Remote Procedures

Distributed Systems

> A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages. A network of interconnected computers, such as the ARPA net, is a distributed system.

Leslie Lamport. 1978. Time, clocks, and the ordering of events in a distributed system.

Defining characteristic: crashing processes and unreliable communication.

All problems of concurrency and more.

> A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

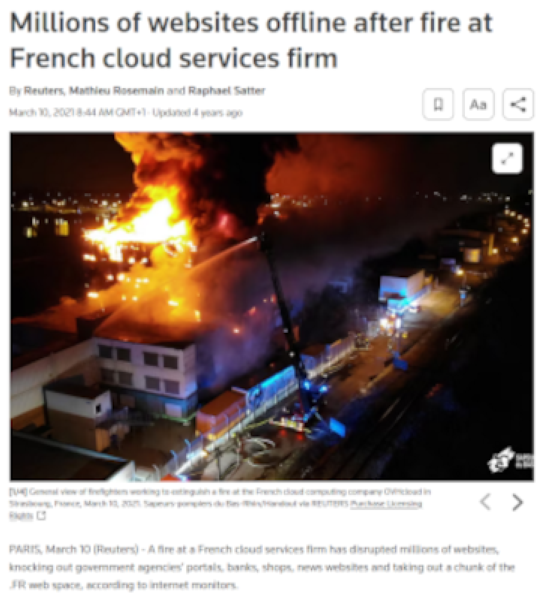
Attributed to Leslie Lamport. 1987.

> Everything fails all the time

Werner Vogels.

Avoid distributed systems!

Failure



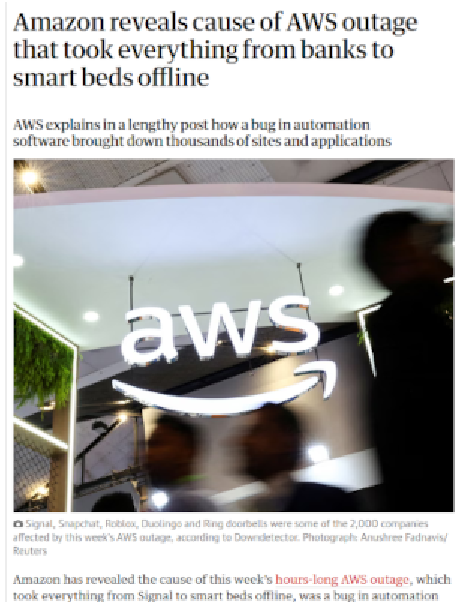
Millions of websites offline after fire at French cloud services firm

By Reuters, Mathieu Rosemain and Raphael Satter

March 10, 2021 8:44 AM GMT+1 · Updated 4 years ago

[1/4] General view of firefighters working to extinguish a fire at the French cloud computing company OVHcloud in Strasbourg, France, March 10, 2021. Sapeurs-pompiers du Bas-Rhin/Handout via REUTERS Purchase Licensing Rights

PARIS, March 10 (Reuters) - A fire at a French cloud services firm has disrupted millions of websites, knocking out government agencies' portals, banks, shops, news websites and taking out a chunk of the .FR web space, according to internet monitors.



Amazon reveals cause of AWS outage that took everything from banks to smart beds offline

AWS explains in a lengthy post how a bug in automation software brought down thousands of sites and applications

Signal, Snapchat, Roblox, Duolingo and Ring doorbells were some of the 2,000 companies affected by this week's AWS outage, according to Downdetector. Photograph: Anushree Fadnavis/Reuters

Amazon has revealed the cause of this week's hours-long AWS outage, which took everything from Signal to smart beds offline, was a bug in automation

https://reuters.com/world/blaze-destroys-servers-europes-largest-cloud-services-firm-2021-03-10/

https://www.theguardian.com/technology/2025/oct/24/amazon-reveals-cause-of-aws-outage

https://blog.cloudflare.com/18-november-2025-outage/

Faults and Failures

Fault: root cause (e.g., hardware dies, software bug, cable broke)

Error: inconsistent internal state (e.g., missing values, index out of bounds)

Failure: system deviates from specification (e.g., unresponsive, wrong)

Machine Faults

Partial fault: some component crashes.

Crash-stop:

- a disk is damaged
- one server crashes
- permanent

Crash-reboot:

- power outage
- network partition
- transient

Byzantine:

- arbitrary, even malicious behavior

Should not lead to system failure!

Network Faults

Messages might be:

- delayed
- omitted
- duplicated
- reordered
- corrupted
- malicious

Should not lead to system failure!

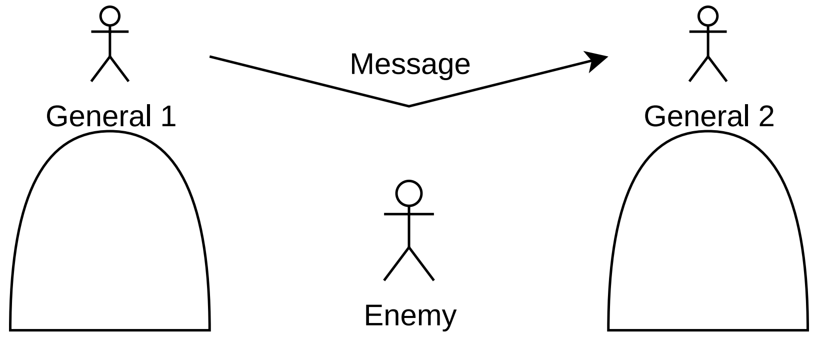
Perfect fault detection is impossible. Can't distinguish between slow and dead.

Two Generals

Two generals must coordinate their attack.

If both attack they win. If only one attacks they both lose.

But messages may get lost.

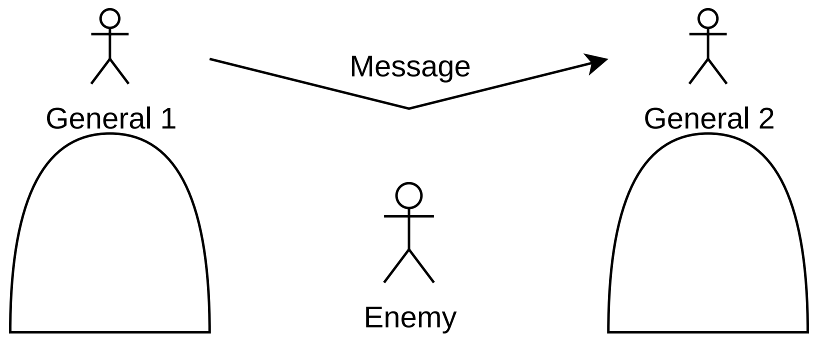


Two Generals

Two generals must coordinate their attack.

If both attack they win. If only one attacks they both lose.

But messages may get lost.



Theoretical problem: perfect agreement over an unreliable channel is impossible.

Practical solution: assume messages eventually arrive after a finite number of retries.

When to use them

Physically distributed

Big data

Big compute

Fault tolerance

Legal requirements

Only whe necessary!

Remote Procedure Calls

Make distributed programming look like normal programming.

Transfer of control and data from one machine to another one and back.

> A principle that we used several times in making design choices is that the semantics of remote procedure calls should be as close as possible to those of local (single-machine) procedure calls.

Andrew D. Birrell and Bruce Jay Nelson. 1984. Implementing remote procedure calls.

gRPC

https://grpc.io

```python
with grpc.insecure_channel("localhost:50051") as channel:
    stub = helloworld_pb2_grpc.GreeterStub(channel)
    response = stub.SayHello(helloworld_pb2.HelloRequest(name="you"))
print("Greeter client received: " + response.message)
```

Created by Google, first released 2015.

demo/helloworld/python/main.py

Interface Definition Language

Define data structures and function signatures once.

Generate compatible code for multiple languages.

Protobuf

```
service Greeter {
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}
```

Created by Google, first released 2008.

https://reasonablypolymorphic.com/blog/protos-are-wrong/

demo/helloworld/greeter.proto

Stubs

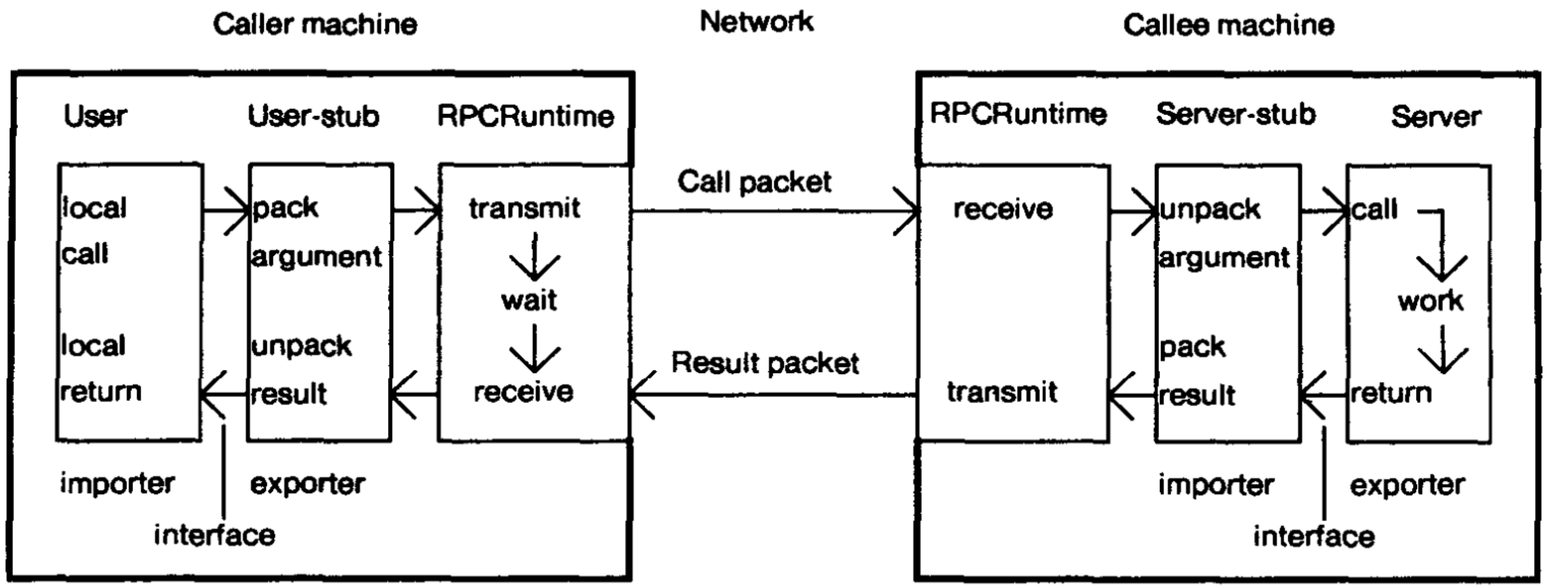


Fig. 1. The components of the system, and their interactions for a simple call.

Andrew D. Birrell and Bruce Jay Nelson. 1984. Implementing remote procedure calls.

demo/helloworld/python/greeter_pb2_grpc.py

demo/helloworld/go/greeter/greeter_grpc.pb.go

Serialization

- Sum types: tagged unions or an emulation.
- Recursive Types: flattened usually depth-first.
- Immutable references: integer identifiers of entities.
- Exceptions: caught and re-raised at the caller.
- Mutable references: do not make sense since address space is not shared.

demo/serialization/types.proto

Request-and-Response versus Fire-and-Forget

> We guarantee that if the call returns to the user then the procedure in the server has been invoked precisely once. Otherwise, an exception is reported to the user and the procedure will have been invoked either once or not at all - the user is not told which.

Andrew D. Birrell and Bruce Jay Nelson. 1984. Implementing remote procedure calls.

Downsides of fire-and-forget:

- no confirmation
- no backpressure

Downsides of request-and-response:

- waiting
- coupling

Timeouts

> Provided the RPCRuntime on the server machine is, still responding, there is no upper bound on how long we will wait for results; that is, we will abort a call if there is a communication breakdown or a crash but not if the server code deadlocks or loops. This is identical to the semantics of local procedure calls.

Andrew D. Birrell and Bruce Jay Nelson. 1984. Implementing remote procedure calls.

Network-level timeout:

- how long we wait for the network
- based on Round Trip Time (RTT)

Application-level timeout:

- how long we wait for the response
- based on Service Level Agreement (SLA)

In theory, no application-level timeout is needed. In practice, you should set one!

demo/helloworld/python/timeout.py

Retries

> In addition to exceptions raised by the callee, the RPCRuntime may raise a call failed exception if there is some communication difficulty. This is the primary way in which our clients note the difference between local and remote calls.

Andrew D. Birrell and Bruce Jay Nelson. 1984. Implementing remote procedure calls.

Network errors (unavailable, timeout): retry!

Other errors (invalid argument, version mismatch, key not found): do not retry!

Problems with Retries

Thundering Herd: synchronized retries

- Solution: jitter (randomly delay retry)

Retry Storm: retries overload already-failing server

- Solution: exponential backoff (wait longer and longer for retry)
- Solution: circuit breaker (stop retrying after threshold, resume later)

Retry Amplification: retries multiply through call chain

- Solution: retry budgets (limit percentage of requests that are retries)
- Solution: limited depth (don't retry if you already are in a retry)

demo/helloworld/python/retries.py

Summary and Outlook

Remote procedures are simple and useful.

Next week: Replicated Data