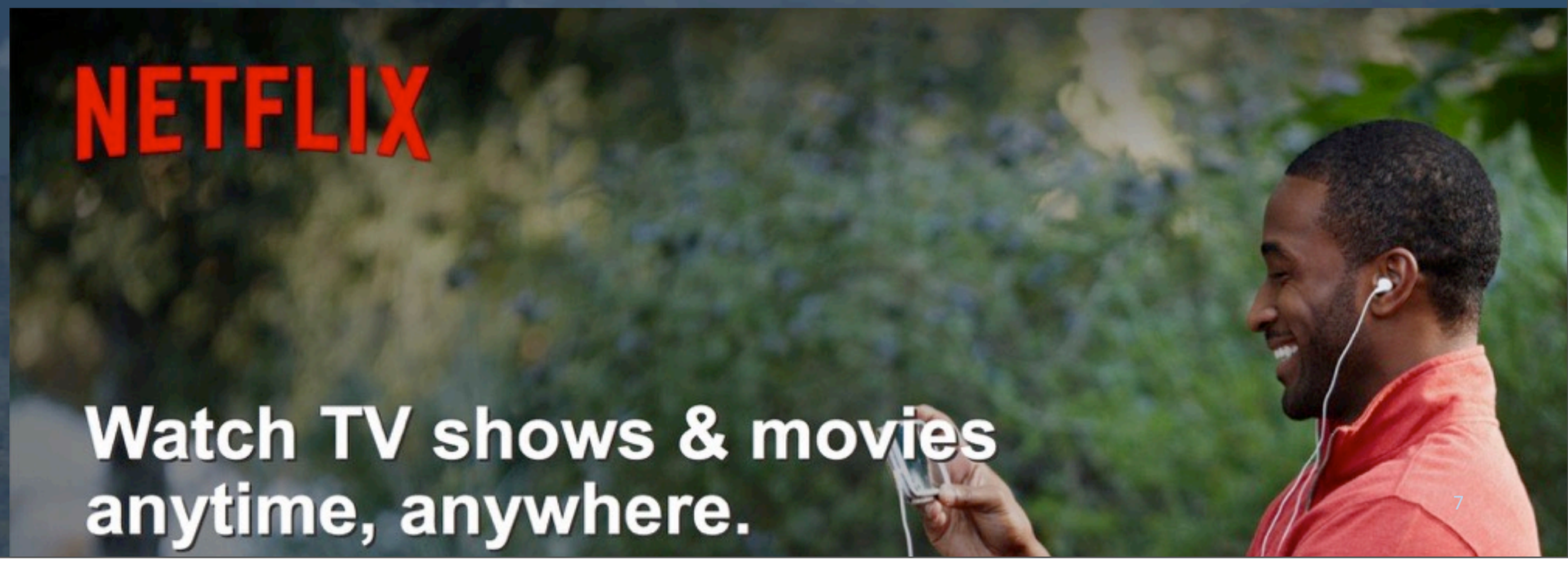
Cyber Monday?



Tuesday, March 17, 15

Your online store needs to scale up and down with demand. It needs to degrade gracefully if some service components are lost or disconnected by a network partition. For example, if the canonical catalog “disappears” behind a network partition, it’s probably better to sell of a stale copy locally.
photo: Amazon home page, http://amazon.com.

On demand?



Tuesday, March 17, 15

Netflix has extreme scale challenges, but they have become the industry leader in building highly resilient, scalable services. What happens when a new season of “House of Cards” is released?
photo: Netflix home page, http://netflix.com.

Motivation:

Internet of Things

Internet of Things has several categories of applications, each of which has needs that motivate reactive programming.

Medical Devices, IT Systems

Phone home:

- Upload data
- Usage patterns
- Predictive diagnostics
- Fetch patches

Tuesday, March 17, 15

Medical devices upload test results (e.g., ultrasound images and video) to servers. Med. devices and IT systems send requests for automated updates, send the equivalent of “click-stream” data used to assess usability, etc., and increasingly send metrics used to predictive potential HW failures or other service needs.
ultrasound photo: http://www.usa.philips.com/healthcare-product/HC795054/hd11-xe-ultrasound-system
switch photo: http://networklessons.com/switching/introduction-to-vtp-vlan-trunking-protocol/

Medical Devices, IT Systems

Characteristics:

- Stable to intermittent network connectivity
- One way and two way
- Mixed bandwidth

Tuesday, March 17, 15

A mobile scanner might move in and out of WiFi zones, so caching data is necessary. IT appliances are (hopefully) always online. Some data is one way, like diagnostic info for predictive analytics, while data uploads or patch requests need acknowledgement. Bandwidth can vary.

Aircraft Engines

Phone home:

- Upload telemetry
  - Predictive diagnostics
- Redundant tracking data!

Tuesday, March 17, 15

Each engine collects 0.5-1TB of data per flight. Most is currently tossed! Our only clue about the final resting place of Malaysian Airlines Flight 370 was engine telemetry picked up by satellites and used to analyze possible routes.

Trucks, Farm Equipment

GPS Tracking:

- Optimize routing, fuel use, etc.
- Spy on drivers?
- Per plant tracking!

Tuesday, March 17, 15

Track data to optimize routing, minimize fuel use with shortest path and/or delivering heaviest items first. Ensure drivers are obeying the rules of the road and company policies. Some farm equipment planting, watering, and fertilizing gear now tracks data per plant!

UPS truck photo: http://en.wikipedia.org/wiki/United_Parcel_Service

Planter/seeder photo: http://www.deere.com/en_US/products/equipment/planting_and_seeding_equipment/planting_and_seeding_equipment.page?

Trucks, Farm Equipment

Connectivity:

- Always along roads
- Intermittent on farms (WiFi in barns?)

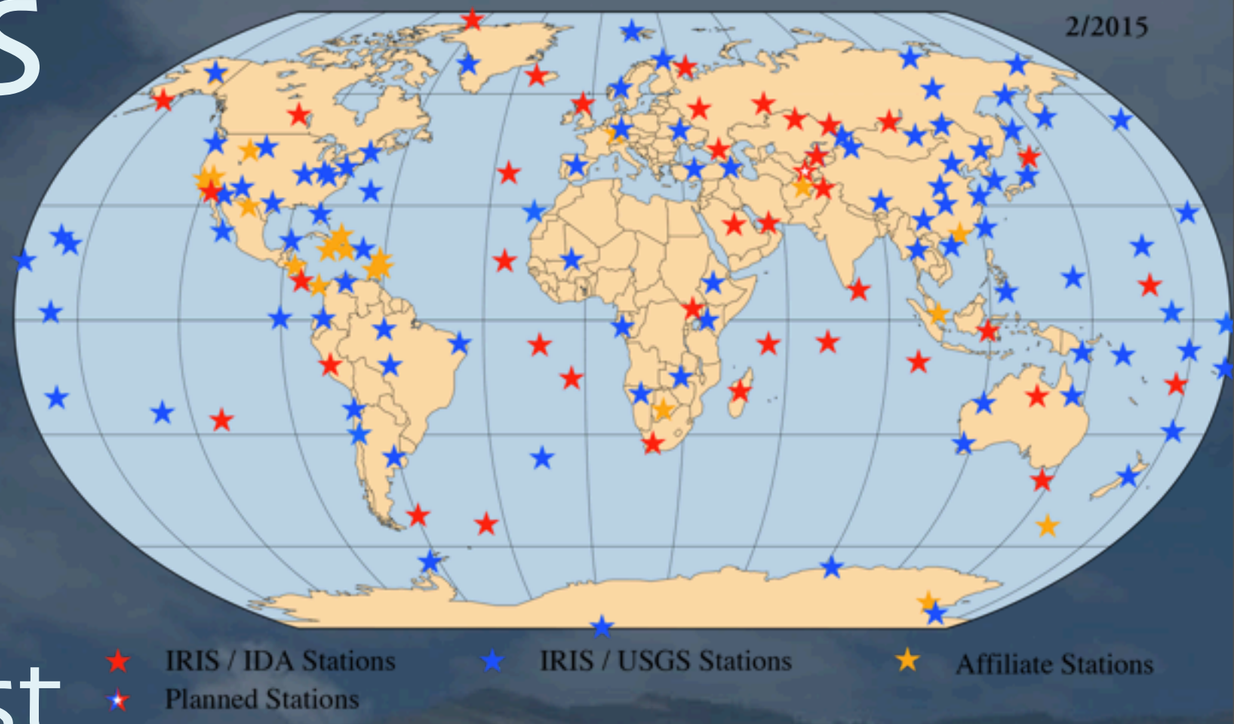
Tuesday, March 17, 15

Some rural areas don’t have sufficient wireless data coverage for farm equipment to remain to online full time.

Remote Sensors

Human to Real-time Responsiveness:

- Earthquake, nuclear test sensor networks.
- Climate change monitoring



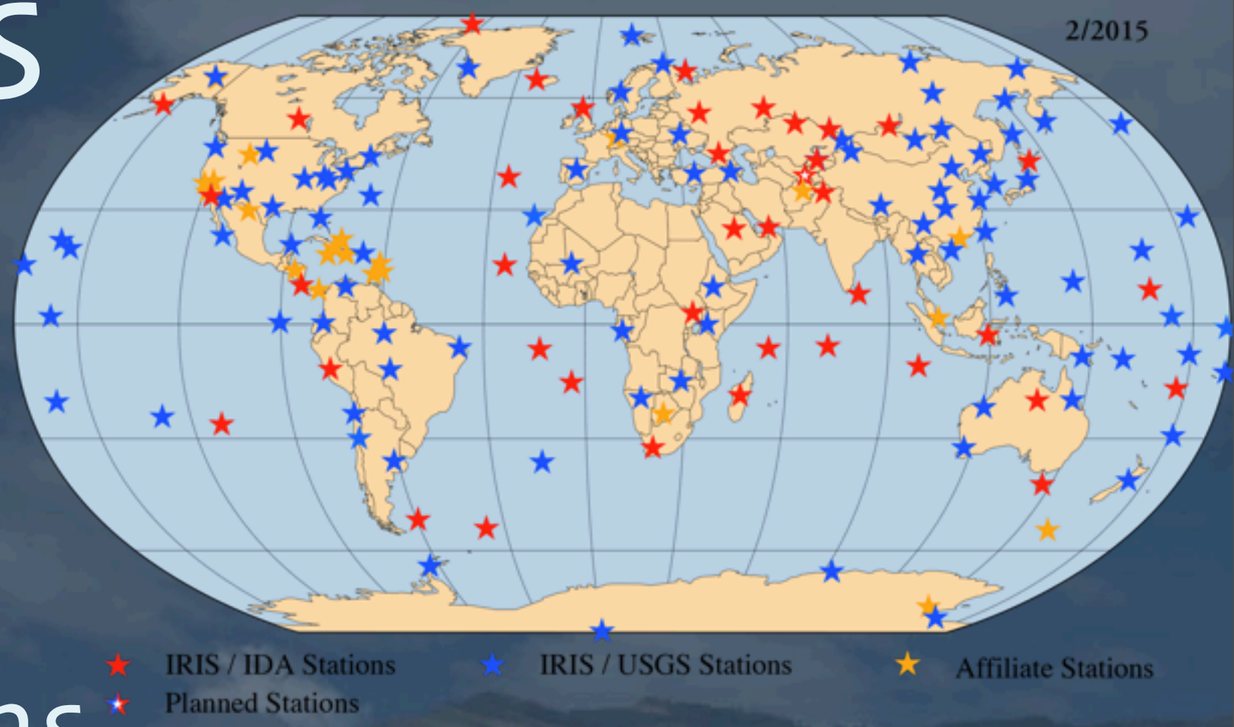
Tuesday, March 17, 15

For earthquake sensor networks, you want to get the information to emergency services and community alarm systems in milliseconds. This requires redundant sensors, always on connectivity, and low-latency connections. The amount of data isn't broad. Some networks, like monitoring rainfall or glaciers for climate change studies, might be offline except for once-per-year downloads onsite!
photo: http://www.iris.edu/hq/programs/gsn

Remote Sensors

Characteristics:

- Redundant sensors
- Low-latency connections
- Low-bandwidth requirements



Tuesday, March 17, 15

For earthquake sensor networks, you want to get the information to emergency services and community alarm systems in milliseconds. This requires redundant sensors, always on connectivity, and low-latency connections. The amount of data isn't broad. Some networks, like monitoring rainfall or glaciers for climate change studies, might be offline except for once-per-year downloads onsite!

Robotics

Connectivity:

- Two-way, but time of flight matters!
- Autonomous?

Tuesday, March 17, 15

Some rural areas don't have sufficient wireless data coverage for farm equipment to remain to online full time.
Quadcopter photo: http://www.dji.com/product/phantom
Mars rover photo: http://en.wikipedia.org/wiki/Mars_Exploration_Rover

Health Monitoring

Characteristics:

- Occasional to always-on connectivity
- Detect health emergencies: call for help?

Tuesday, March 17, 15

Health monitoring tools are most popular for gathering activity data and some vital signs for analysis later. Some monitors are designed to detect medical emergencies and call for help when needed.
photo: http://www.fitbit.com/force

Home Automation

Characteristics:

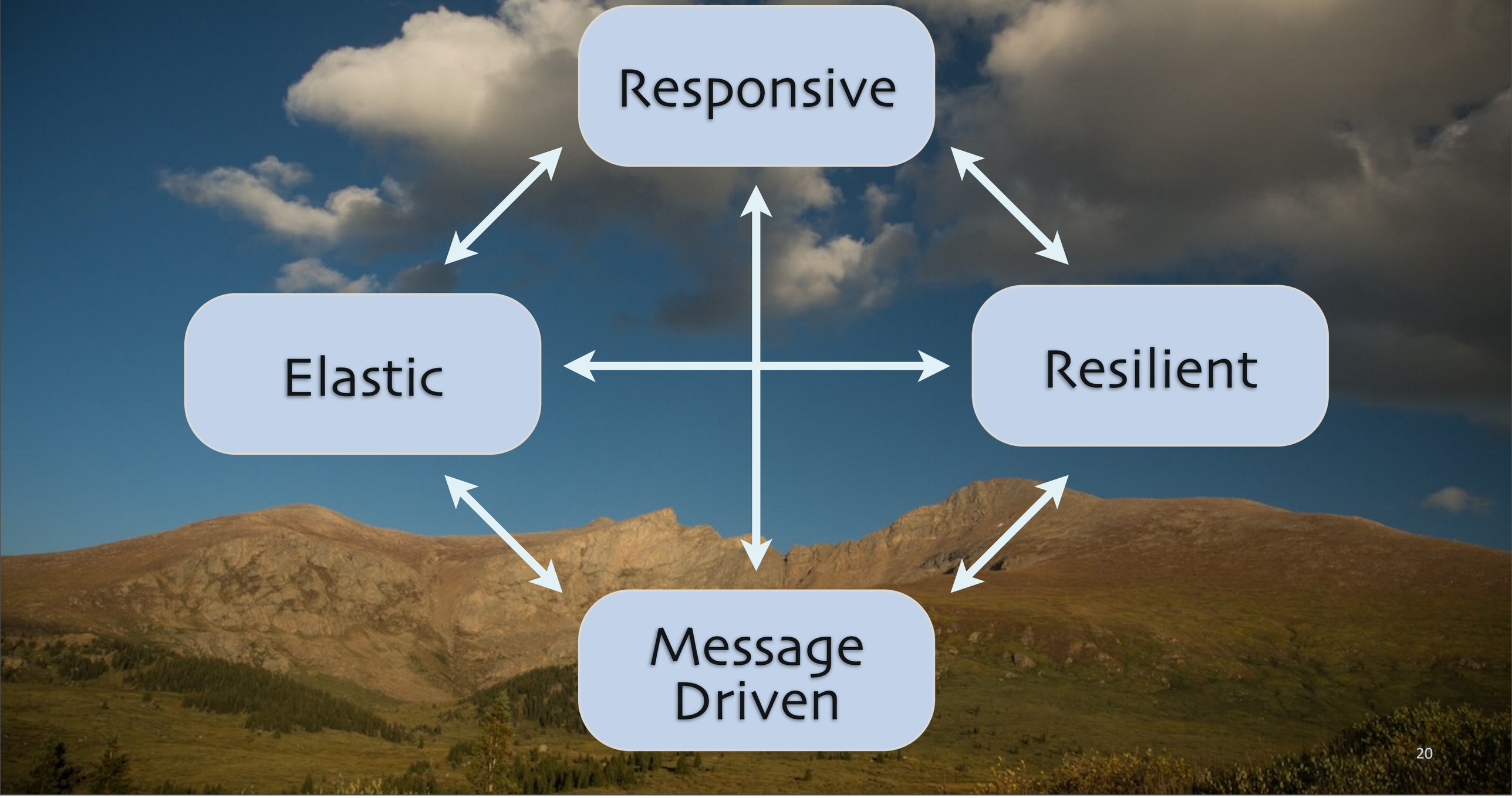
- Morning packet storms
- Fire & break-in detection: automatic notification of authorities

Tuesday, March 17, 15

photo: https://store.nest.com/product/thermostat/

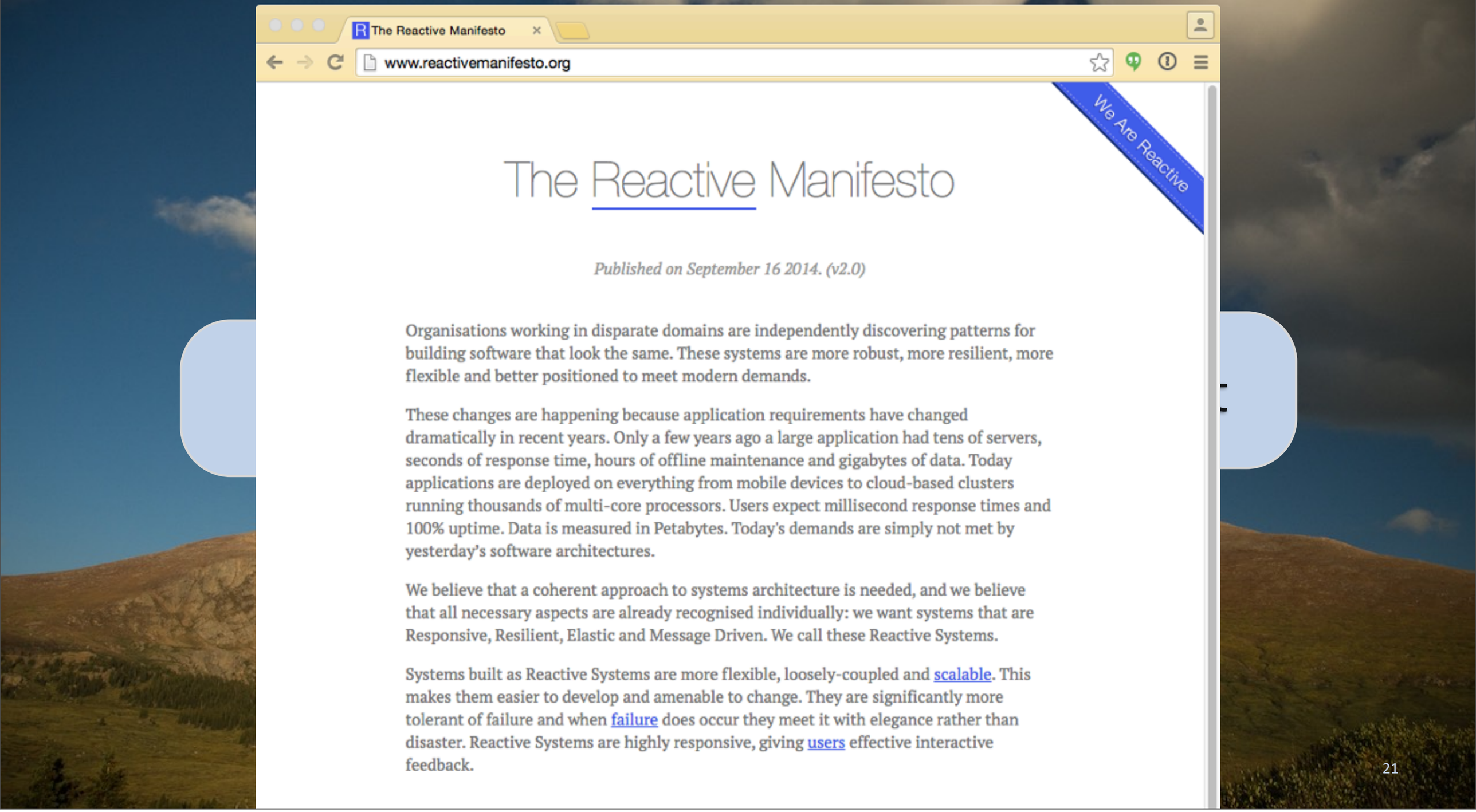
Reactive Systems

The idea of Reactive Systems emerged to catalog universally common characteristics of the systems we have to build to support these scenarios.



Tuesday, March 17, 15

The four characteristics of Reactive Systems... as articulated by the Reactive Manifesto, which attempts to codify lessons learned across many projects, industries, and years building highly available, scalable, and reliable systems.



The four characteristics of Reactive Systems... as articulated by the Reactive Manifesto, which attempts to codify lessons learned across many projects, industries, and years building highly available, scalable, and reliable systems.

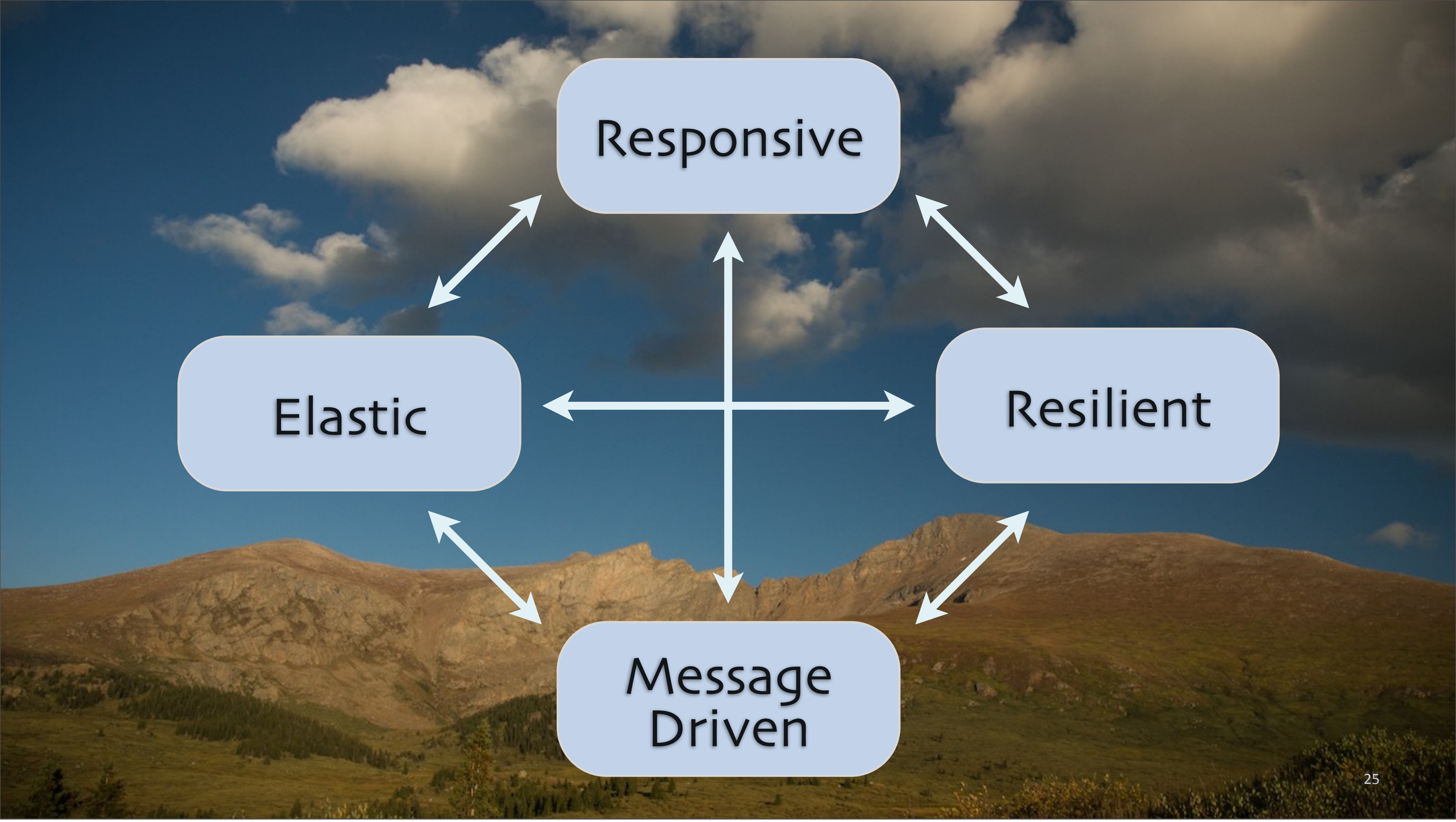

Myths

Tuesday, March 17, 15

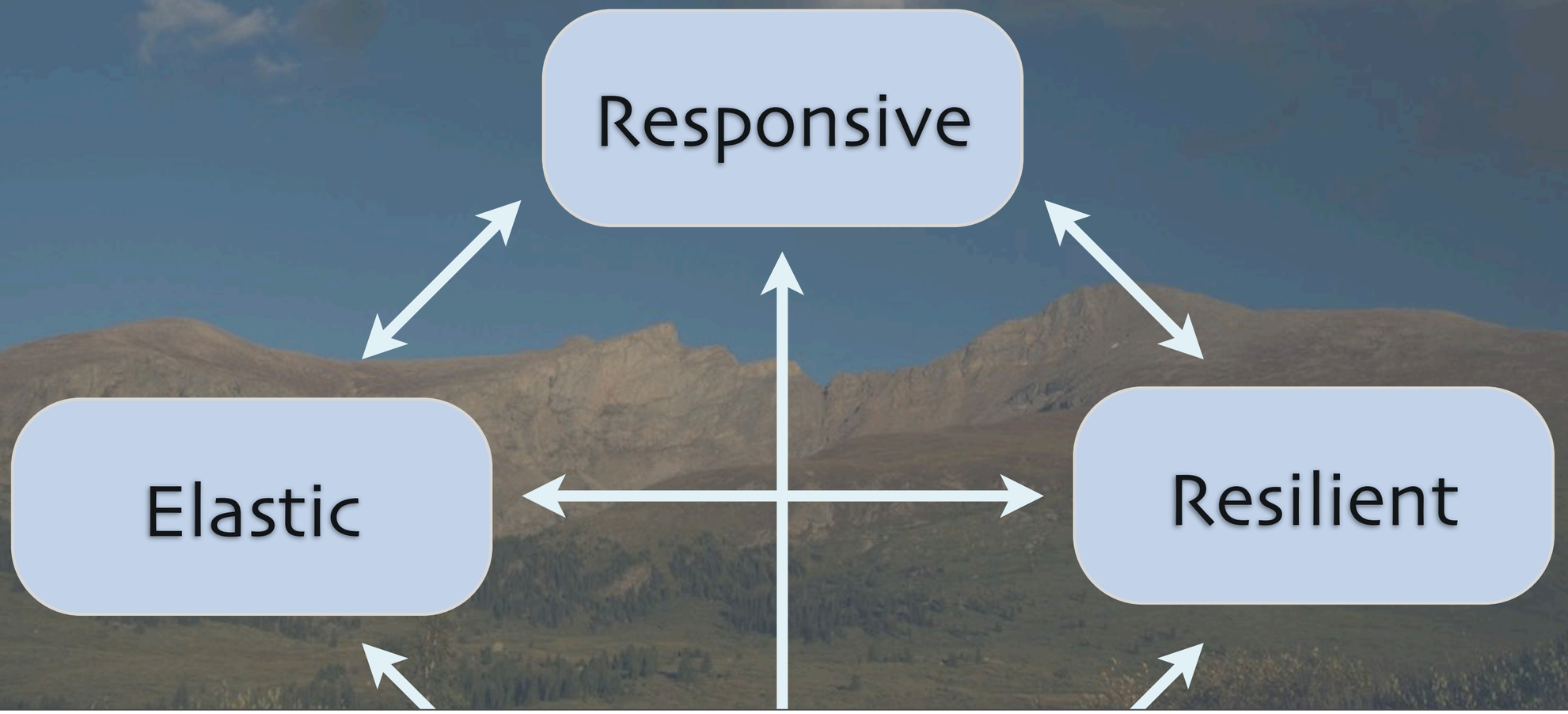
The RM attempts to codify lessons learned over many years in many scenarios. It’s not new. It doesn’t claim to be new.

Tuesday, March 17, 15

While Typesafe's Jonas Bonér was one of the originators of the RM, other originators and contributions include experts in many companies and specialties.


Responsive
Elastic
Resilient
Message
Driven

Requests or commands require timely responses.




Responsive

Responsive

Cornerstone of usability and utility.

Responsive

Requires rapid detection of errors and quick responses.

Responsive

Requires predictable response times and quality of service.

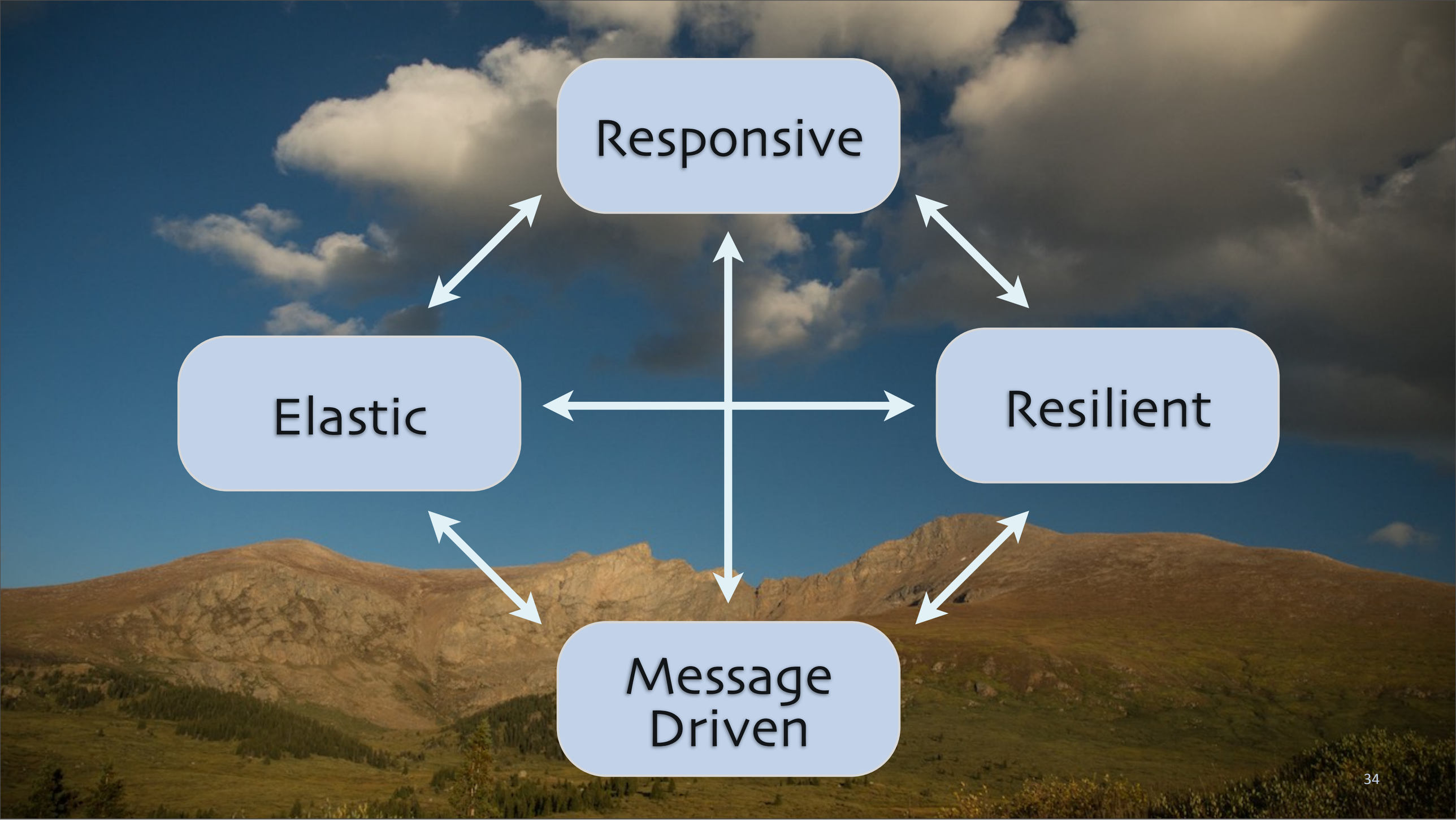
Responsive

Awareness of time is first class.

Example:
Netflix Simian Army

Image from http://devops.com/features/netflix-the-simian-army-and-the-culture-of-freedom-and-responsibility/

Clobber services, servers,
even datacenters
in production,
to verify service continuity.


Responsive
Elastic
Resilient
Message
Driven

Tuesday, March 17, 15

Truly resilient systems must make failures first class citizens, in some sense of the word, because they are inevitable when the systems are big enough and run long enough.


Resilient

Resilient

Failure is not disruptive.


Resilient
Failure is
first class.


Resilient
Requires replication,
containment, isolation,
and delegation.

Example: Failure-handling in Actor Systems

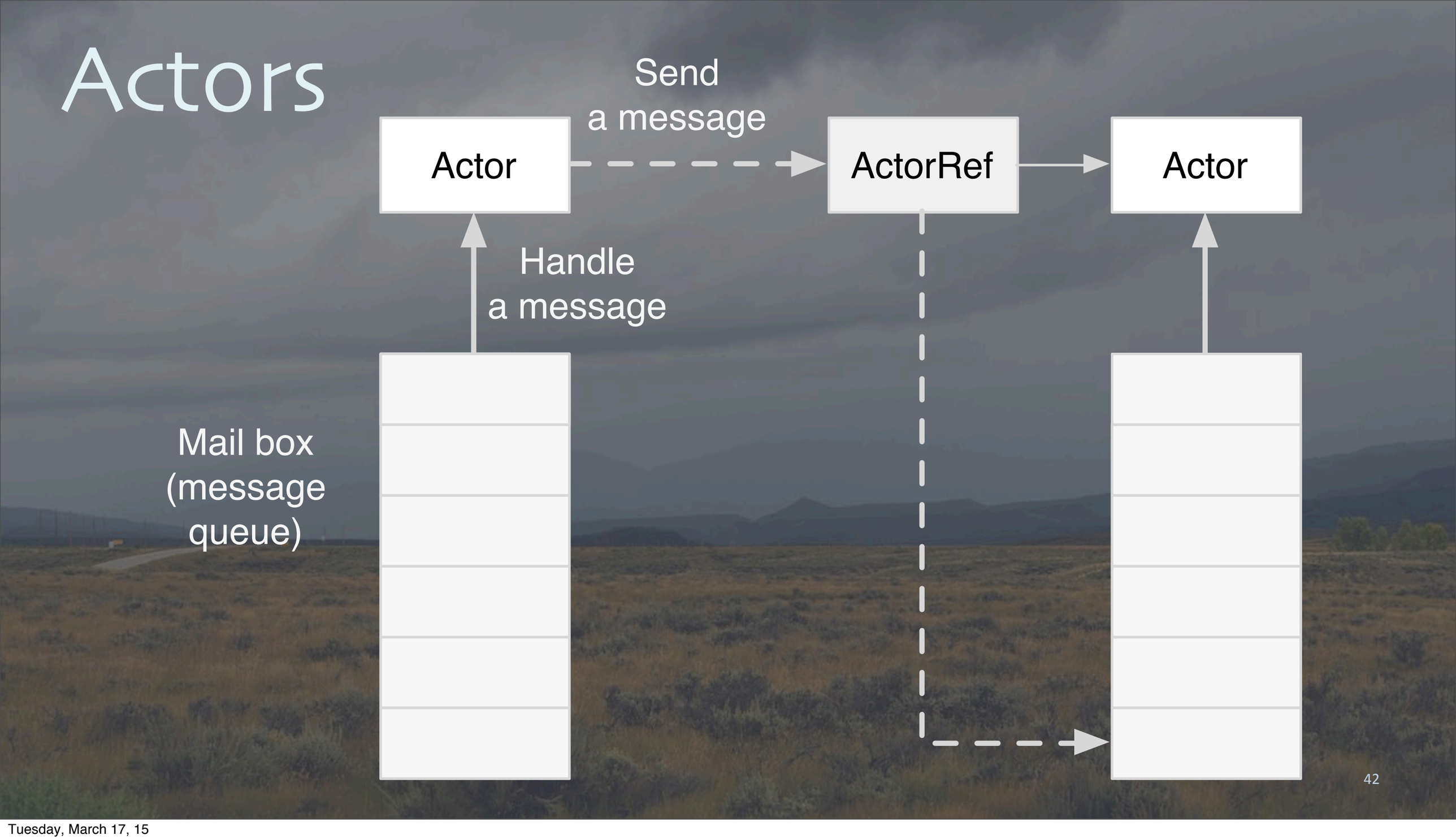
Tuesday, March 17, 15

The Netflix Simian Army could also be cited here.
We'll come back and fill in details of Actor systems shortly. For now, let's focus on error handling.

Tuesday, March 17, 15

Rather than attempt to recover from errors inside the domain logic (e.g., elaborate exception handling), allow services to fail, but with failure detection and reconstruction of those services, plus failover to other replicas.

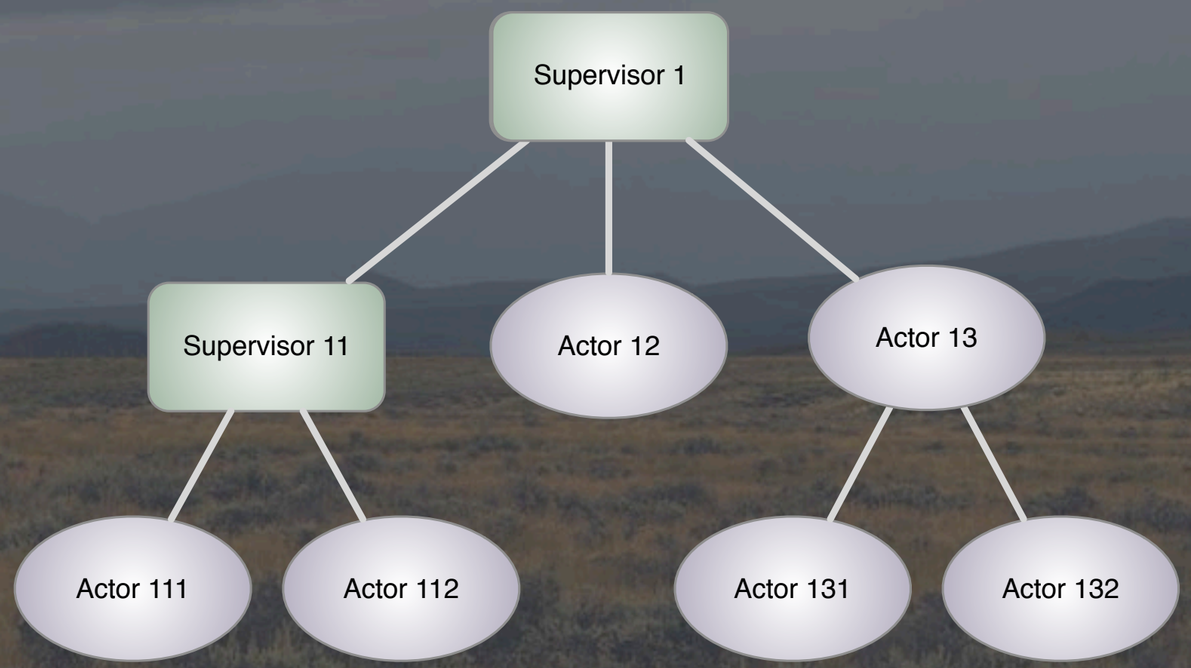
Actors



Tuesday, March 17, 15

Actors are similar to objects in Smalltalk and similar systems; autonomous agents with defined boundaries that communicate through message passing. Actors, though process each message in a threadsafe way, so they are great for concurrency. (This diagram illustrates the Akka implementation – http://akka.io)

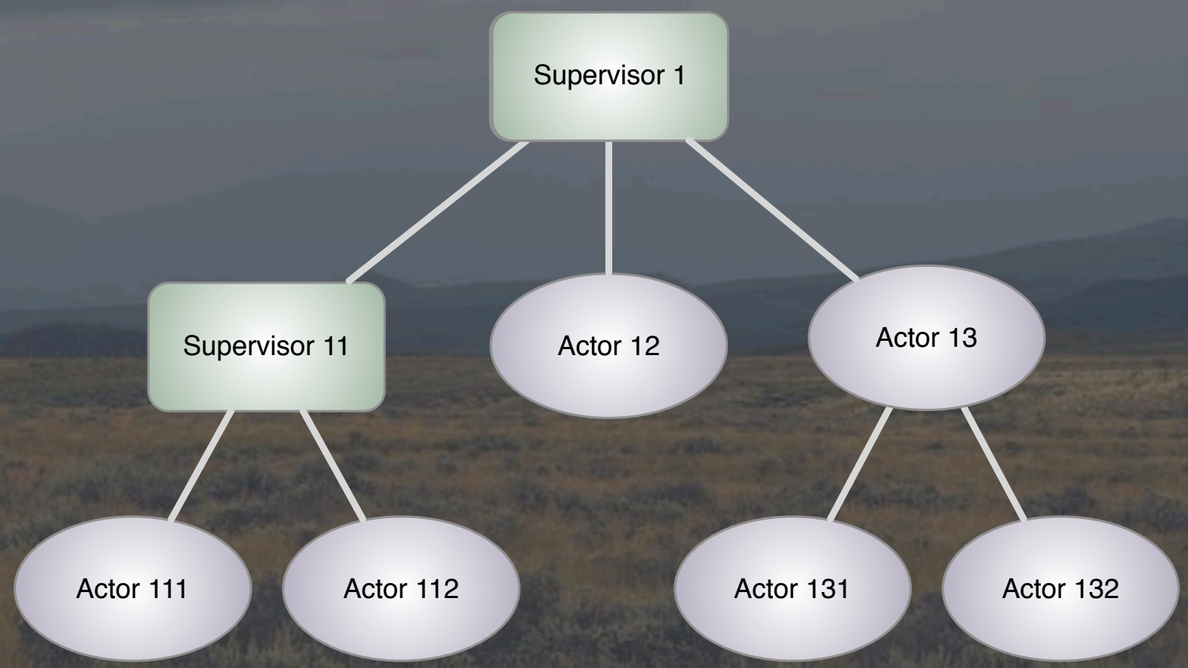
Erlang introduced supervisors. A hierarchy of actors that manage each "worker" actor's lifecycle.

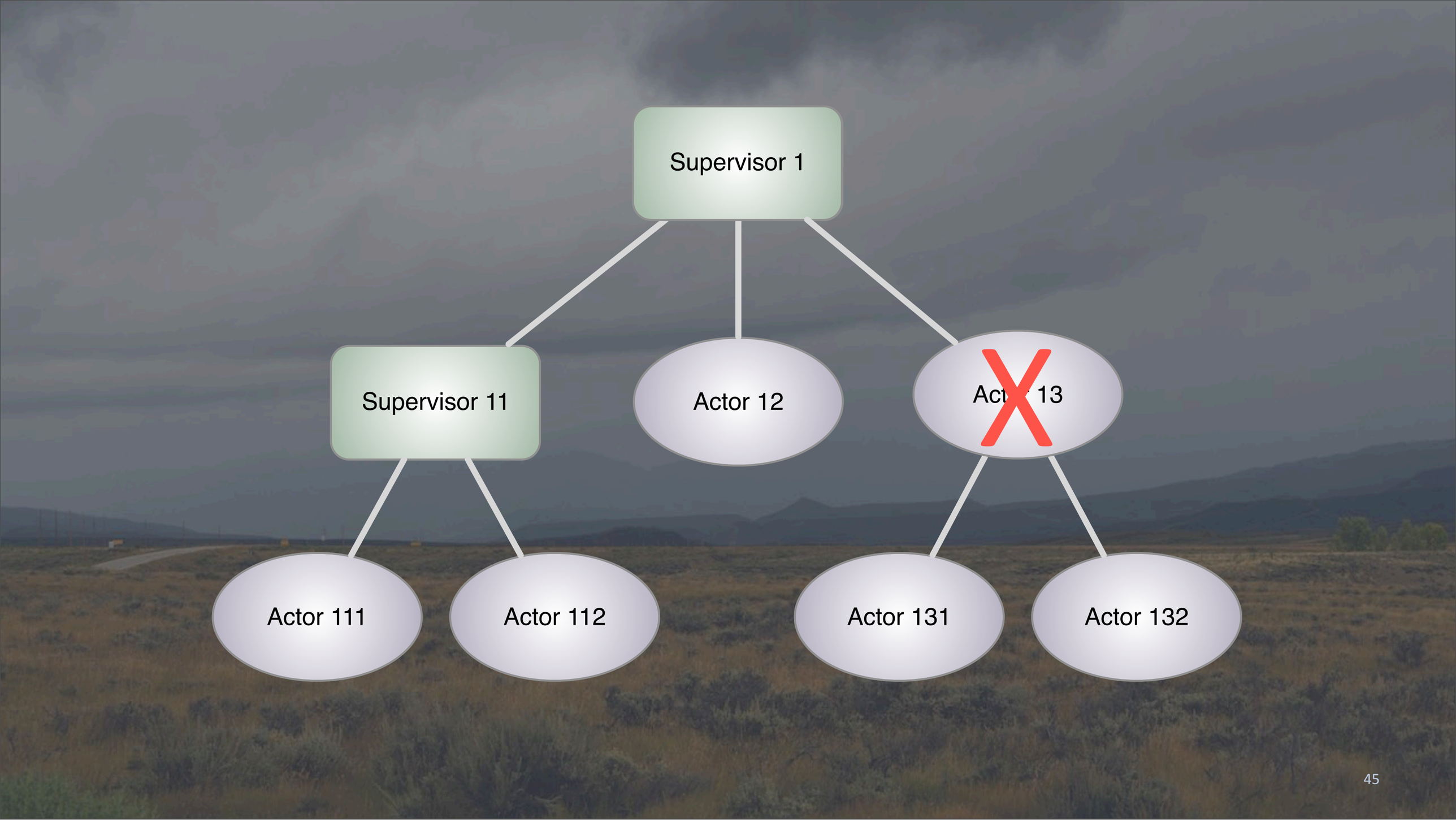


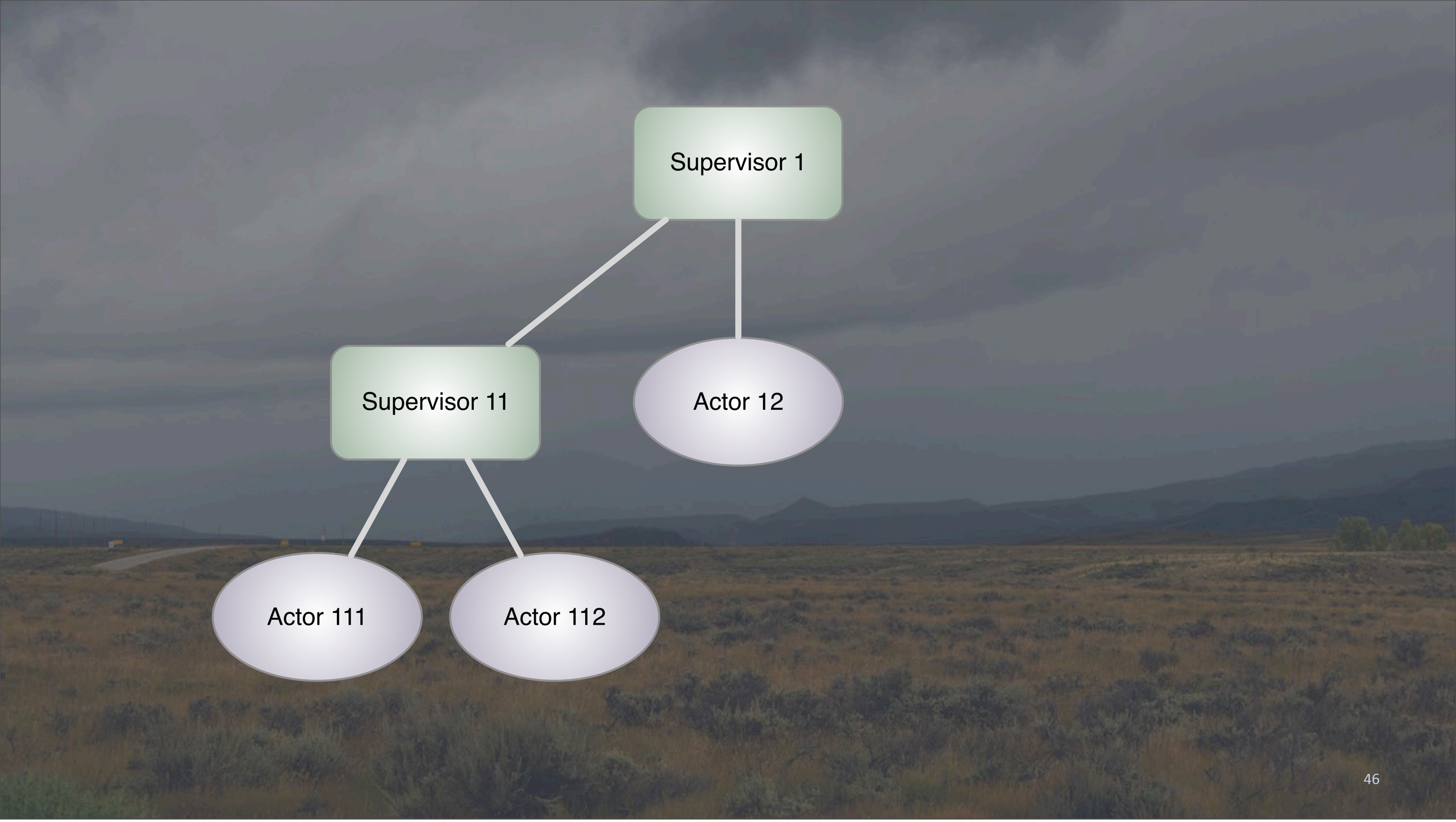
Tuesday, March 17, 15

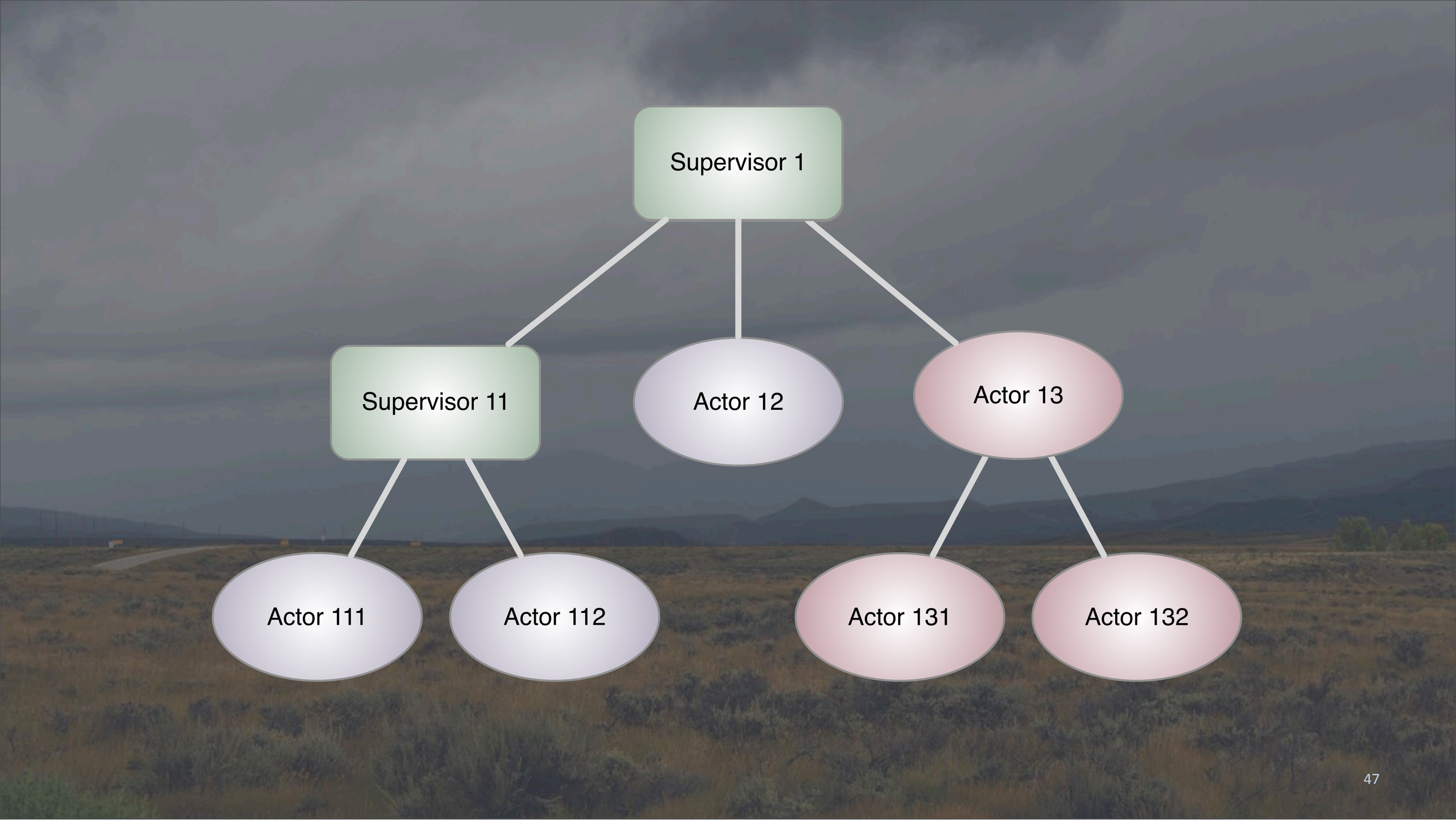
There are lots of so-called Actor systems, but be wary of them unless they have this sophisticated supervision model or something like it (even though the original Actor model of Hewitt, et al., didn't include supervision like this...).

Generalizes nicely to distributed actor systems.




Supervisor 1
Supervisor 11
Actor 12
Actor 13
Actor 111
Actor 112
Actor 131
Actor 132


Supervisor 1
Supervisor 11
Actor 12
Actor 111
Actor 112

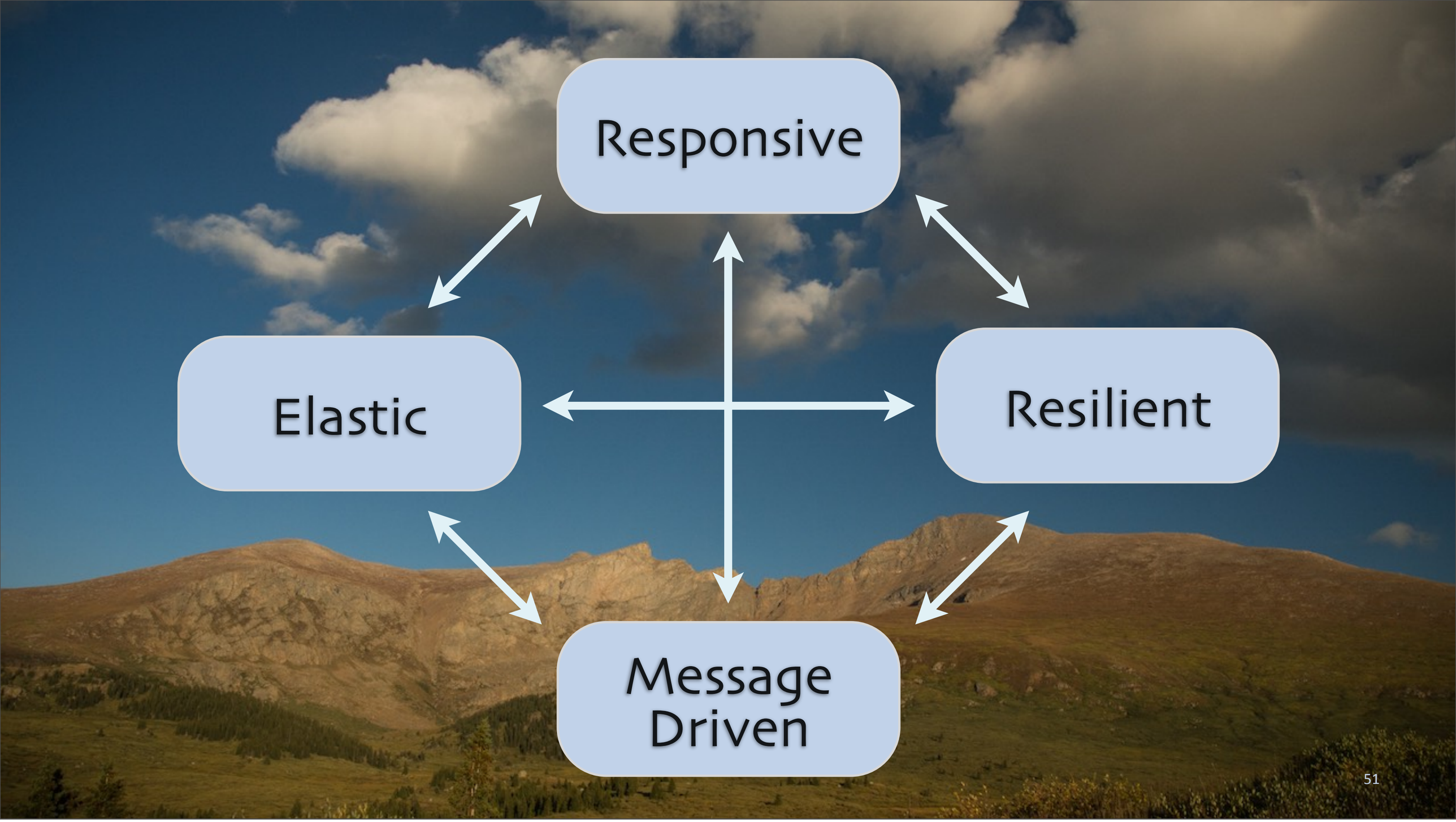

Supervisor 1
Supervisor 11
Actor 12
Actor 13
Actor 111
Actor 112
Actor 131
Actor 132

The most
sophisticated error recovery
in reactive systems.

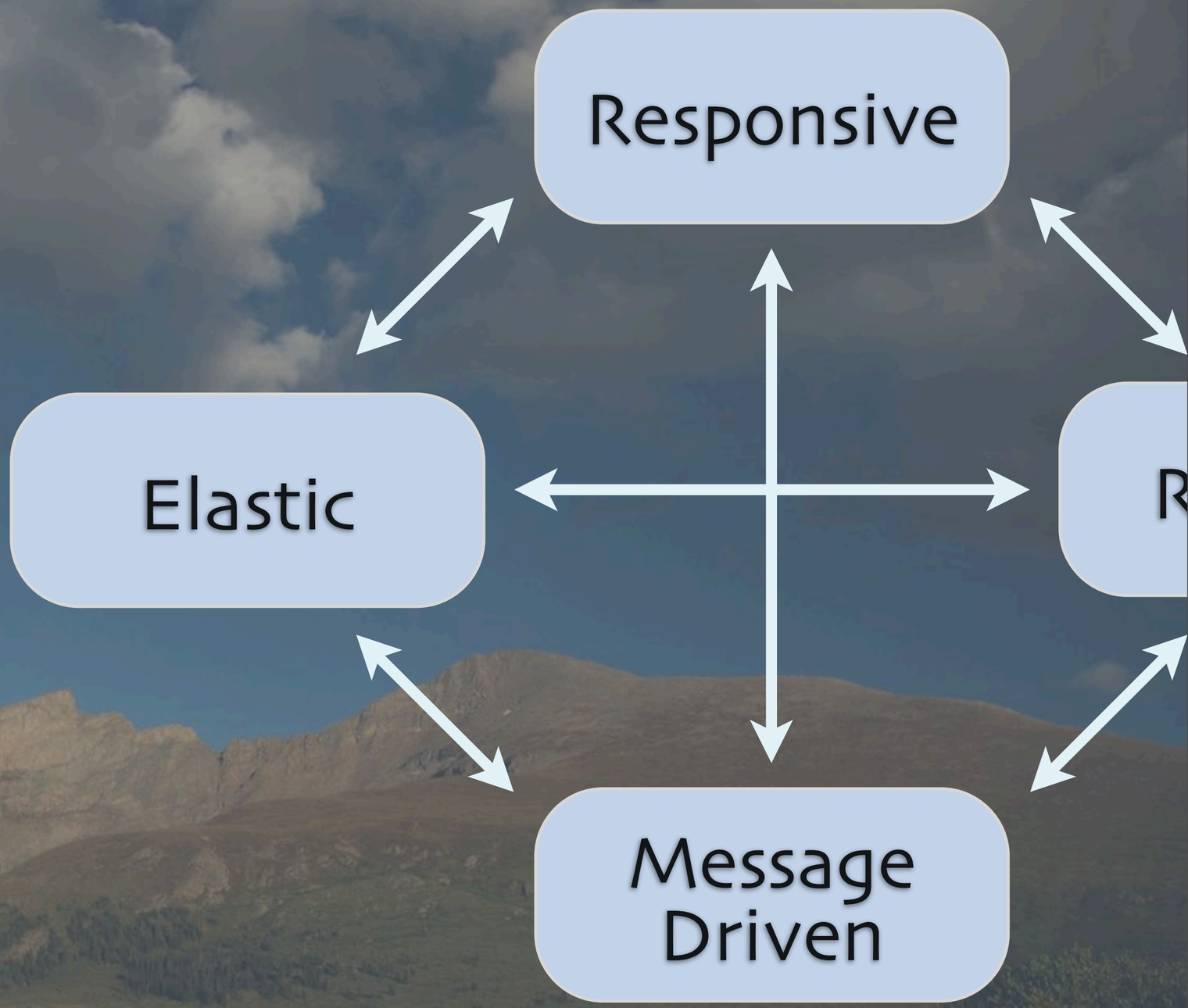

Clean separation
of normal processing
from recovery.

Not using Actors?
See Hystrix
from Netflix.

https://github.com/Netflix/Hystrix


Responsive
Elastic
Resilient
Message
Driven

Scale up and down



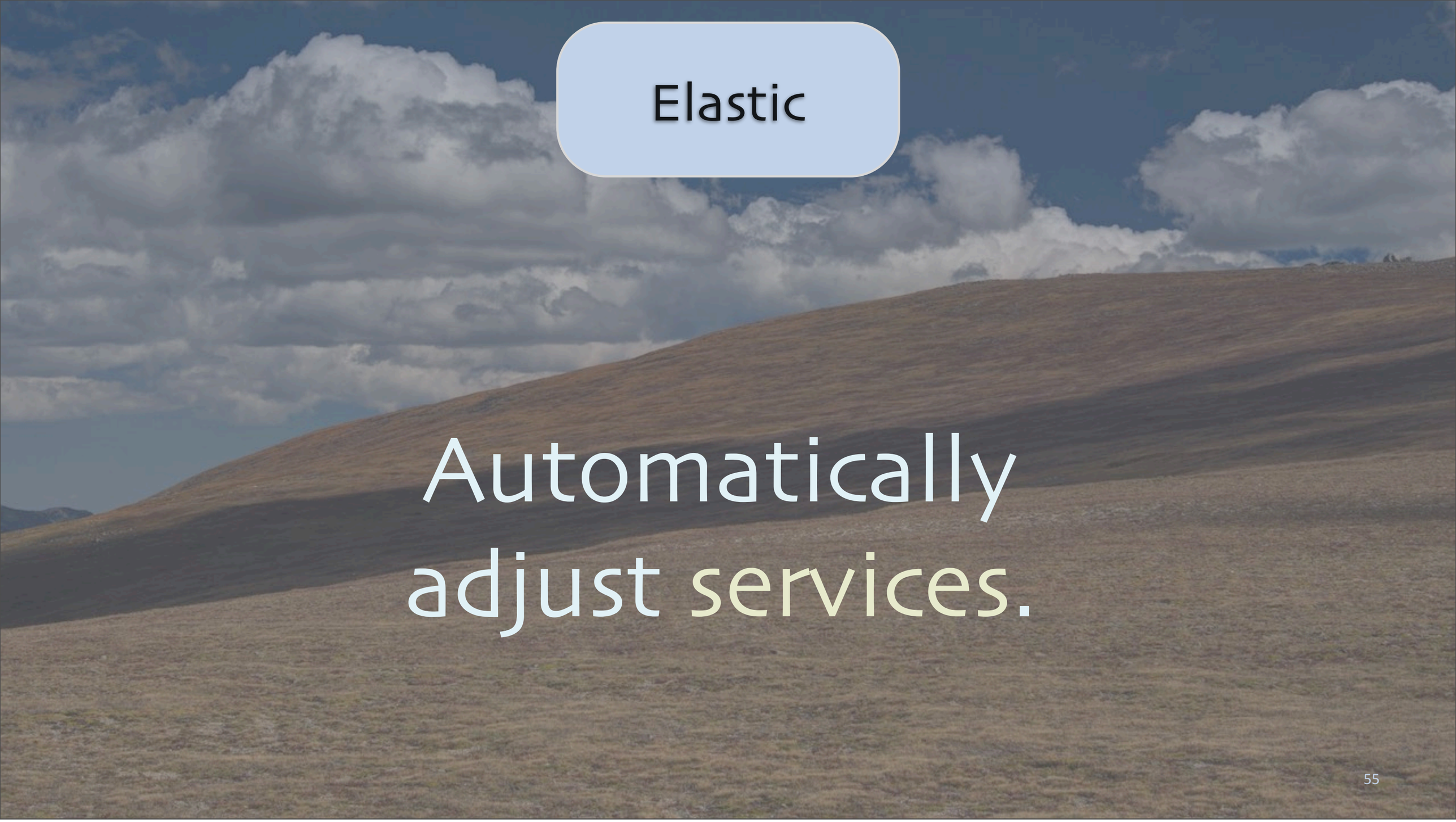
Tuesday, March 17, 15

As demand rises and falls, you must gracefully scale up to meet increasing demand and scale down to conserve resources.


Elastic

Elastic

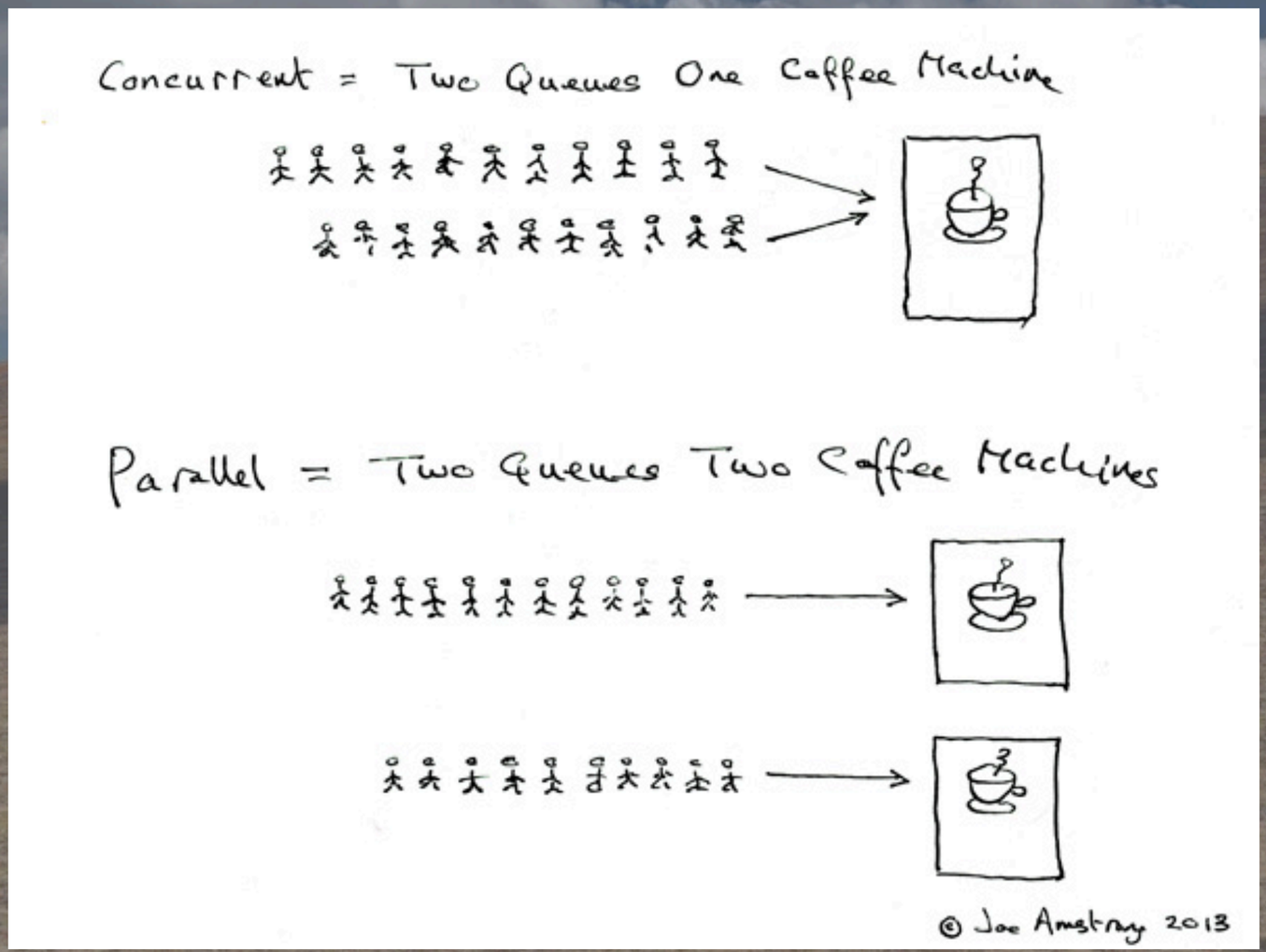
Detect changing input patterns.


Elastic
Automatically
adjust services.


Elastic
Scale across
commodity hardware.

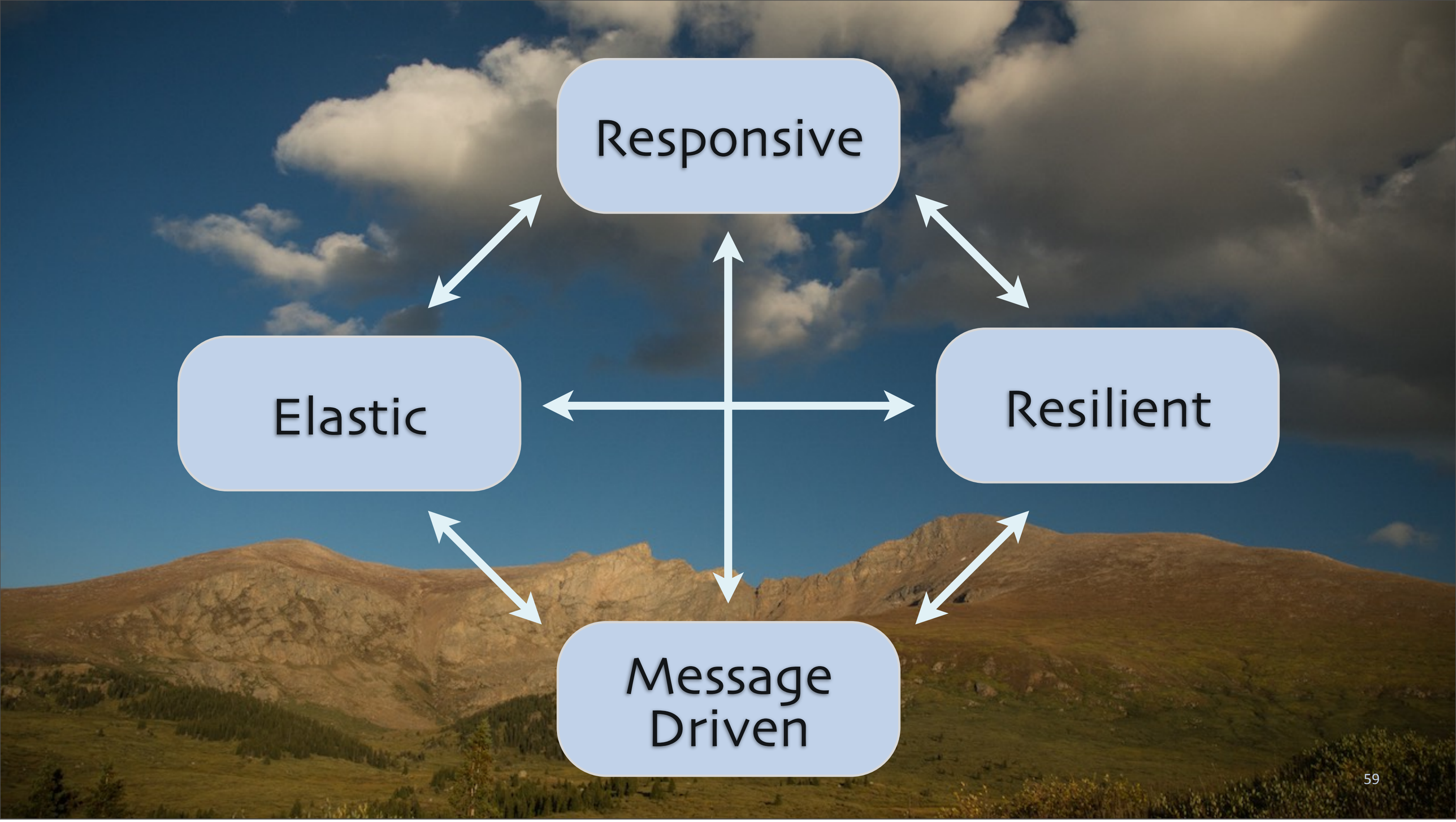

Elastic
No bottlenecks
or contention points.

Elastic



Tuesday, March 17, 15

"Concurrent" vs. "parallel".


Responsive
Elastic
Resilient
Message
Driven

Tuesday, March 17, 15

To be responsive to the world around you, you must interact with the world through messages.

Message Driven

Message Driven

Asynchronous message passing.

Message Driven

Defines boundaries, promotes loose coupling and isolation.

Message Driven

Promotes location transparency.

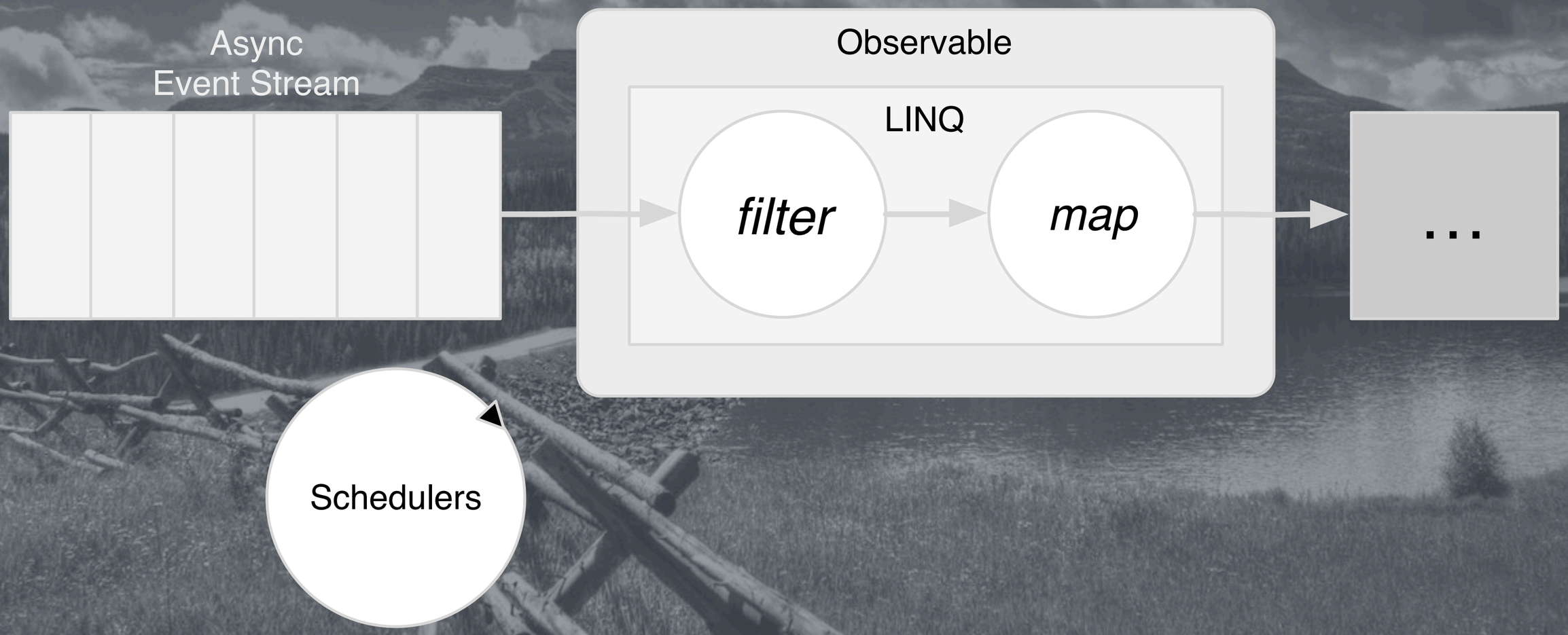
Handle errors as messages.

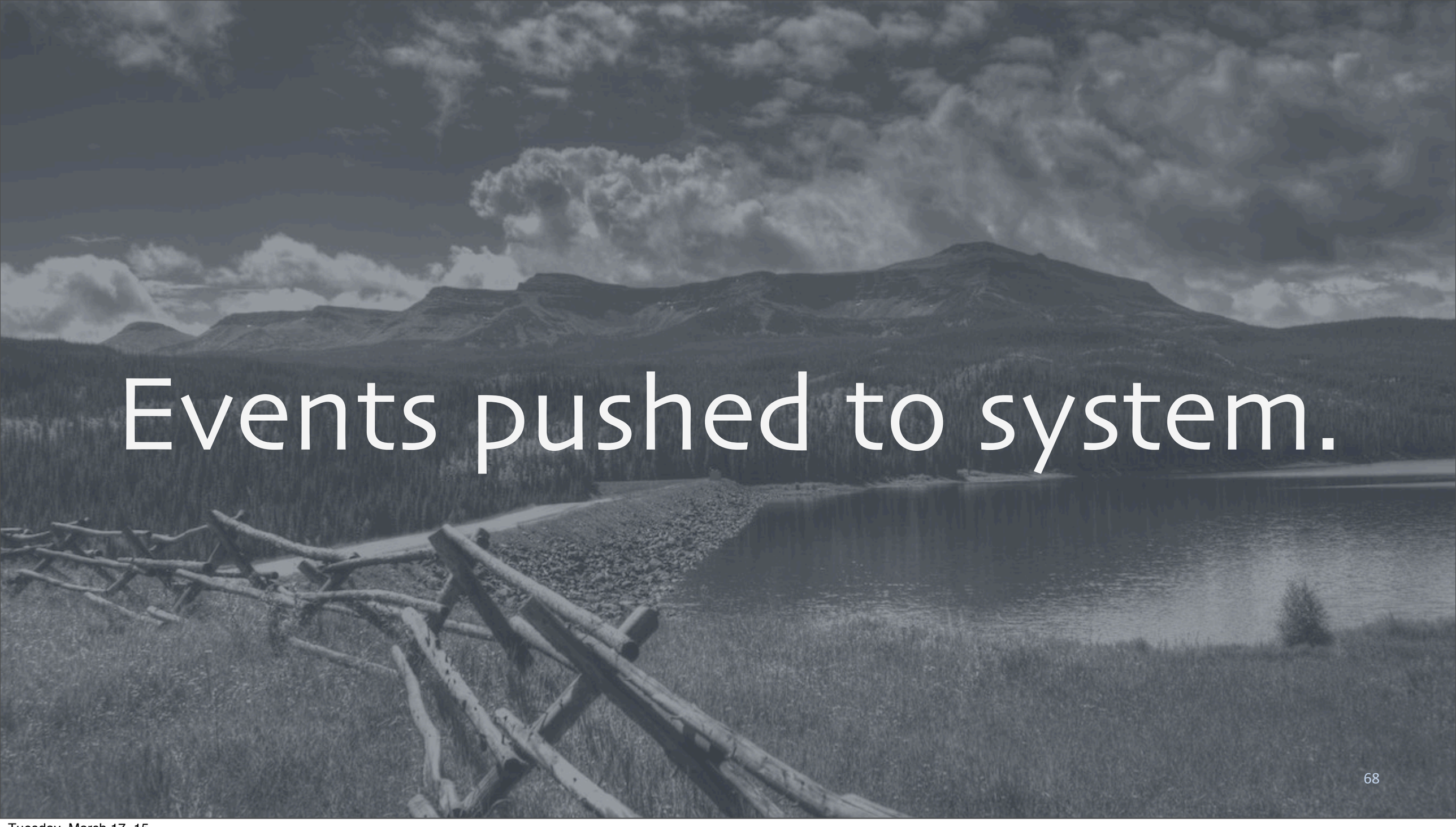
Message Driven

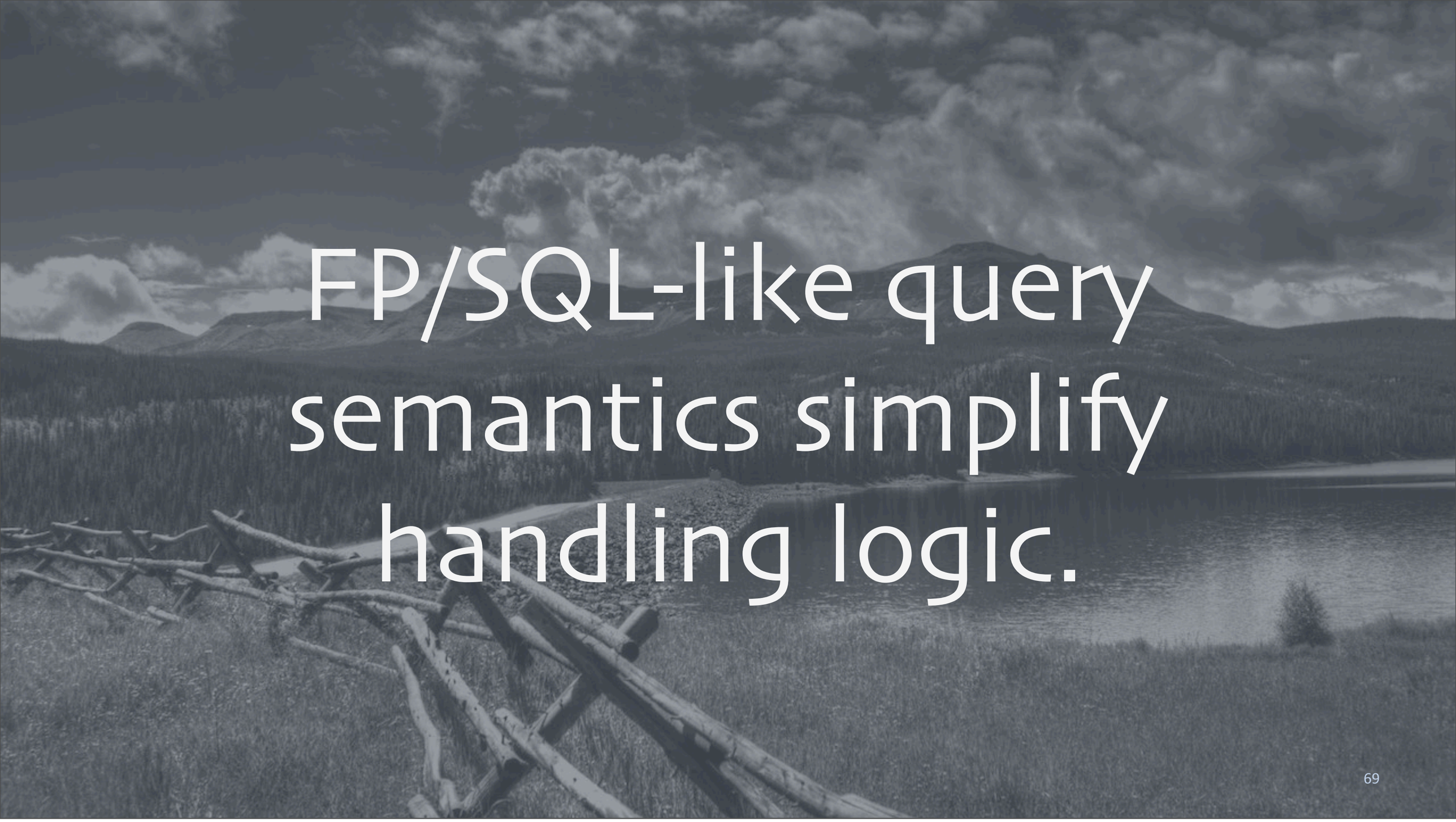
Promotes global management and flow control through backpressure.


Reactive Extensions (Rx)

LINQ Rx




Events pushed to system.


FP/SQL-like query
semantics simplify
handling logic.

How: Reactive Tools

Tuesday, March 17, 15

We mentioned a few already, let's fill in some details. This won't be an exhaustive list.
Hat tip to Jamie Allen at Typesafe for some of these ideas.

How: Reactive Tools

- Functional Programming
- Distributed Computing "Laws"
- Software Transactional Mem.
- Event Loops

Tuesday, March 17, 15

We mentioned a few already, let's fill in some details. This won't be an exhaustive list.

How: Reactive Tools

- CSP
- Futures
- Actors - Erlang or Akka
- Rx and variants
- Reactive Streams

CSP - Communicating Sequential Processes.

Functional Programming

Functional Programming

Need to be asynchronous and nonblocking.
Avoid locks.

Functional Programming

Prefer immutable values and side-effect free functions.

Functional Programming

Objects - suitable for modules.

Functional - for everything else.

Architecture Side Note:

The biggest mistake of OOP was the idea that we should faithfully model the world in code.

Distributed Computing

Distributed Computing

Serializability (order) and Linearizability (change history results in same order?).

CRDTs, Lattices.

Tuesday, March 17, 15

CRDTs - Commutative Replicated Data Types (http://pagesperso-systeme.lip6.fr/Marc.Shapiro/papers/RR-6956.pdf)
Lattices are more general concept applied here.

Software Transactional Memory

Popularized first in Hardware, then in Software by Haskell. Now used in persistent datastructures.

Software Transactional Memory

Principled local state mutation through transactions.

Limited scalability, no distribution.

Tuesday, March 17, 15

Great description of STM by Simon Peyton-Jones, from the O'Reilly Book Beautiful Code, http://research.microsoft.com/en-us/um/people/simonpj/papers/stm/#beautiful

Event Loops

Tuesday, March 17, 15

As explicitly managed in tools.

Event Loops

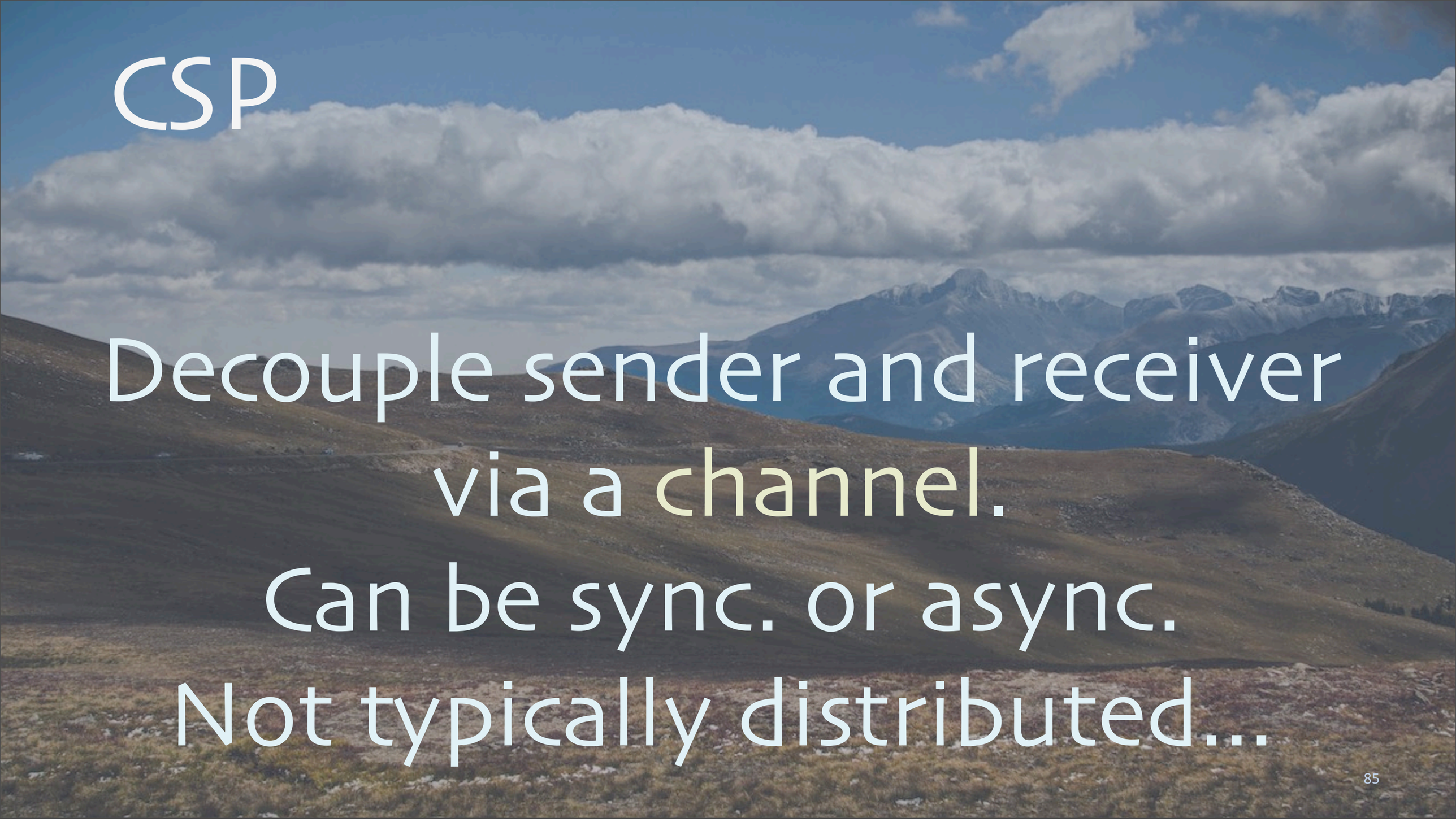
Pull from a thread.
Limited resiliency, scalability.
Callback hell...

Tuesday, March 17, 15

Many of these systems don't provide facilities for distribution or error recovery. Scalability can also be limited by the available threads, for those systems that pin a thread to each control flow construct. Some use a callback programming model exclusively, which quickly becomes unwieldy.

Tuesday, March 17, 15

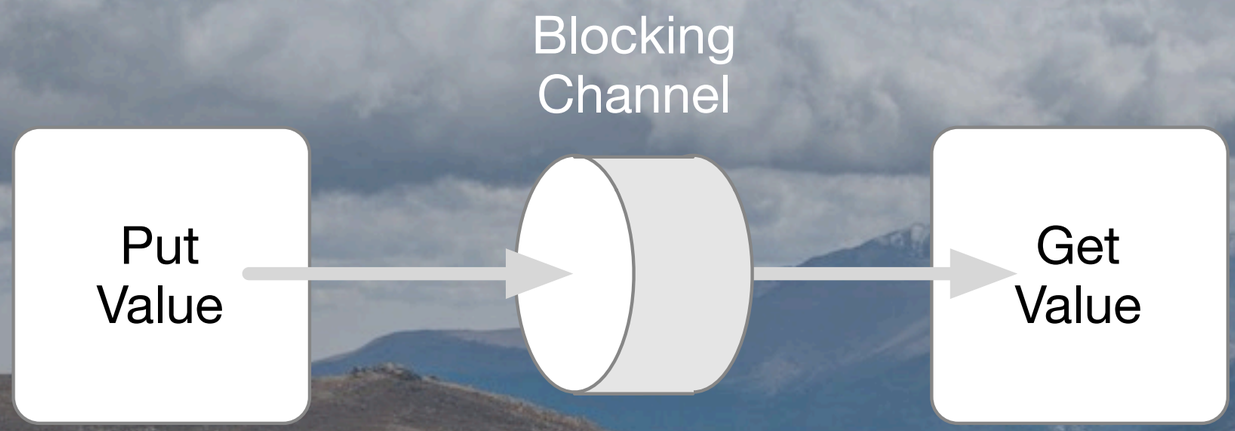
Communicating Sequential Processes - The first mathematical model of distributed computing. It has evolved somewhat and it's still popular in Clojure and Go, for example.

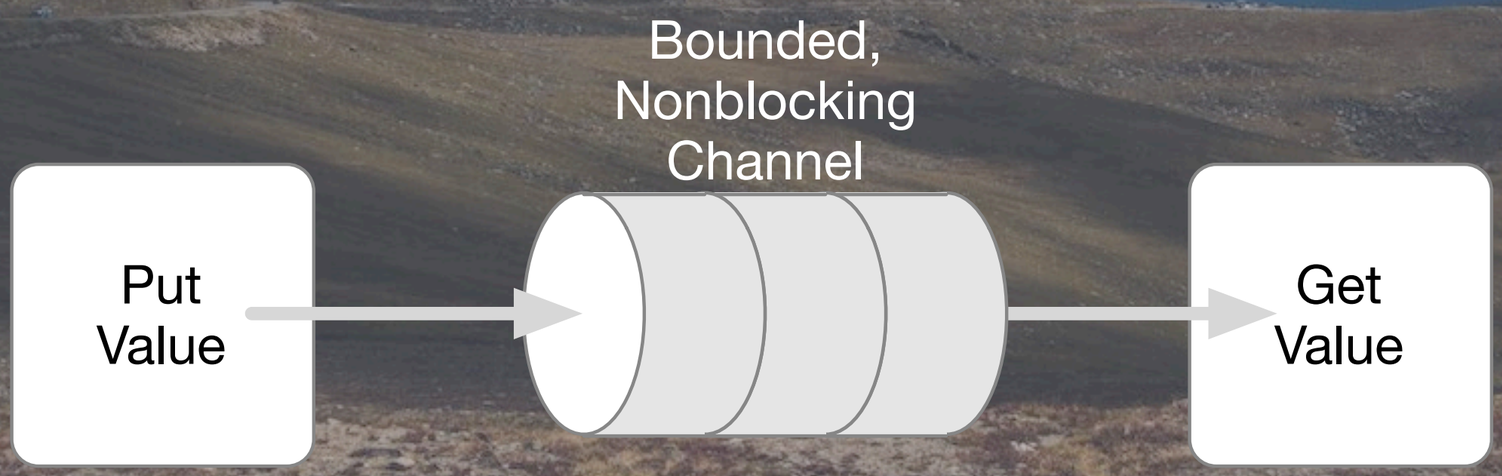


Tuesday, March 17, 15

Great abstraction for coordinating exchange of data, with a level of abstraction, the channel. Not really a scalable distributed system in the general sense.

CSP





Tuesday, March 17, 15

Schematic view of simple CSP interactions. See my talk on error handling in reactive systems where I discuss CSP in more detail.

Futures

Tuesday, March 17, 15

Fill more or less the same niche as CSP...

Futures

Define behaviors to run async.

Apply map, flatMap, ...

Or define success and failure callbacks.

Tuesday, March 17, 15

Like using the UNIX shell to fire a process in the background, but you either define callbacks to handle the success or failure (a more procedure-oriented approach) or use functional operations like map, etc. to process the results on success.

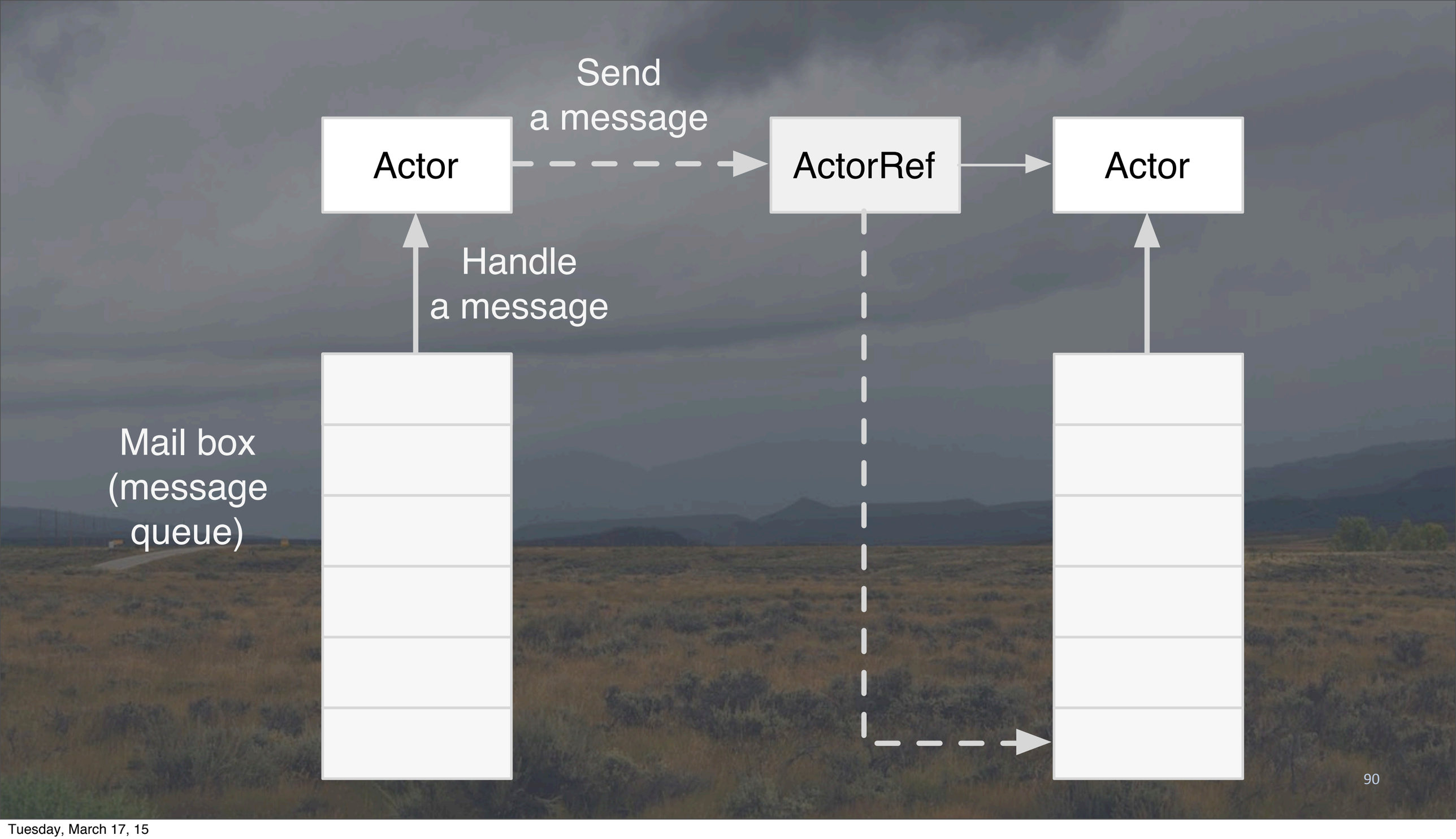
Actor Systems

Let it Crash!

Tuesday, March 17, 15

We briefly visited this before.

Rather than attempt to recover from errors inside the domain logic (e.g., elaborate exception handling), allow services to fail, but with failure detection and reconstruction of those services, plus failover to other replicas.



Tuesday, March 17, 15

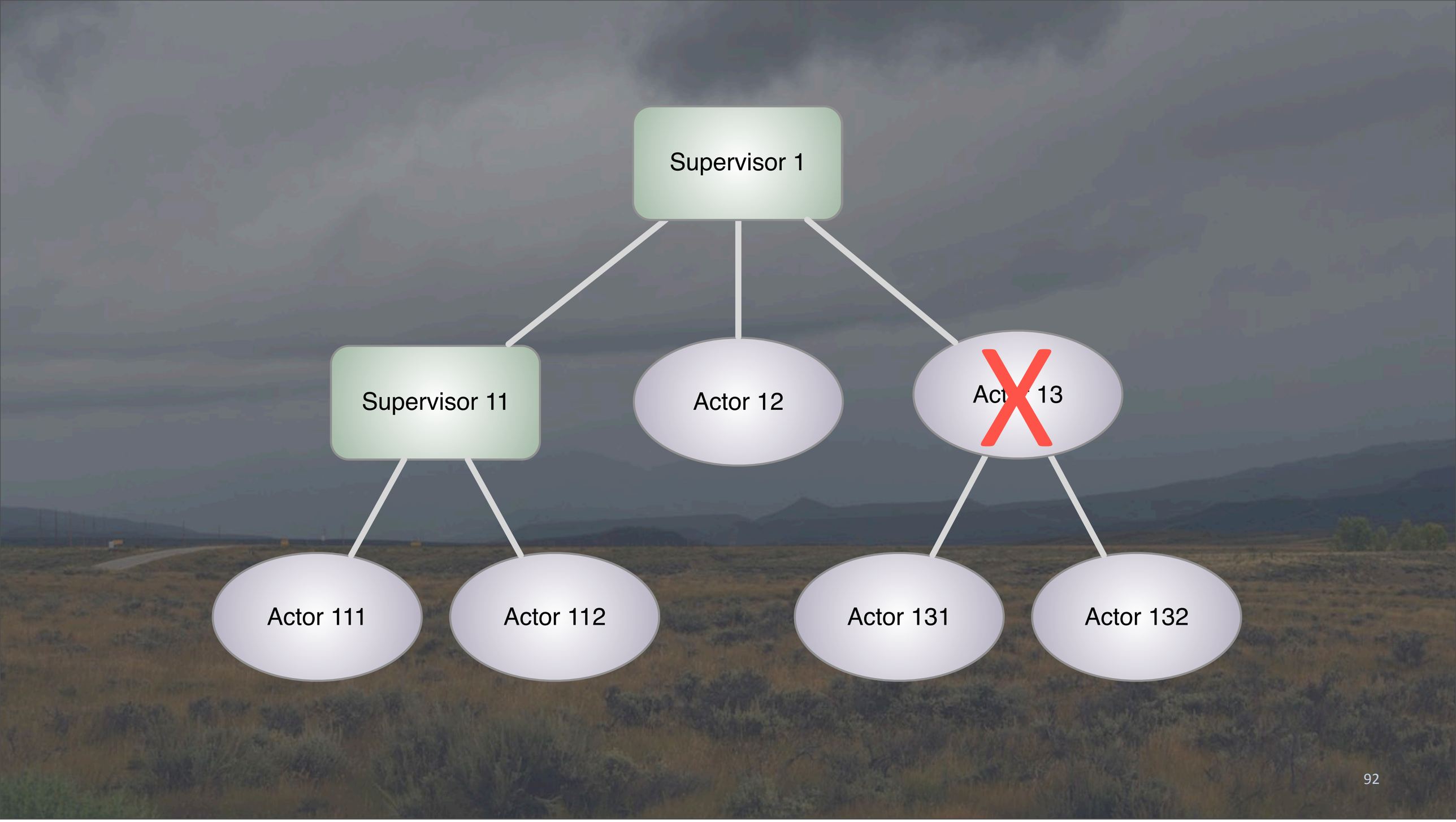
Actors are similar to objects in Smalltalk and similar systems; autonomous agents with defined boundaries that communicate through message passing. Actors, though process each message in a threadsafe way, so they are great for concurrency. (This diagram illustrates the Akka implementation – http://akka.io)

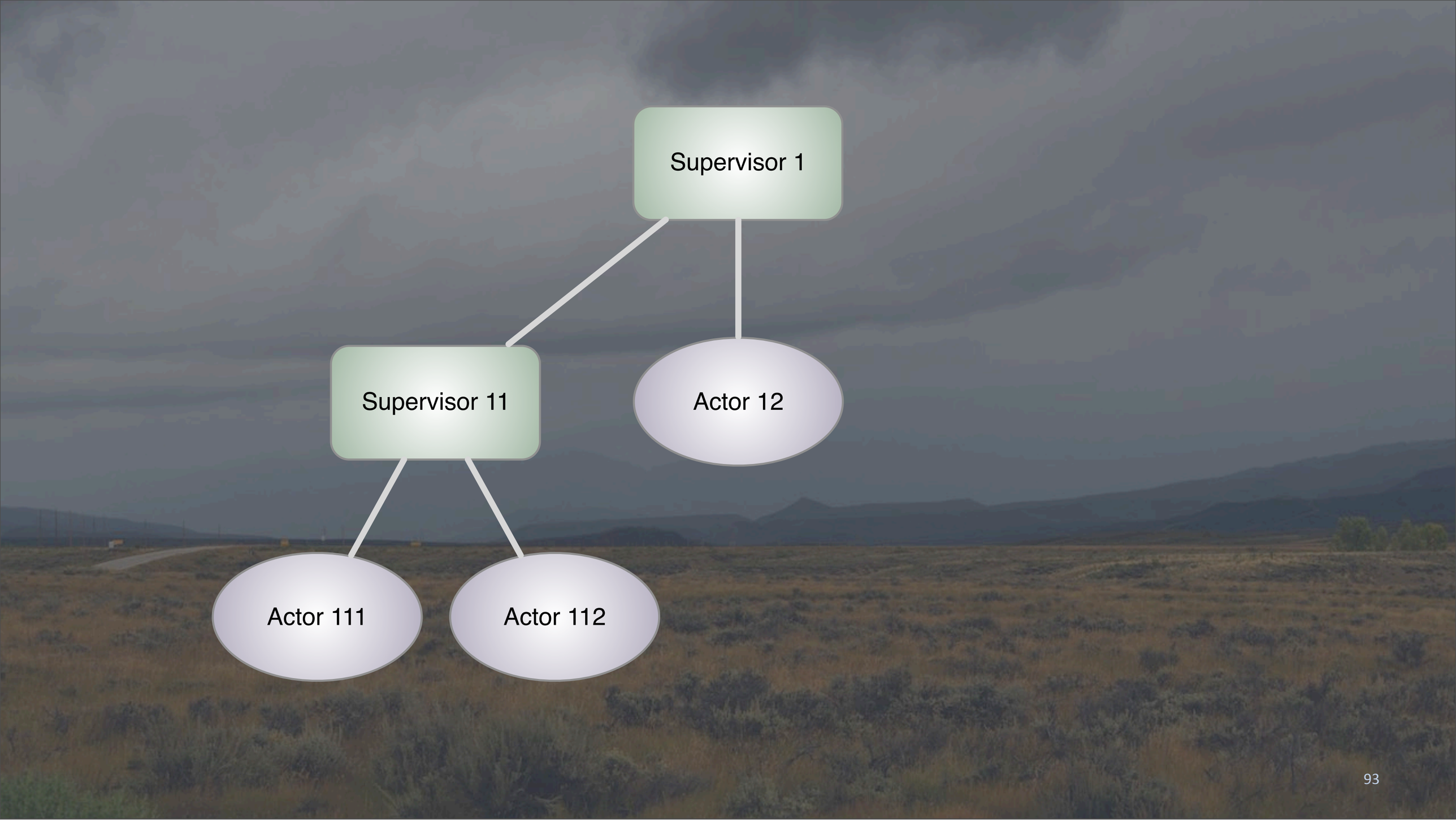
Actor Systems

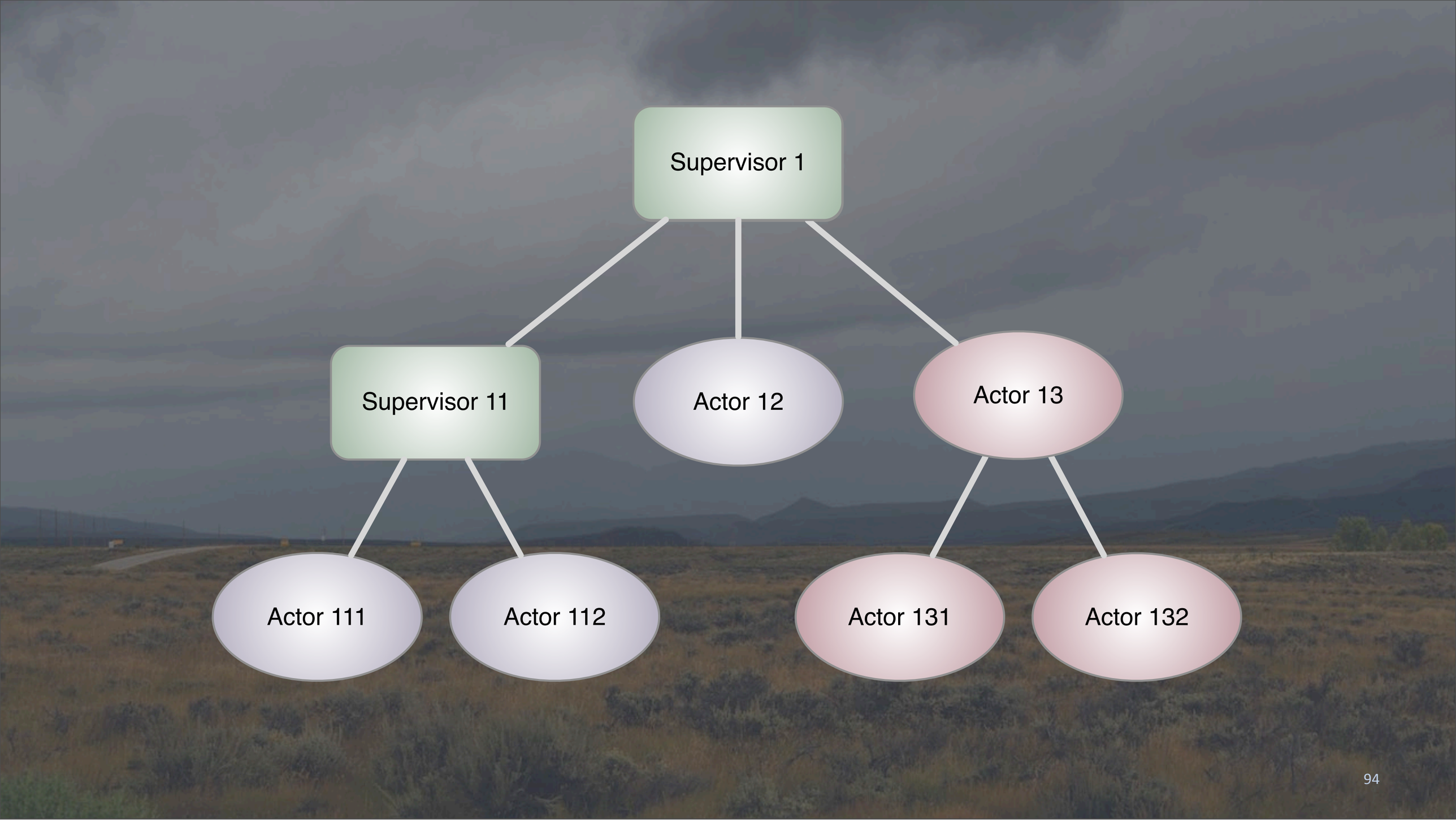
Pioneered by Hewitt, et al. 1973.
Made popular by Erlang,
which introduced Supervision.
Natural distribution model.

Tuesday, March 17, 15

Erlang is a simple language with actor semantics baked in. It has been used to create extremely reliable telecom switches, databases (e.g., Riak), and other services (e.g., GitHub).


Supervisor 1
Supervisor 11
Actor 12
Actor 13
Actor 111
Actor 112
Actor 131
Actor 132


Supervisor 1
Supervisor 11
Actor 12
Actor 111
Actor 112


Supervisor 1
Supervisor 11
Actor 12
Actor 13
Actor 111
Actor 112
Actor 131
Actor 132


The most
sophisticated error recovery
in reactive systems.

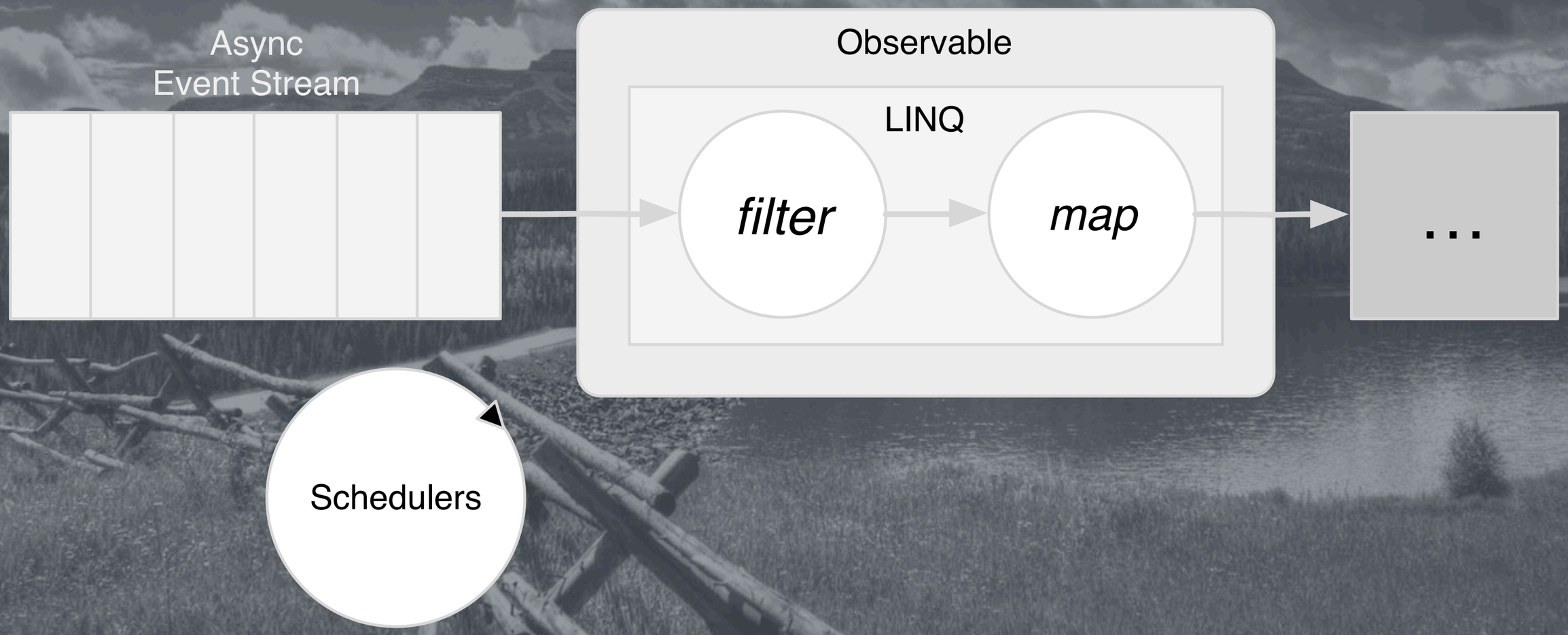

Clean separation
of normal processing
from recovery.

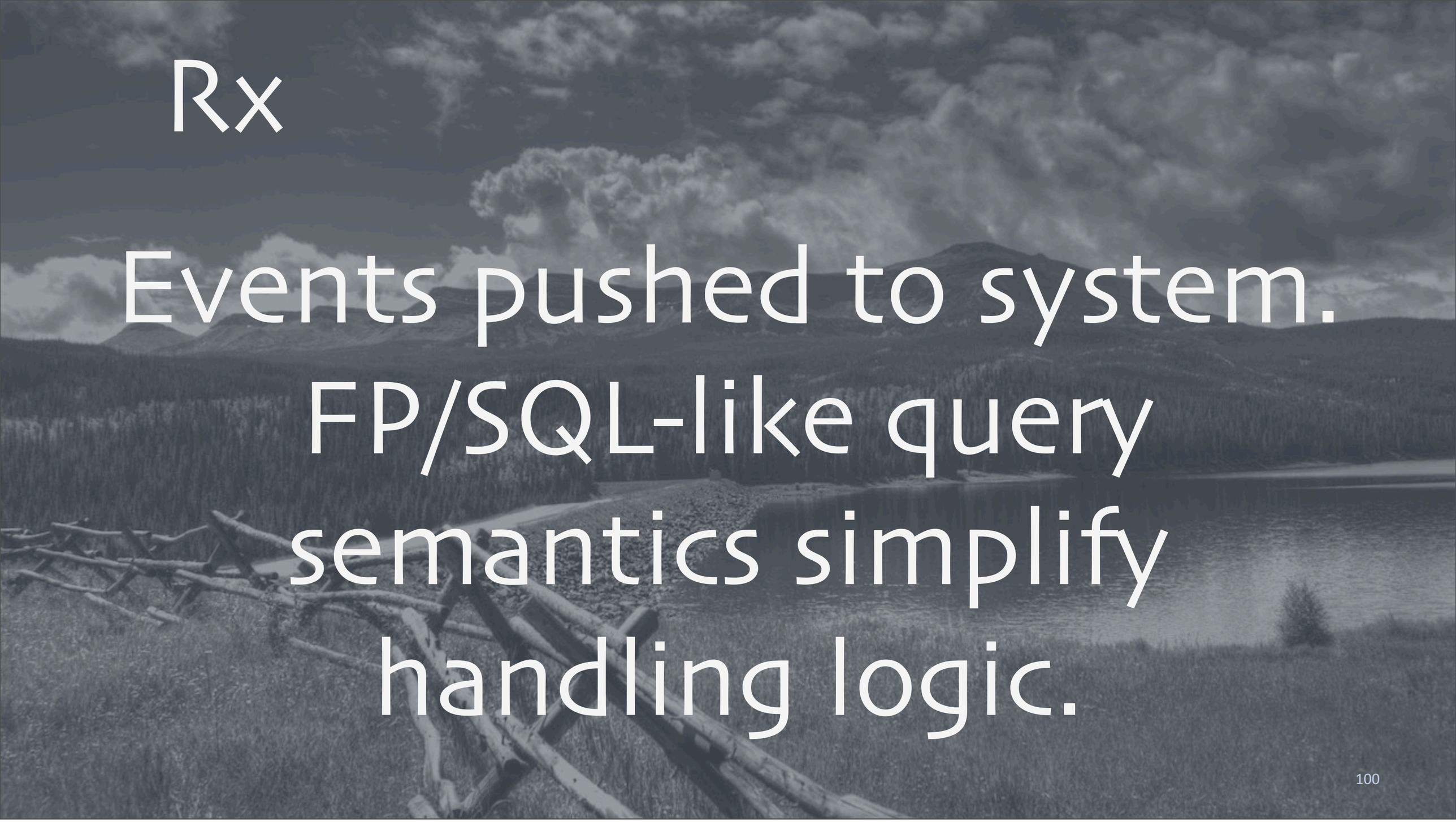
State mutation “firewalls”.
Supports location transparency.

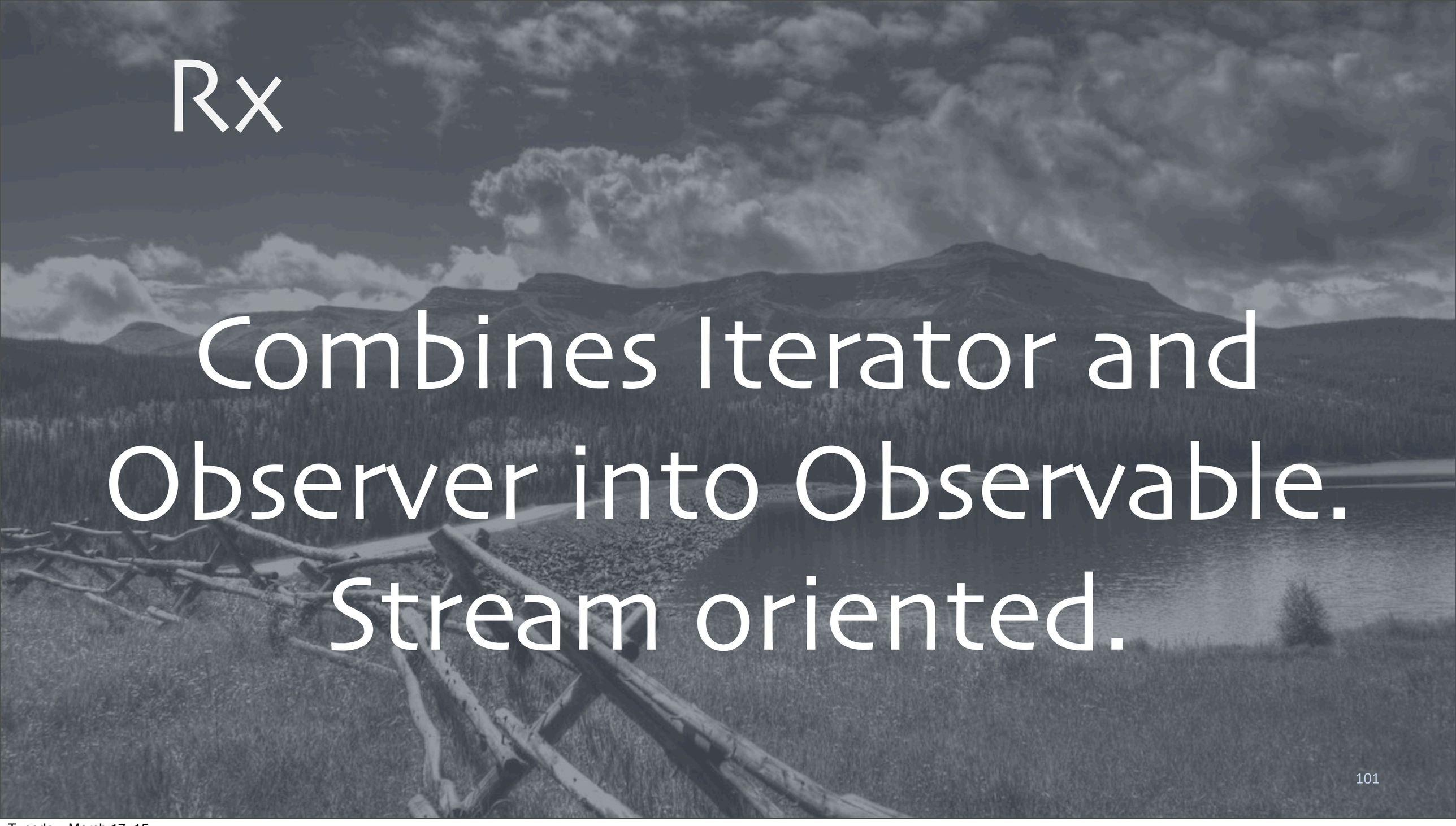
Reactive Extensions (Rx)

Pioneered by Erik Meijer for .NET. Now ported to several languages, including RxJava (Netflix) and React (JS – Facebook).

Rx



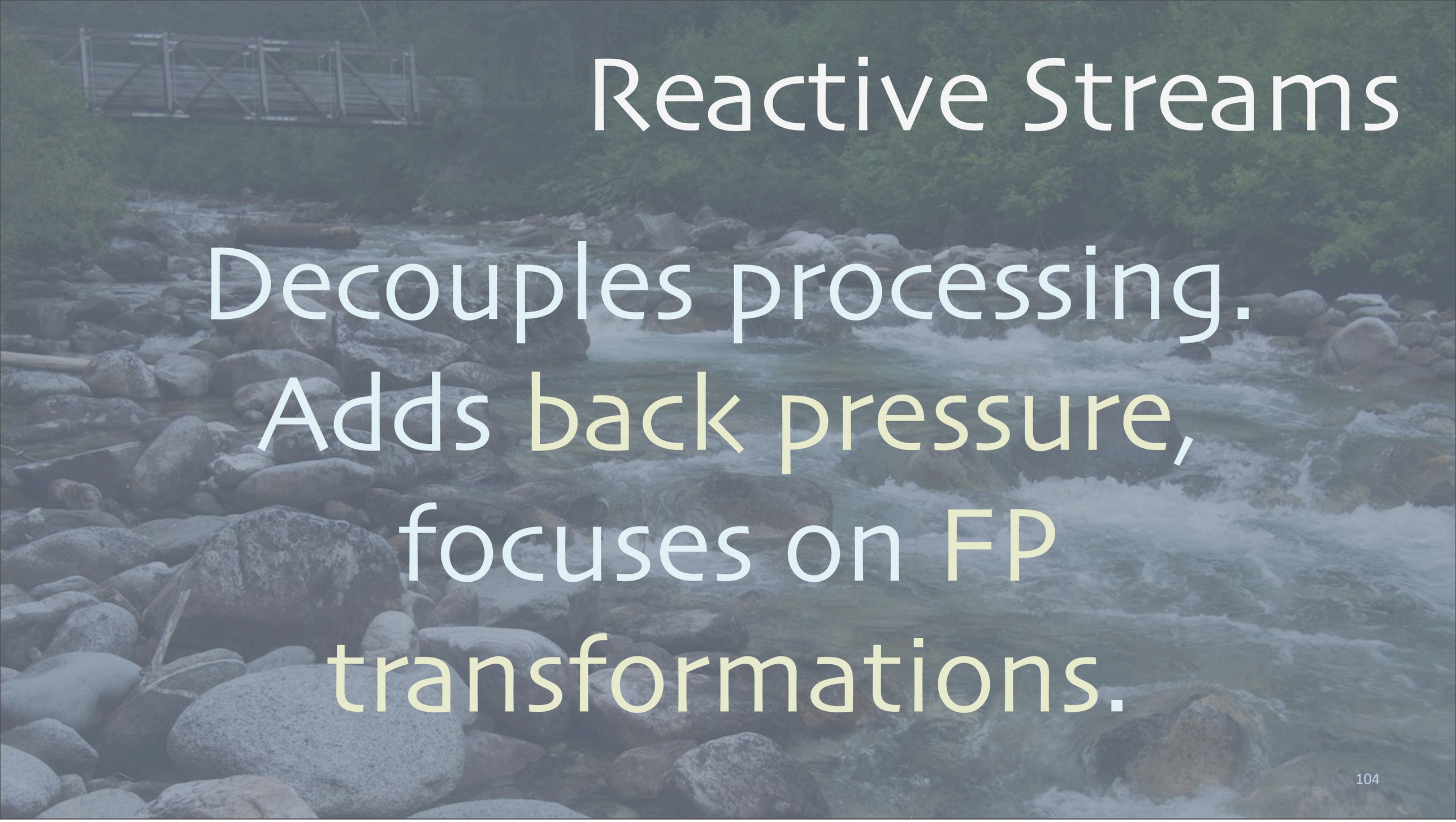

Rx
Events pushed to system.
FP/SQL-like query
semantics simplify
handling logic.


Rx
Combines Iterator and Observer into Observable.
Stream oriented.


Rx
Need to add your own
fault-tolerance model.

Tuesday, March 17, 15

A standard with many implementations for streaming systems with truly “reactive” behavior.
photo: Bridge Creek, North Cascades National Park, Washington State (not Colorado ;)



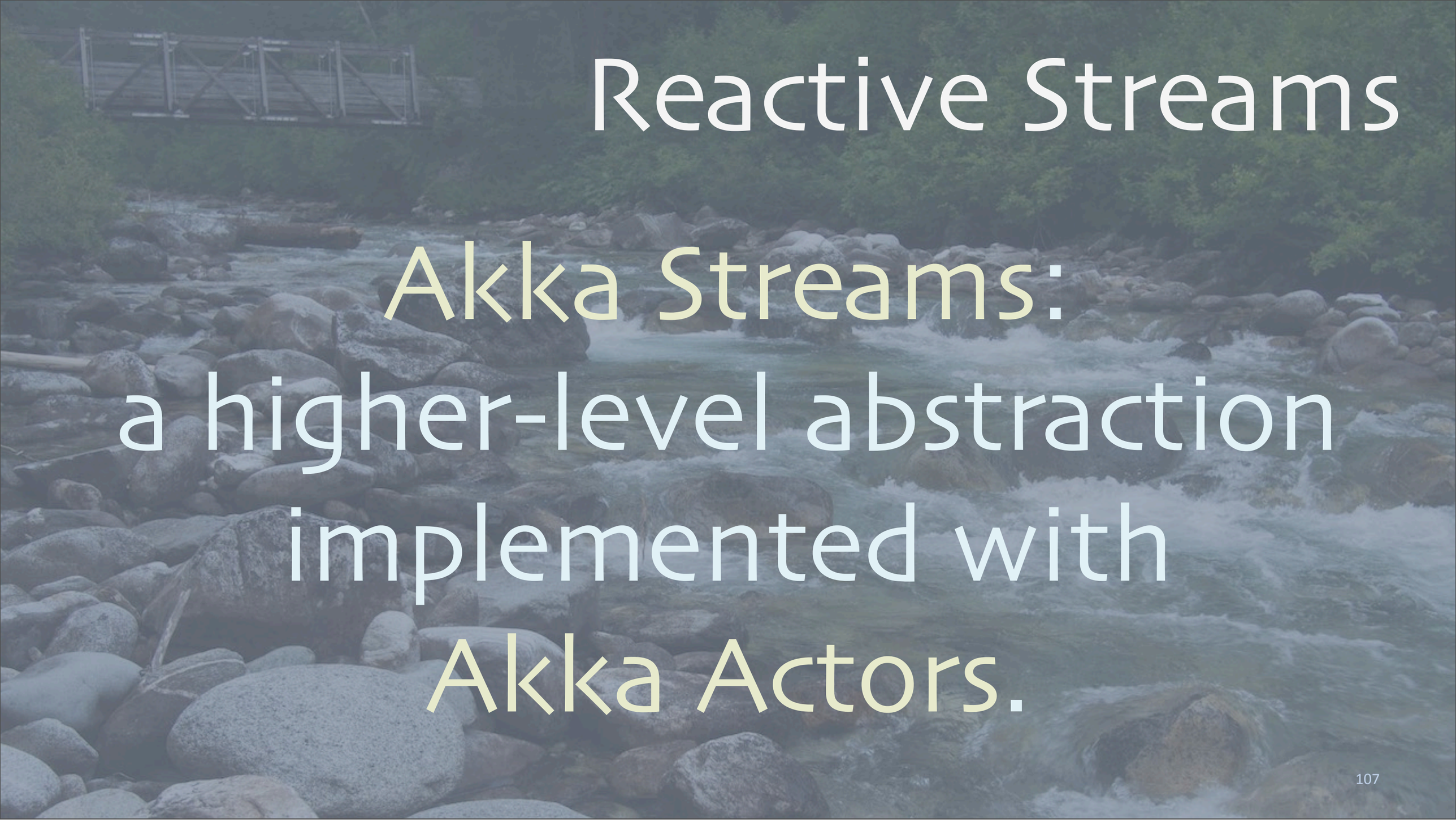
Tuesday, March 17, 15

Back pressure, where the producer and consumer negotiate the flow rate dynamically, is essential for avoiding many failure scenarios, like buffer overflows, ad hoc dropping of messages, etc.

Reactive Streams

Logical extension of Rx.
Focus on possibly
infinite streams of Data.

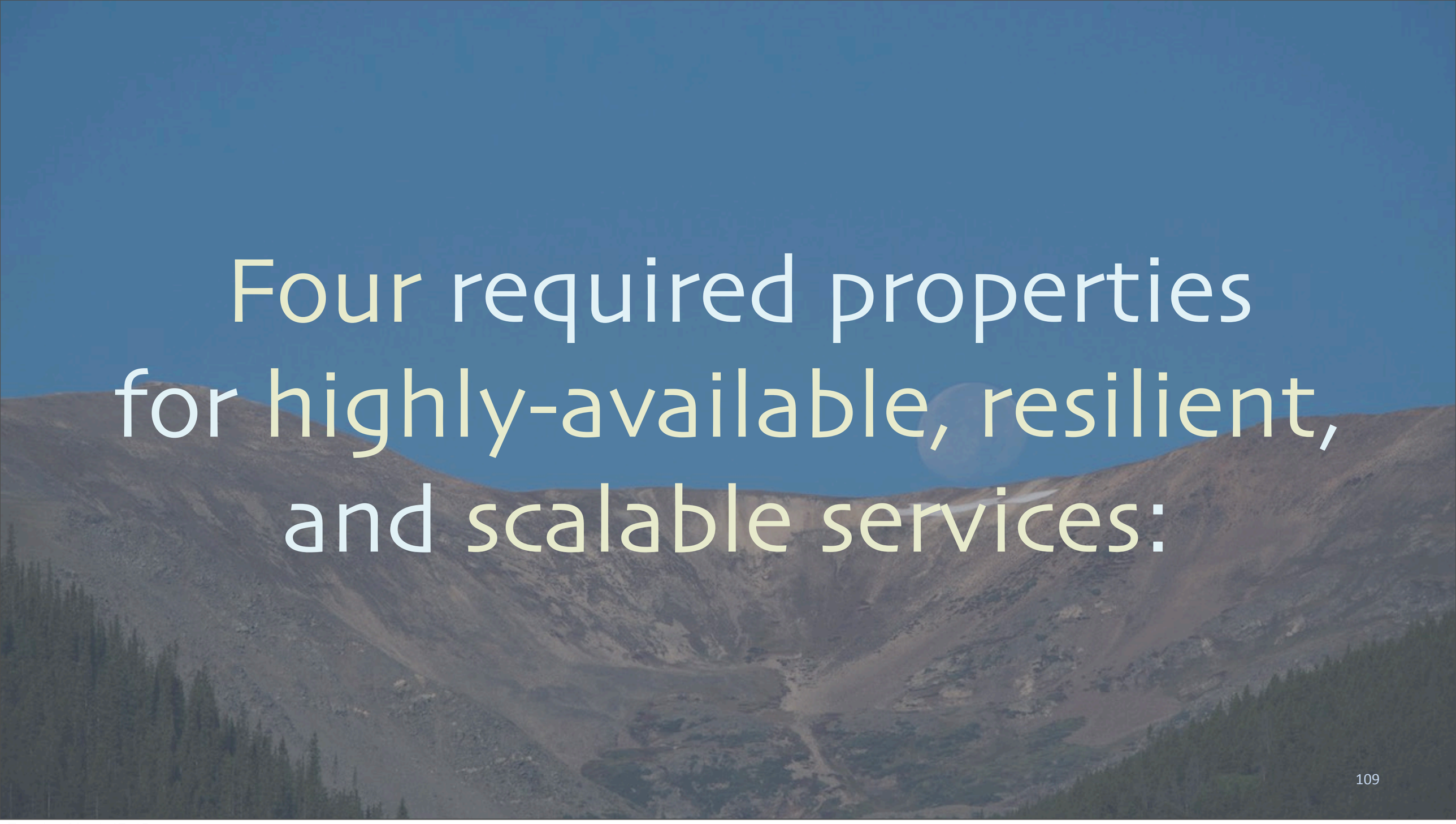
reactive-streams.org

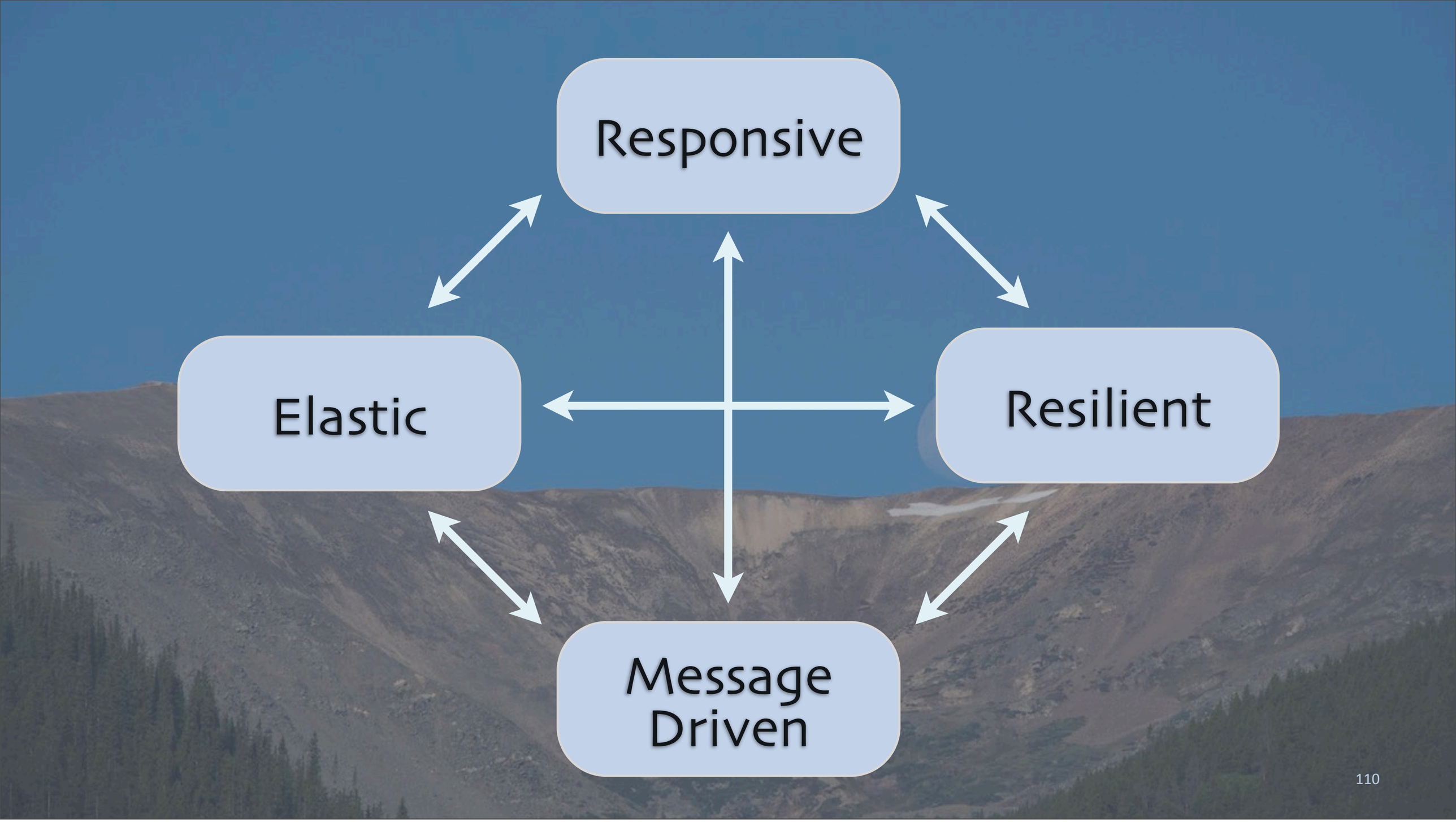


Tuesday, March 17, 15

We're realizing that Actors are a low-level primitive and Actor systems can become unwieldy. Typesafe thinks that higher-level abstractions, like reactive streams, implemented on low-level concurrency systems, like Actors, will be the way to go for most future systems.

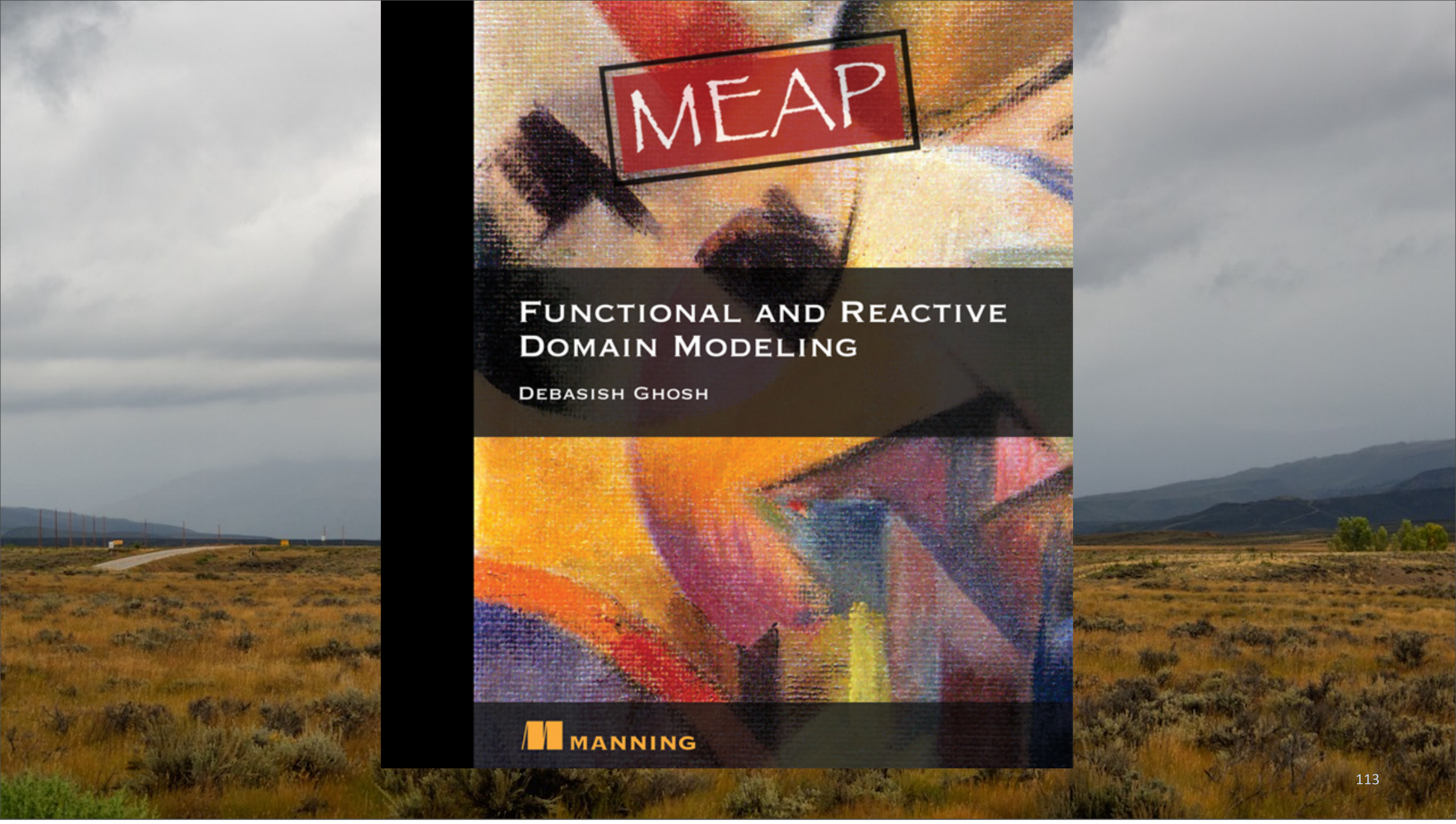
Recap


Four required properties
for highly-available, resilient,
and scalable services:


Responsive
Elastic
Resilient
Message
Driven

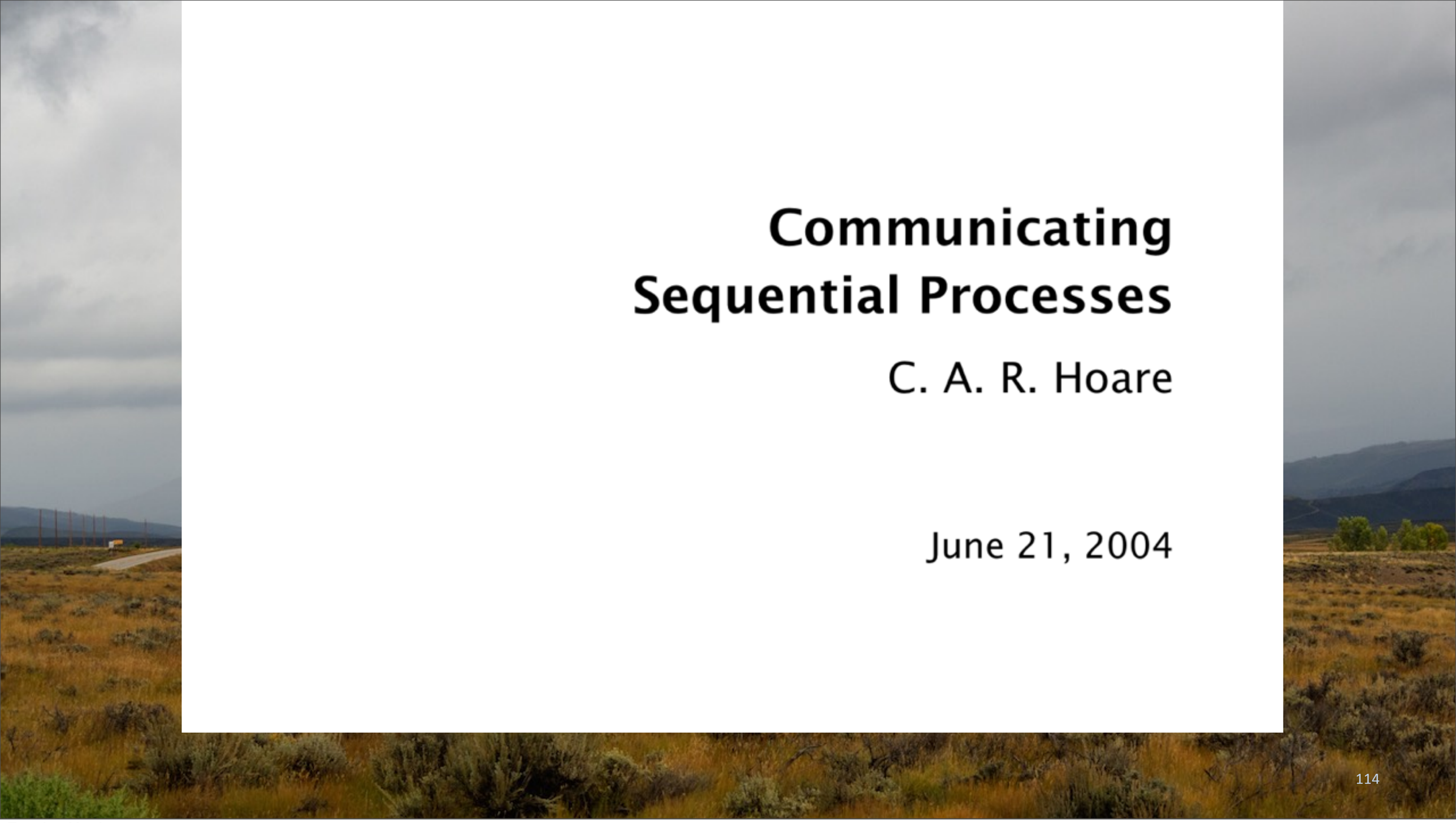
References

See also links earlier in the presentation.



Tuesday, March 17, 15

Lots of interesting practical ideas for combining functional programming and reactive approaches to class Domain-Driven Design by Eric Evans.



Tuesday, March 17, 15

Hoare's book on CSP, originally published in '85 after CSP had been significantly evolved from a programming language to a theoretical model with a well-defined calculus. The book itself has been subsequently refined. The PDF is available for free.

The Theory and Practice of Concurrency

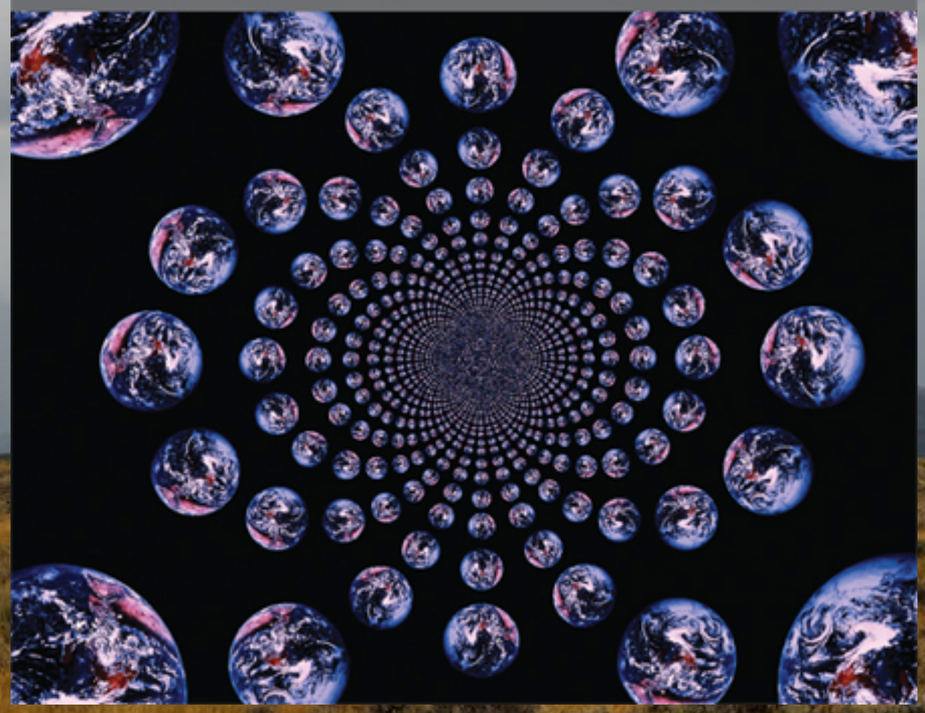
A.W. Roscoe

Published 1997, revised to 2000 and lightly revised to 2005.

The original version is in print in April 2005 with Prentice-Hall (Pearson).

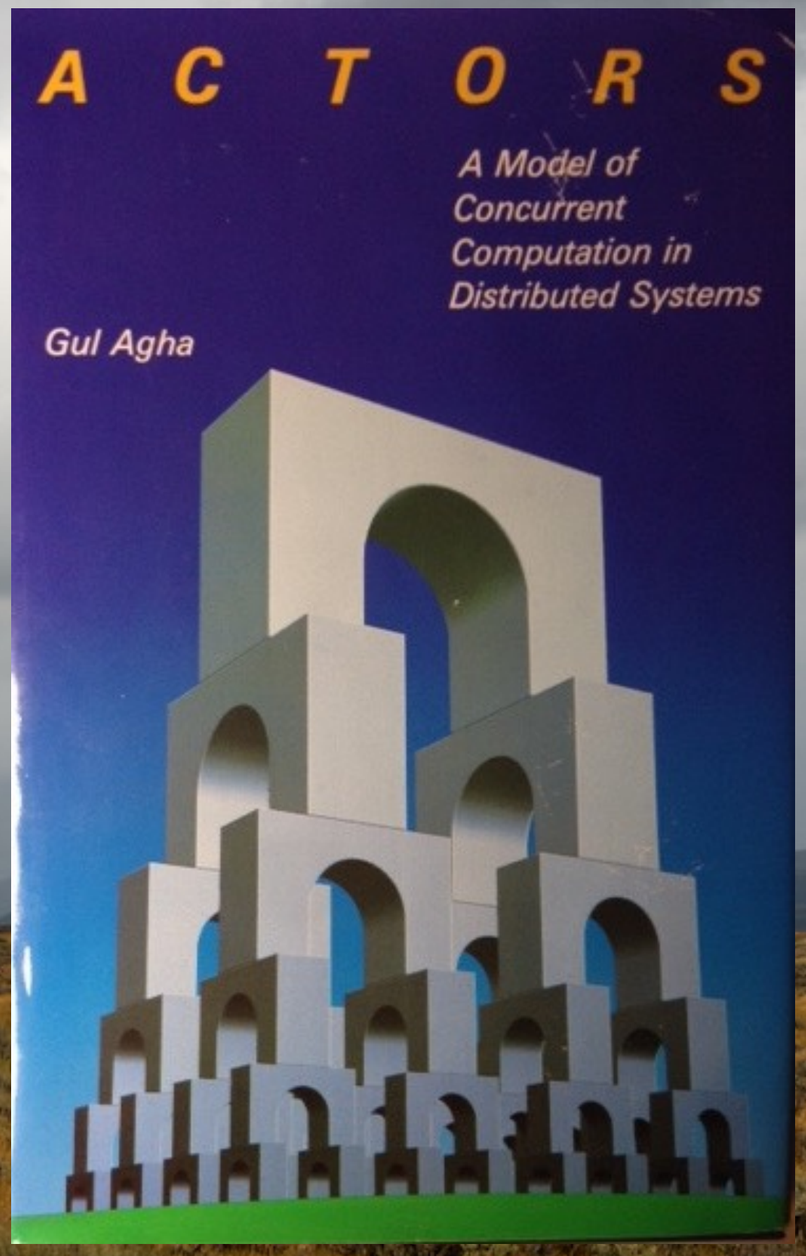
Tuesday, March 17, 15

Modern treatment of CSP. Roscoe helped transform the original CSP language into its more rigorous, process algebra form, which was influenced by Milner's Calculus of Communicating Systems work. This book's PDF is available free. The treatment is more accessible than Hoare's book.



Tuesday, March 17, 15

A survey of theoretical models of distributed computing, starting with a summary of lambda calculus, then discussing the pi, join, and ambient calculi. Also discusses the actor model. The treatment is somewhat dry and could use more discussion of real-world implementations of these ideas, such as the Actor model in Erlang and Akka.

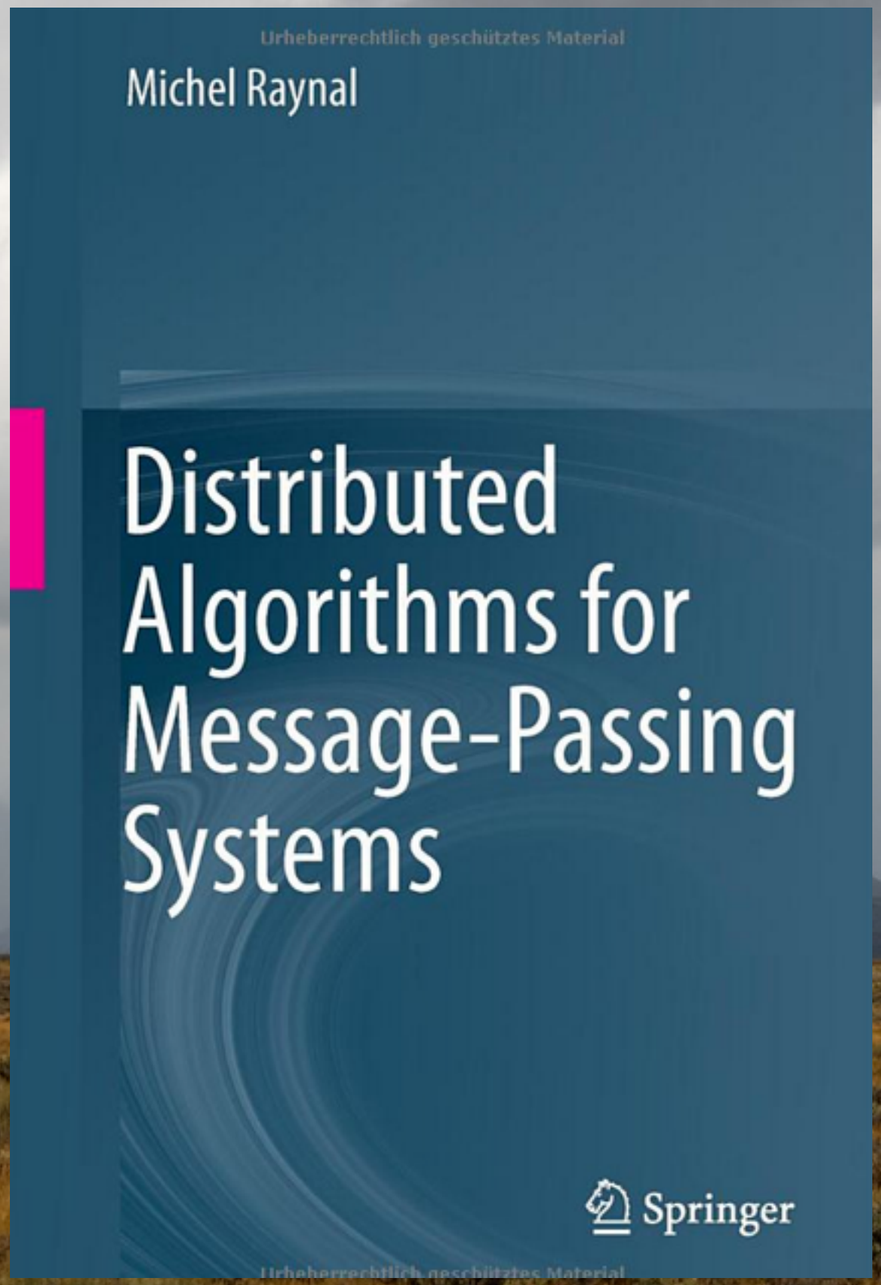


Tuesday, March 17, 15

Gul Agha was a grad student at MIT during the 80s and worked on the actor model with Hewitt and others. This book is based on his dissertation.
It doesn't discuss error handling, actor supervision, etc. as these concepts .

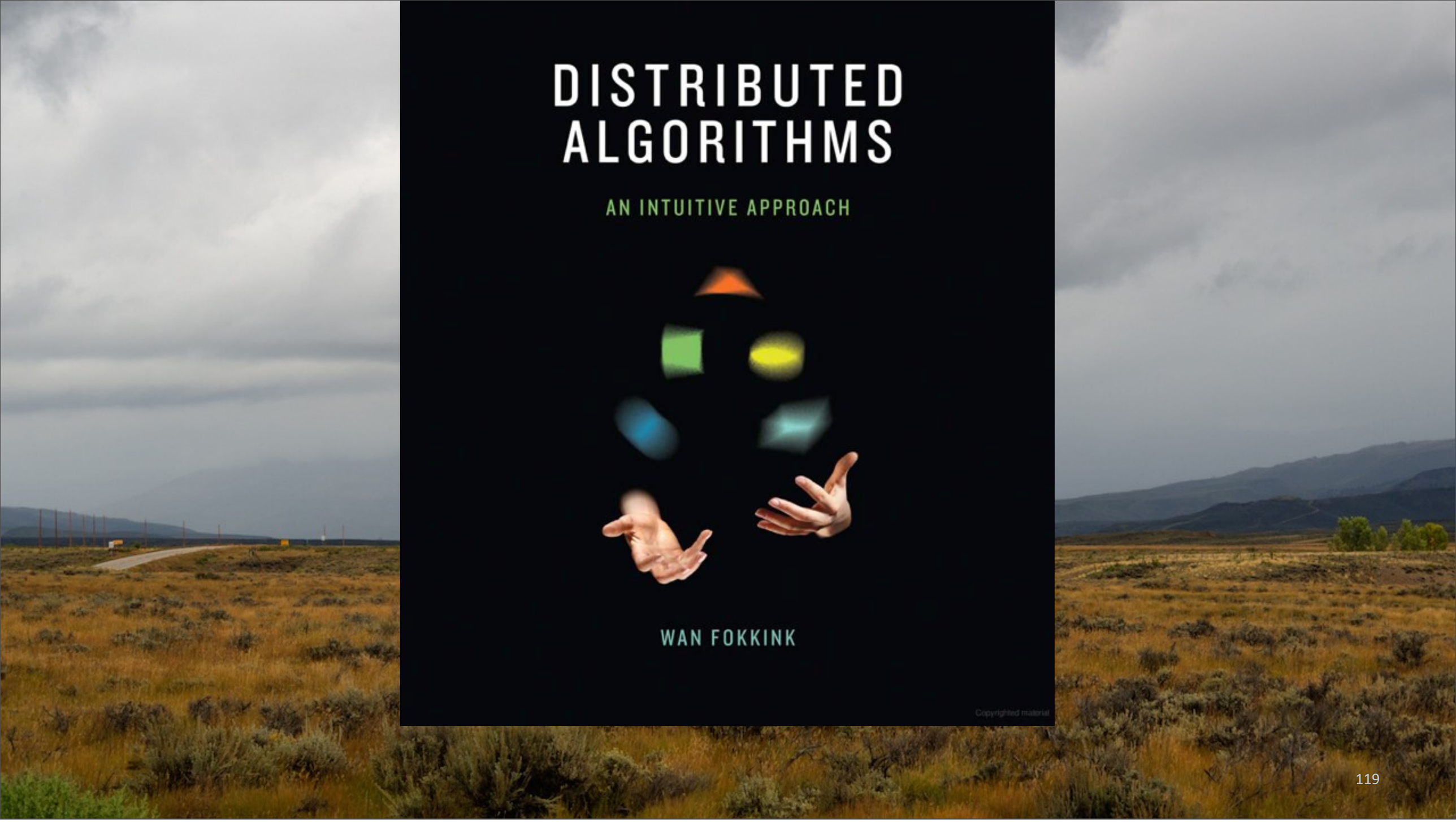
His thesis, http://dspace.mit.edu/handle/1721.1/6952, the basis for his book,http://mitpress.mit.edu/books/actors

See also Paper for a survey course with Rajesh Karmani, http://www.cs.ucla.edu/~palsberg/course/cs239/papers/karmani-agha.pdf



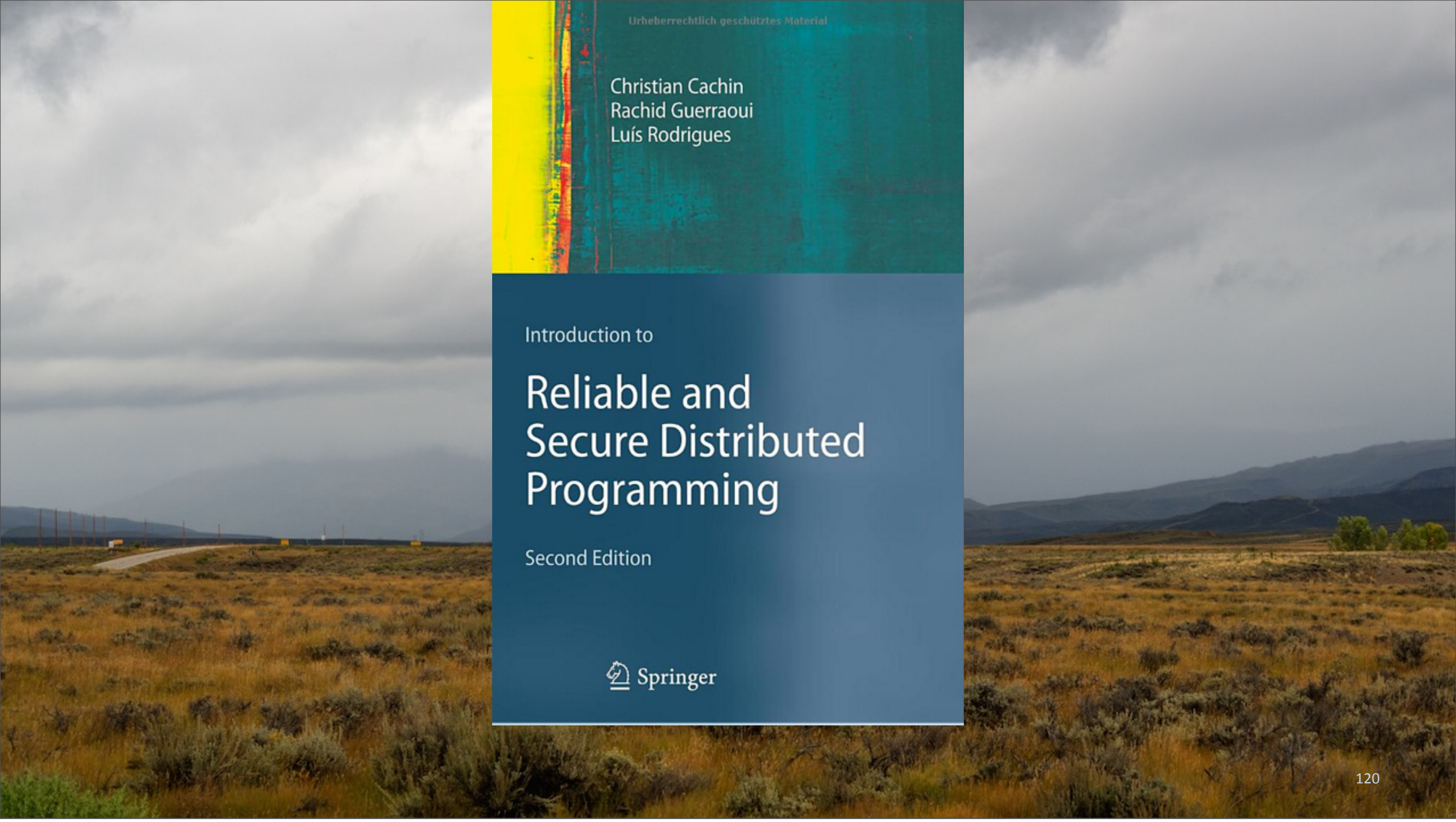
Tuesday, March 17, 15

Survey of the classic graph traversal algorithms, algorithms for detecting failures in a cluster, leader election, etc.



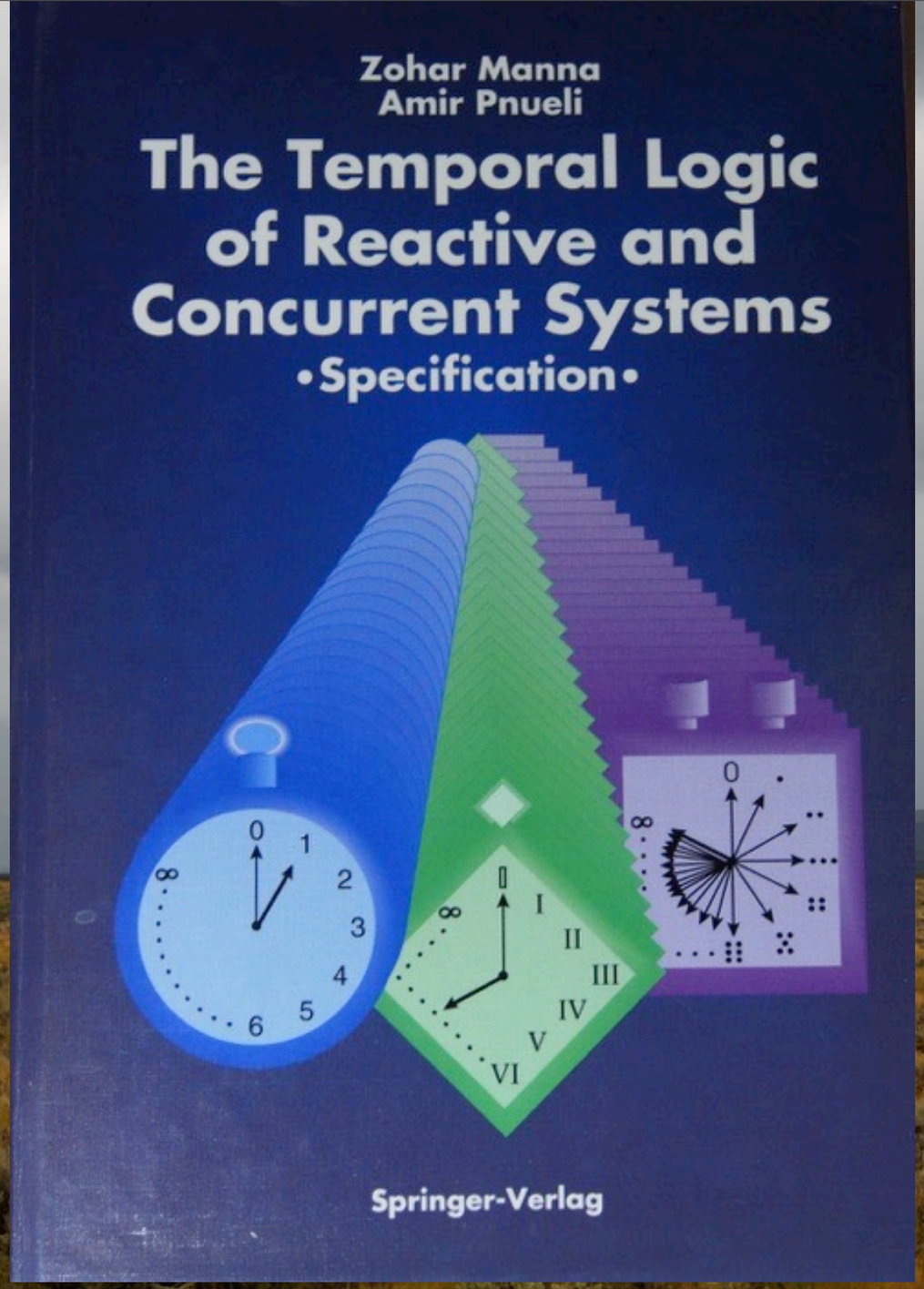
Tuesday, March 17, 15

A less comprehensive and formal, but more intuitive approach to fundamental algorithms.



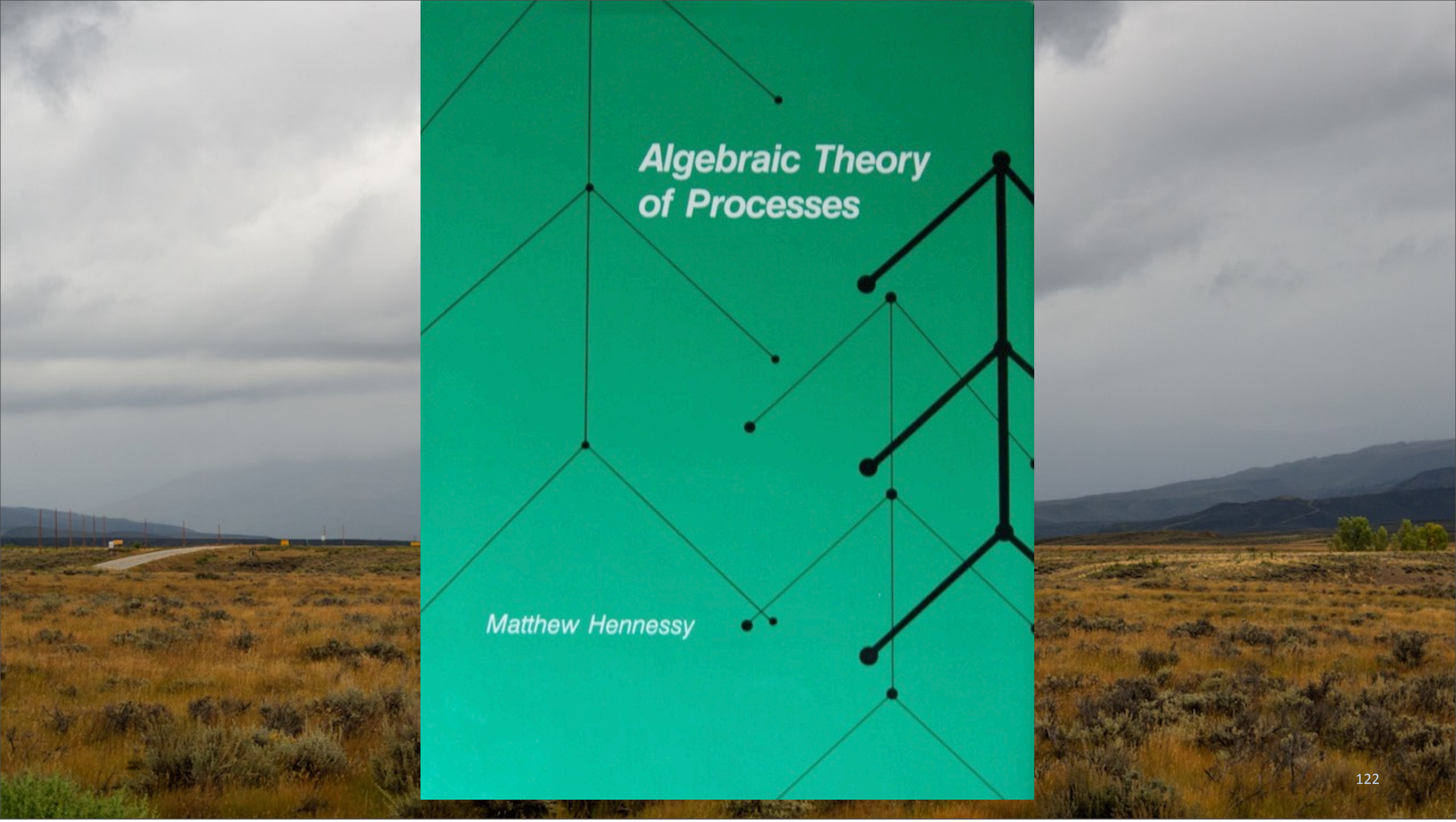
Tuesday, March 17, 15

Comprehensive and somewhat formal like Raynal's book, but more focused on modeling common failures in real systems.



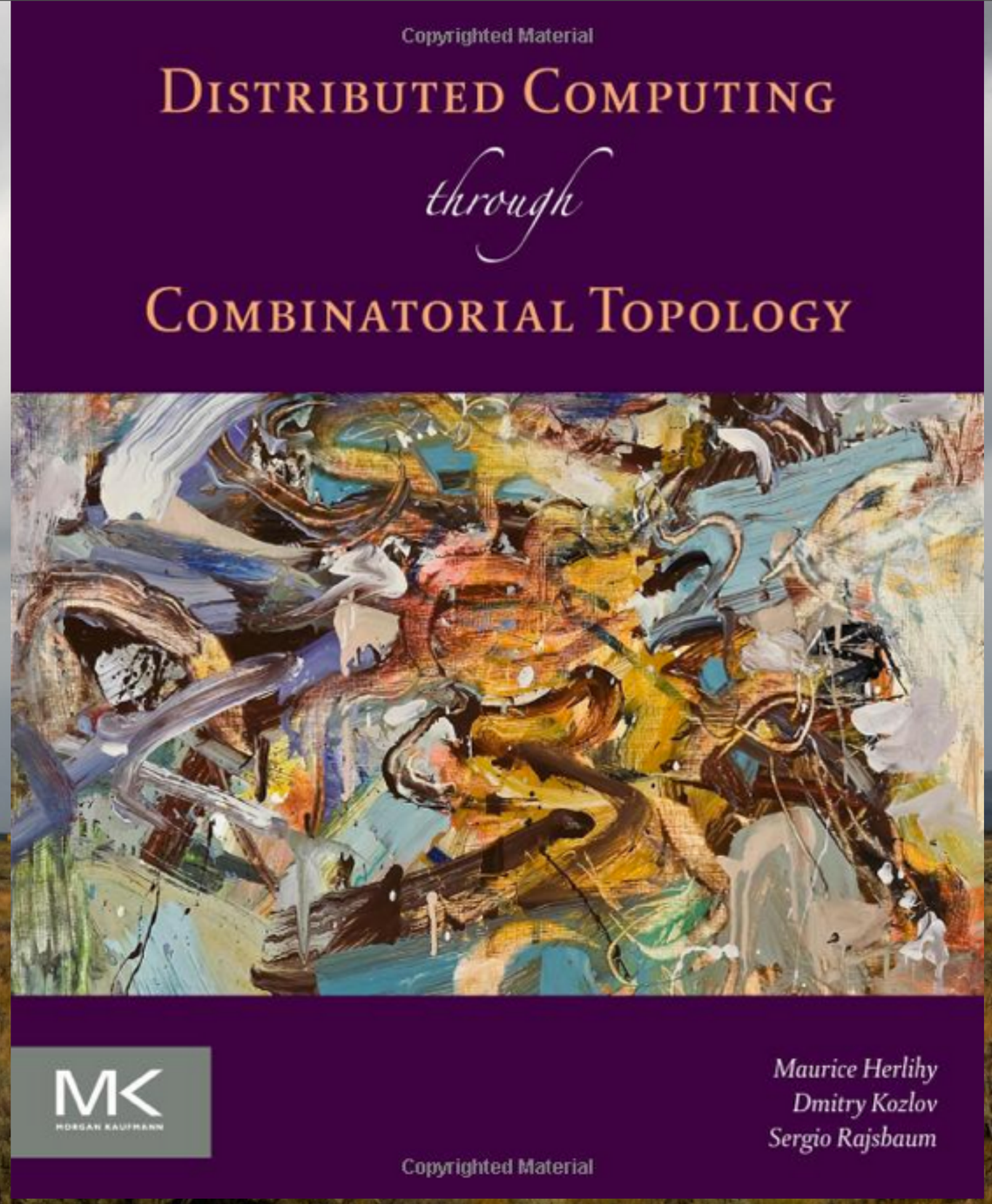
Tuesday, March 17, 15

1992: Yes, “Reactive” isn’t new ;) This book is lays out a theoretical model for specifying and proving “reactive” concurrent systems based on temporal logic. While its goal is to prevent logic errors, It doesn’t discuss handling failures from environmental or other external causes in great depth.



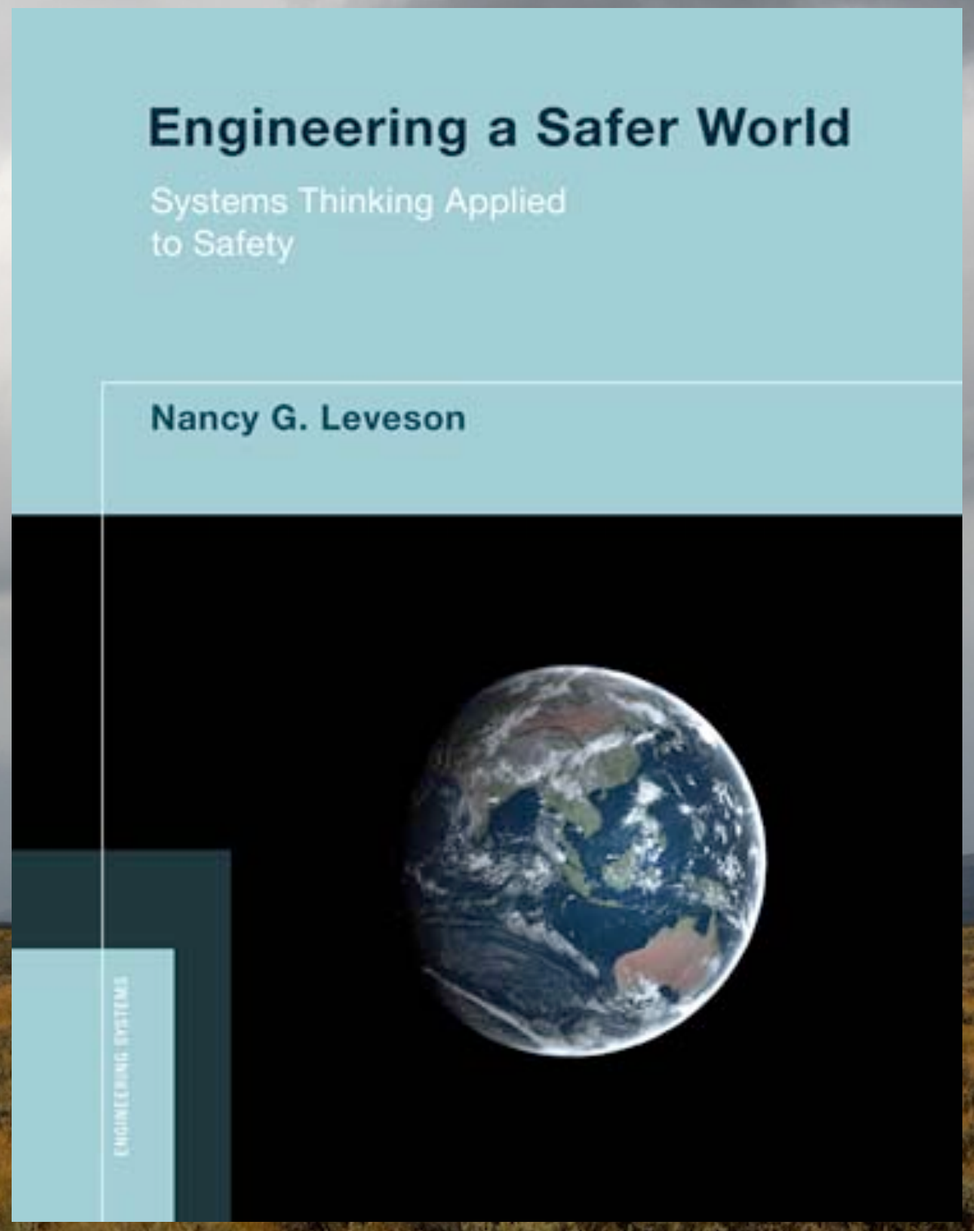
Tuesday, March 17, 15

1988: Another treatment of concurrency using algebra. It's not based on CSP, but it has similar constructs.



Tuesday, March 17, 15

A recent text that applies combinatorics (counting things) and topology (properties of geometric shapes) to the analysis of distributed systems. Aims to be pragmatic for real-world scenarios, like networks and other physical systems where failures are practical concerns.



Tuesday, March 17, 15

http://mitpress.mit.edu/books/engineering-safer-world

Farther afield, this book discusses safety concerns from a systems engineering perspective.