



Debugging distributed systems

Bert Jan Schrijver
bertjan@openvalue.eu





~~Networking 101~~

How the internet works

~~Debugging distributed systems~~

Bert Jan Schrijver
bertjan@openvalue.eu

 @bjschrijver

Why?



Let's meet
Bert Jan Schrijver



● OPENVALUE

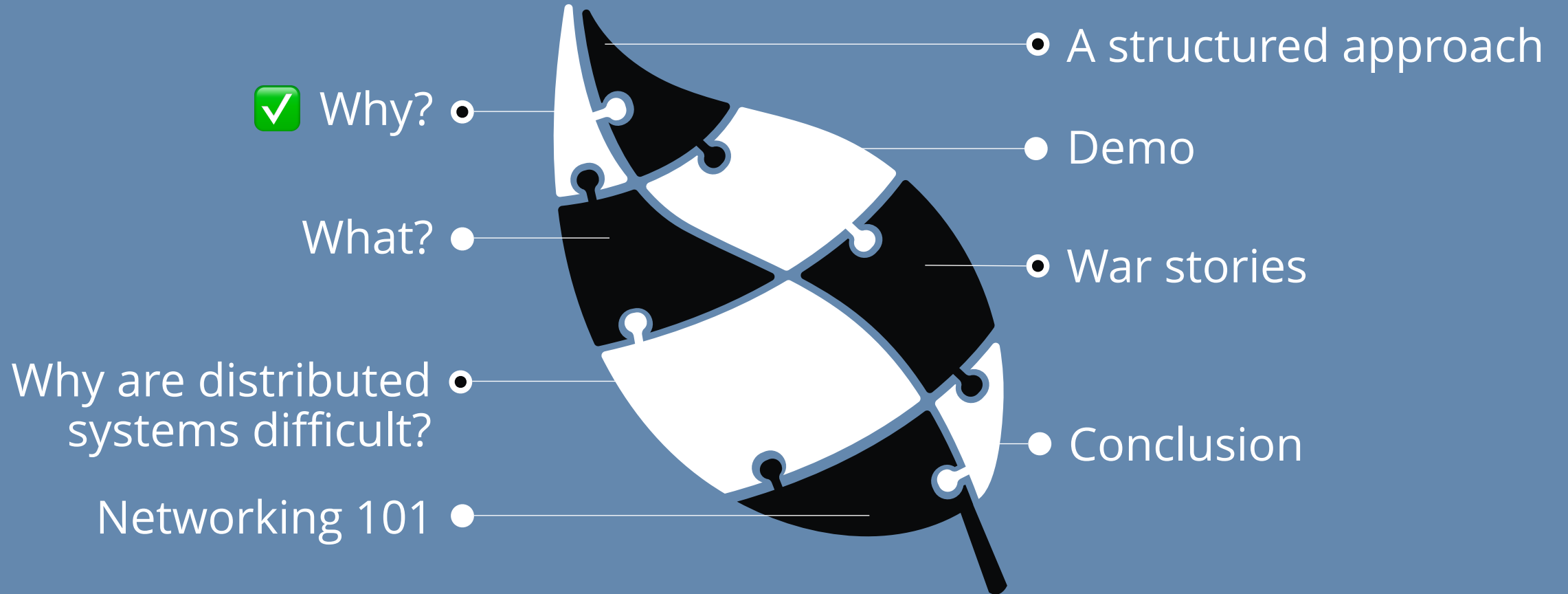
● .nl.
jug



 @bjschrijver

What's next?

Outline



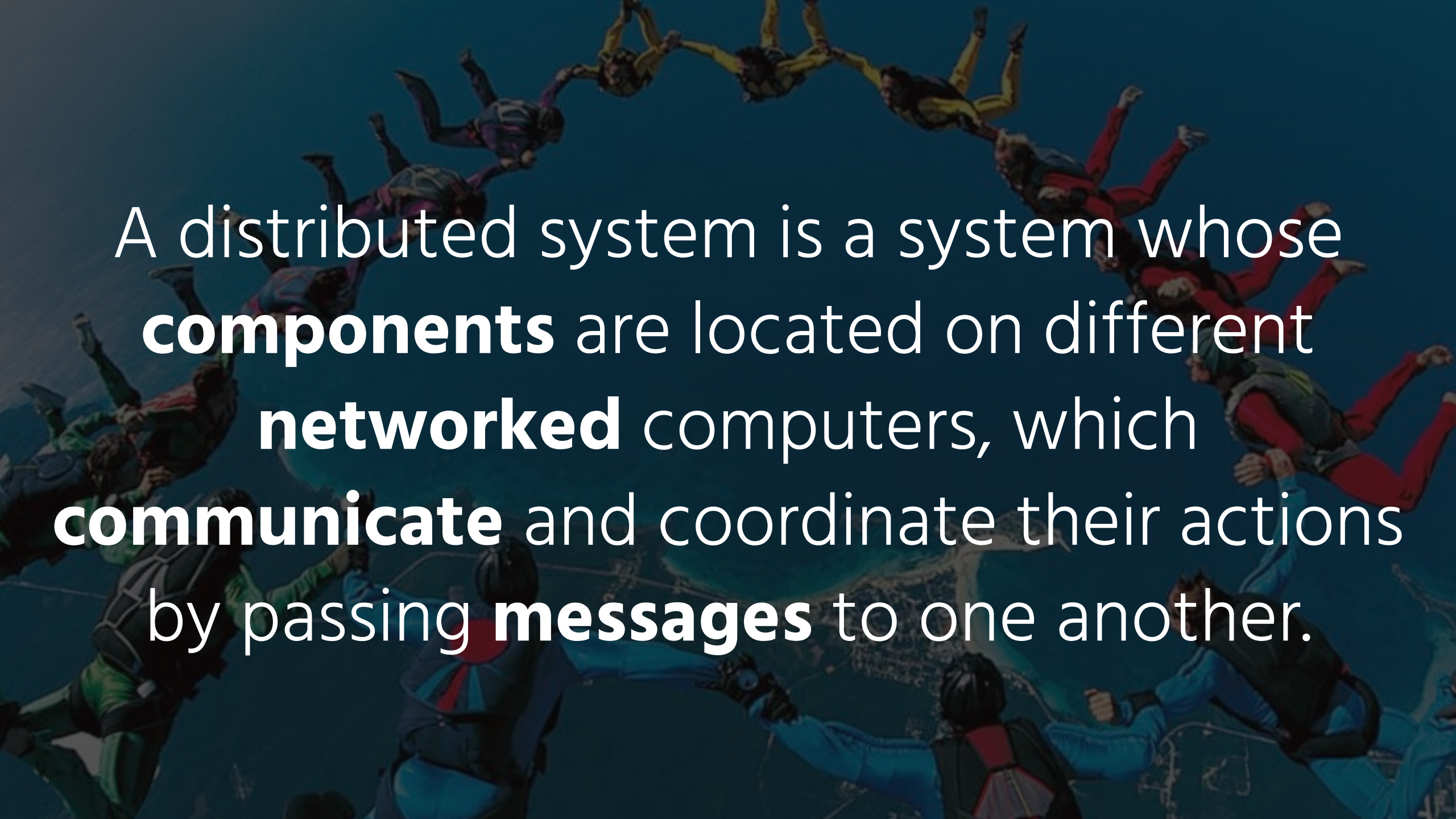
The hierarchy of complexity for debugging

- Level 1: Non-concurrency
- Level 2: Concurrency
- Level 3: Distribution



A photograph of a basketball game in progress, featuring several San Antonio Spurs players on the court. The players are wearing white jerseys with "SPURS" and their numbers (15, 14, 4, and 45) visible. The player with number 45, Blair, is in the foreground, kneeling on the court. Other players are standing around him. The background shows a large crowd of spectators in the arena stands. The text "What is a distributed system?" is overlaid in white on the image.

What is a distributed system?



A distributed system is a system whose **components** are located on different **networked** computers, which **communicate** and coordinate their actions by passing **messages** to one another.

Characteristics of distributed systems

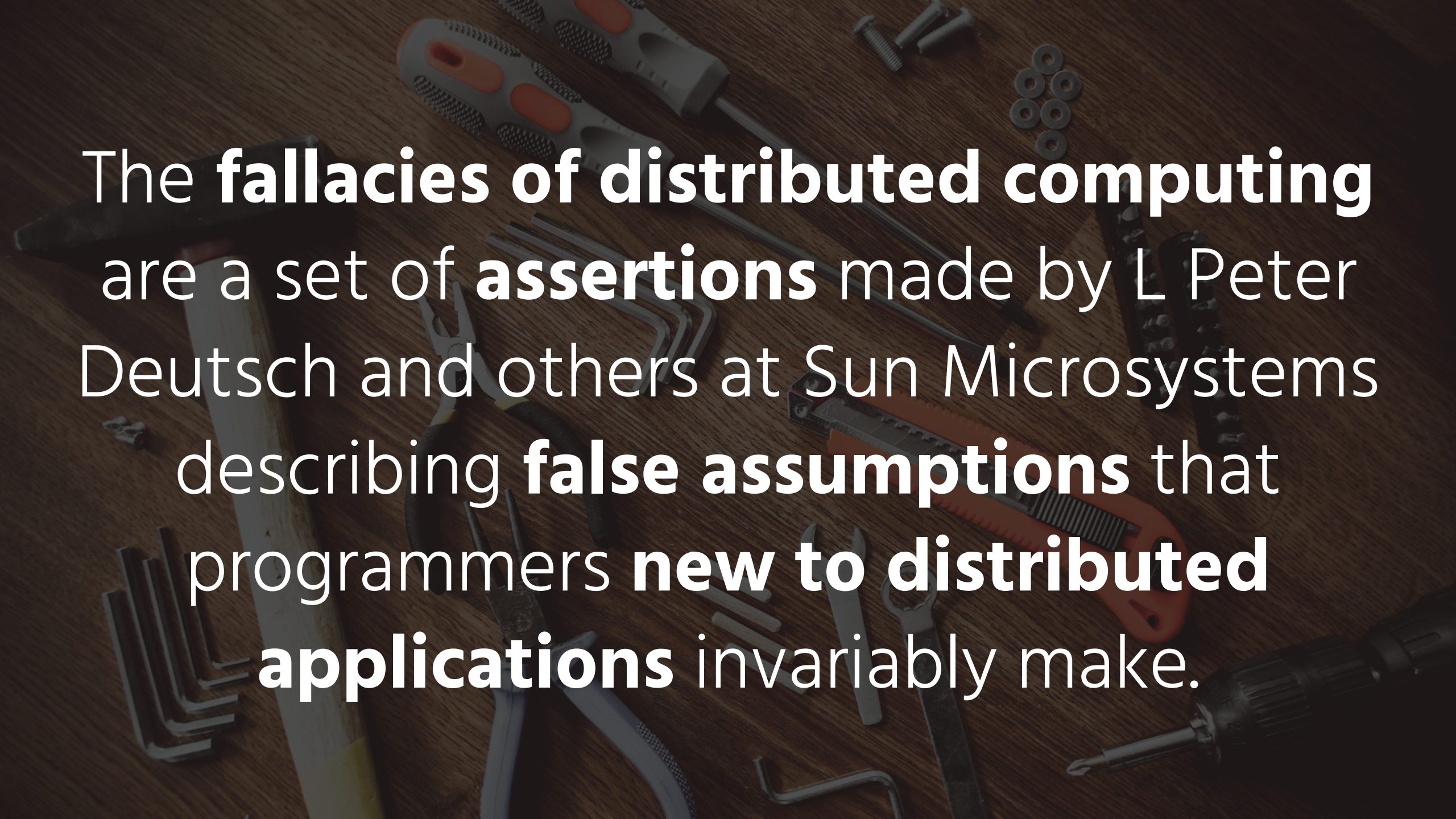
- Concurrency of components
- Lack of a global clock
- Independent failure of components
- Distributed systems are harder to reason about

A photograph of two skateboarders on a paved road in a desert landscape. One skateboarder in a pink shirt is riding away from the camera, while another in a yellow shirt is performing a trick, with one foot on the board and the other leg extended. The background shows a vast, arid landscape under a clear sky.

“ Working with distributed systems is **fundamentally** different from writing software on a single computer - and the main difference is that there are lots of new and exciting ways for things to go **wrong.**”
- Martin Kleppmann

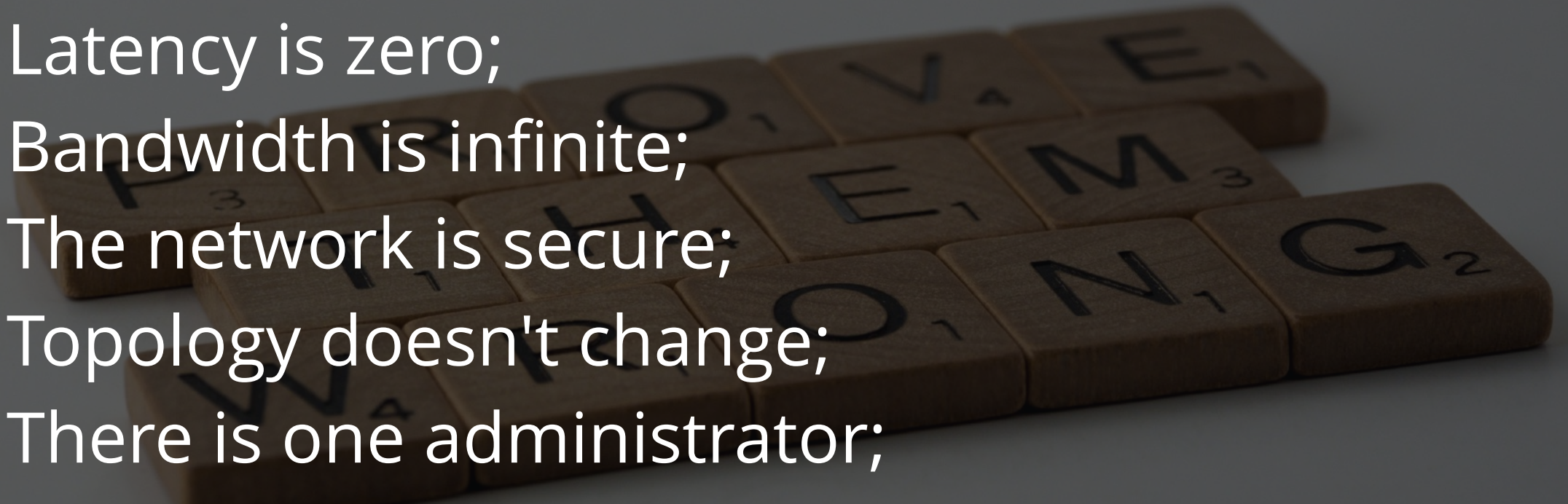
A photograph of two skateboarders on a desert road. The skateboarder on the left is wearing a pink t-shirt and dark pants, riding smoothly. The skateboarder on the right is wearing a yellow t-shirt and blue jeans, falling off their board. The background shows a vast, arid landscape with rolling hills and a clear sky. The text "Why do things go wrong?" is overlaid in the center of the image.

“ Why do things go wrong? ”

The background of the image is a wooden workbench with various tools scattered across it. Visible tools include several screwdrivers with different handle designs (some orange and black, some grey), a pair of pliers, several hex keys (Allen keys) of different sizes, and a small cluster of grey washers or nuts. The lighting is somewhat dim, creating a professional and technical atmosphere.

The **fallacies of distributed computing** are a set of **assertions** made by L Peter Deutsch and others at Sun Microsystems describing **false assumptions** that programmers **new to distributed applications** invariably make.

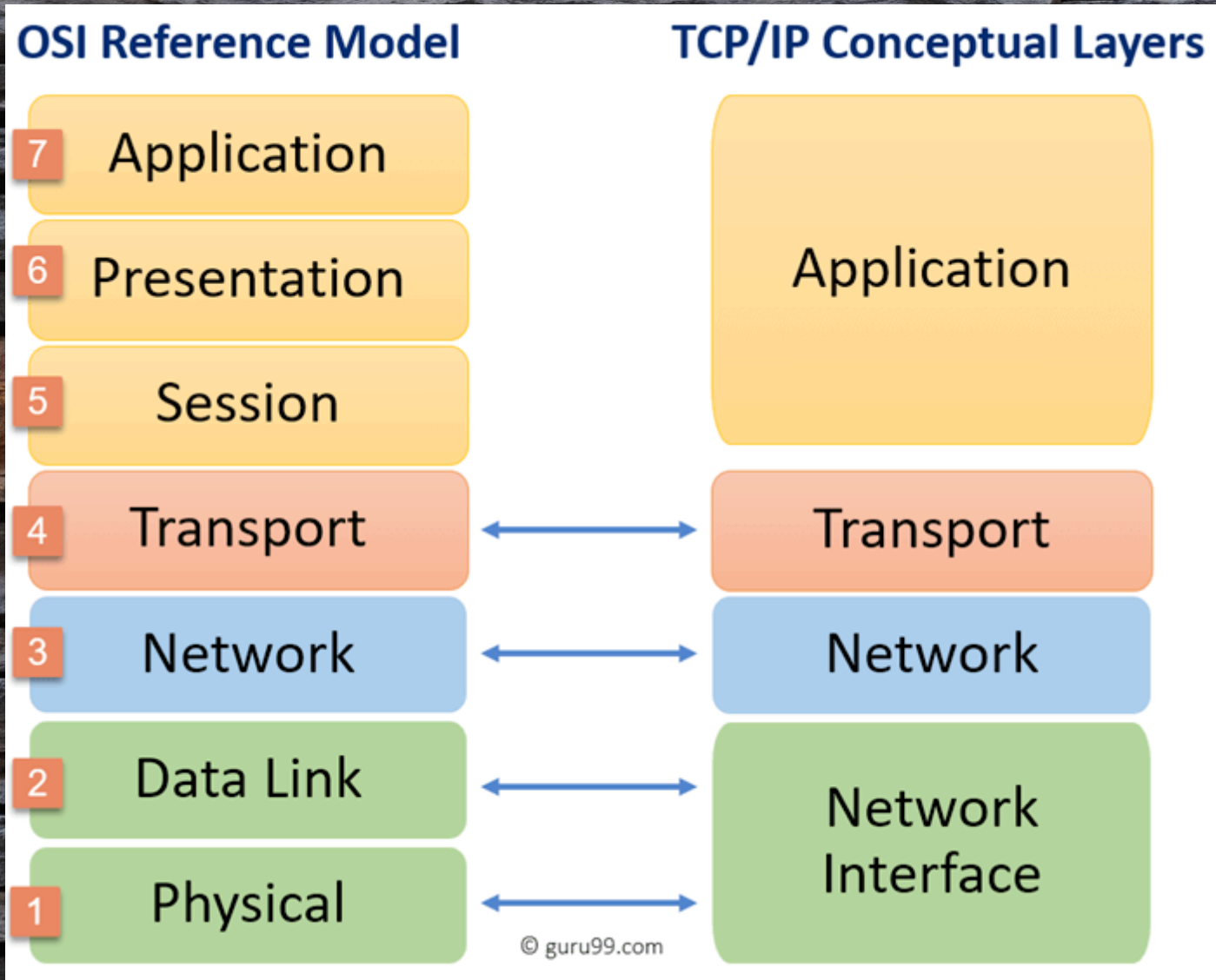
Fallacies of distributed computing

1. The network is reliable;
 2. Latency is zero;
 3. Bandwidth is infinite;
 4. The network is secure;
 5. Topology doesn't change;
 6. There is one administrator;
 7. Transport cost is zero;
 8. The network is homogeneous.
- 

A photograph of two skateboarders on a desert road. One skateboarder, wearing a pink shirt and dark pants, is riding away from the camera. The other skateboarder, wearing a yellow shirt and blue jeans, is falling off their board, with one arm raised and legs in the air. The road is paved and has yellow dashed lines. The background shows a desert landscape with hills and sparse vegetation. The text "What could possibly go wrong?" is overlaid in white on the image.

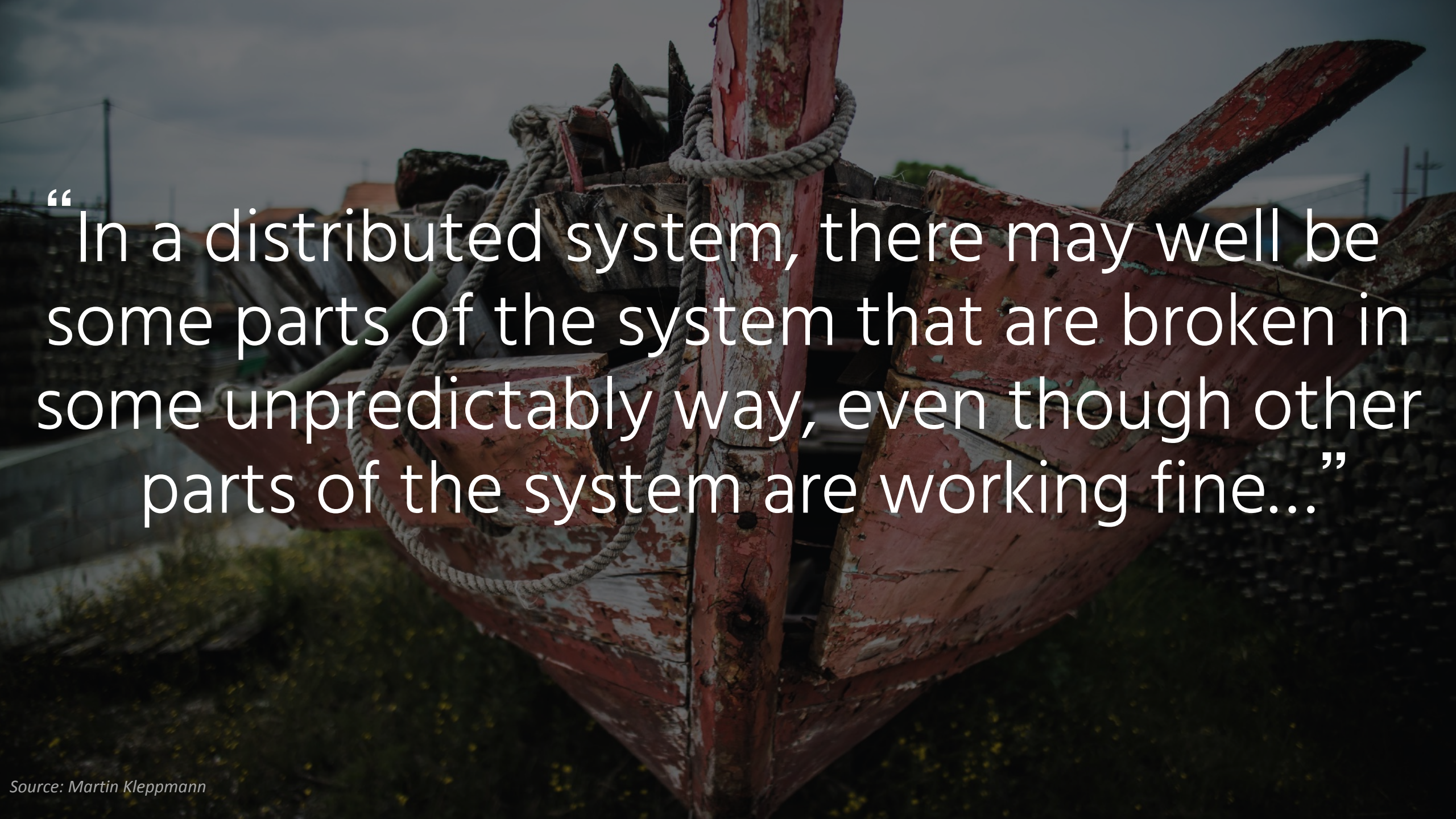
“What could possibly go wrong?”

OSI & TCP/IP



What happens when you type google.com...

.. in your browser's address bar and press Enter

A photograph of a weathered wooden boat, possibly a fishing boat, with red paint that is peeling and chipped away. The boat is made of dark wood and has several thick, light-colored ropes tied around it. The background is a dark, overcast sky and a blurred shoreline with some buildings and utility poles. The overall mood is somber and suggests a state of decay or failure.

“In a distributed system, there may well be some parts of the system that are broken in some unpredictable way, even though other parts of the system are working fine...”

“

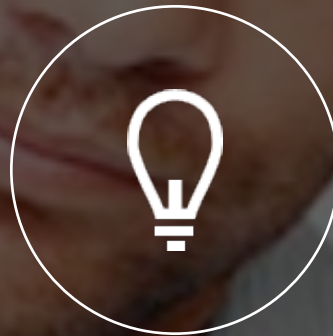
... but in a system with thousands of nodes, it is reasonable to assume that **something** is always broken. ”

- Martin Kleppmann

**ASK MORE
QUESTIONS**









Are we done yet?





Where do I start?

A structured approach to debugging distributed systems

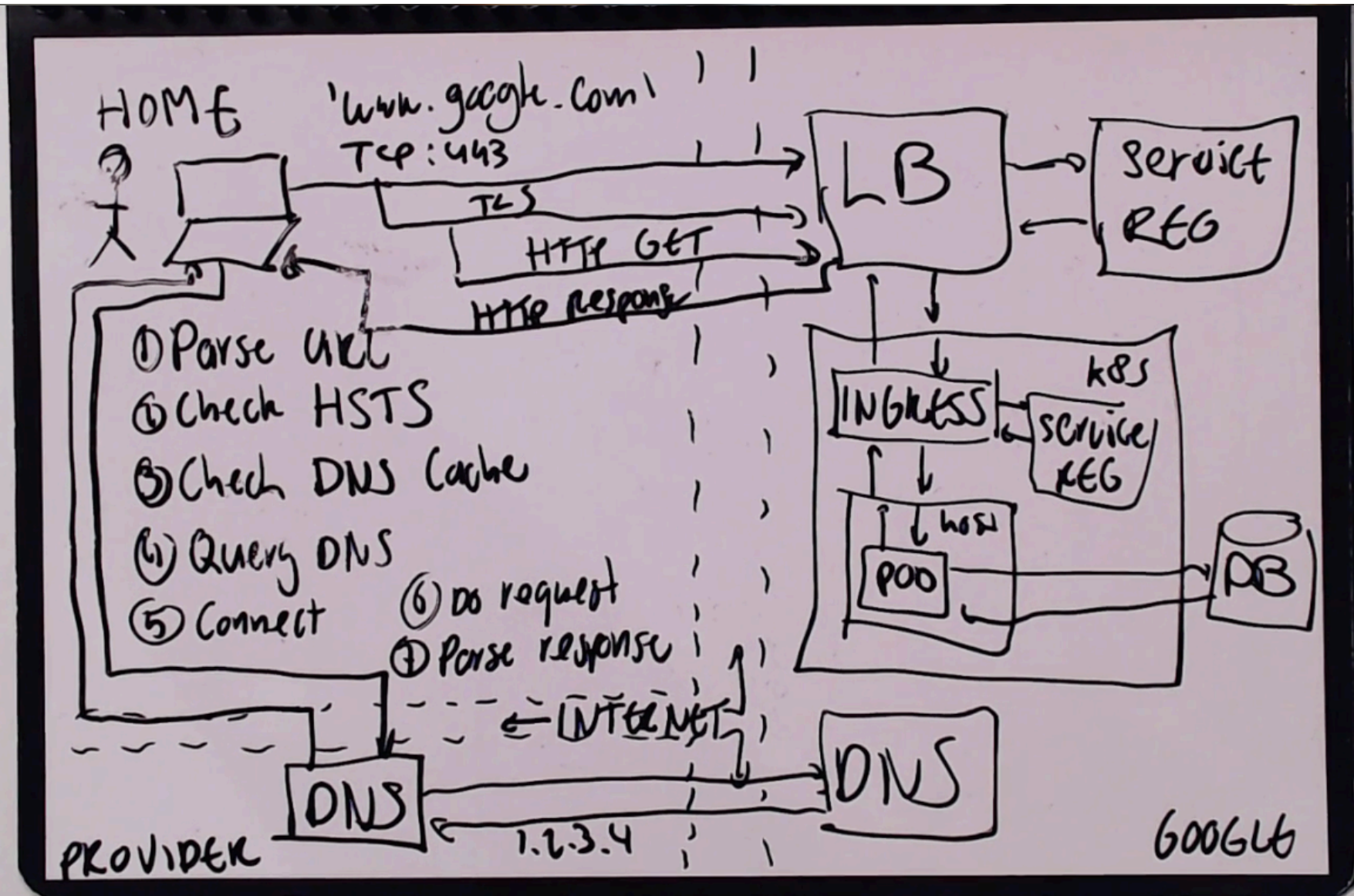
-  Observe & document
-  Create minimal reproducer
-  Debug client side
-  Check DNS & routing
-  Check connection
-  Inspect traffic / messages
-  Debug server side
-  Wrap up & post mortem



Step 1: Observe & document

- What do you know about the problem?
- Inspect logging, errors, metrics, tracing
- Draw the path from source to target - what's in between? Focus on details!
- Document what you know
- Can we reproduce in a test?
 - By injecting errors, for example

Step 1: Observe & document

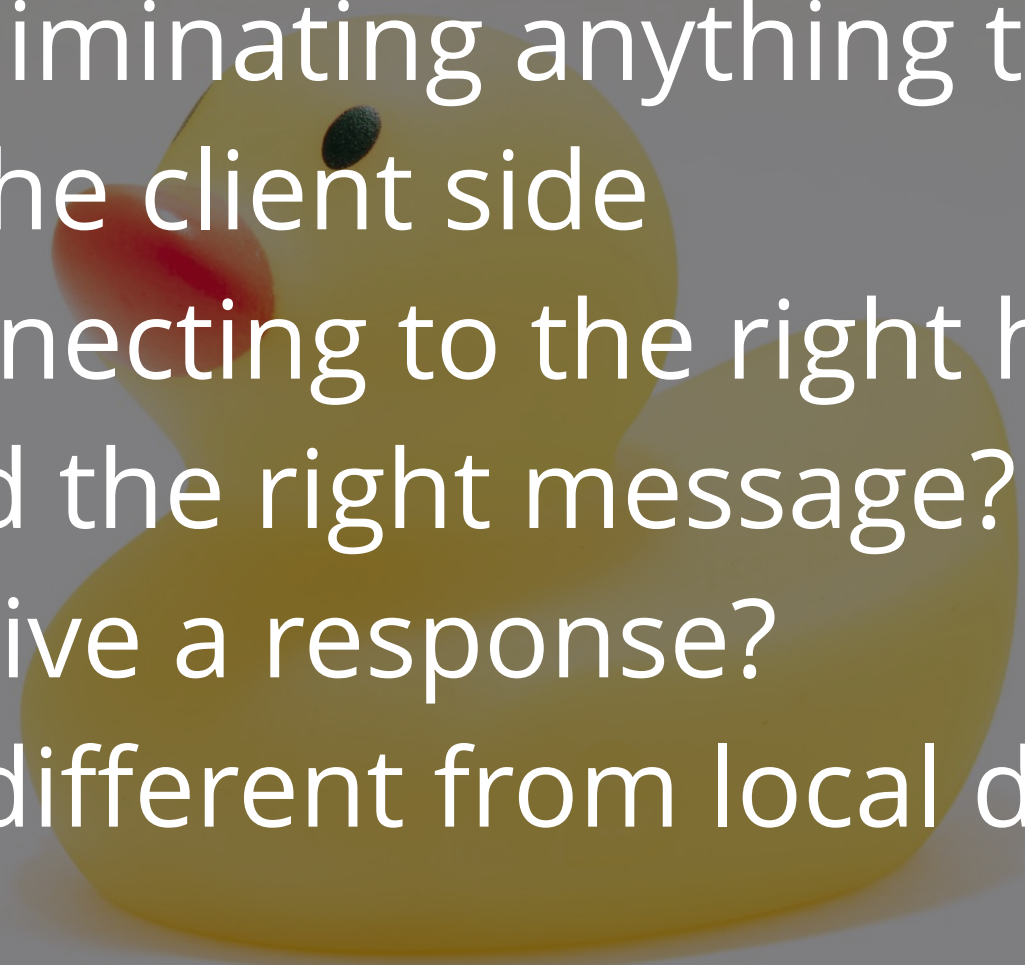


Step 2: Create minimal reproducer

- Goal: maximise the amount of debugging cycles
- Focus on short development iterations / feedback loops
- Get close to the action!

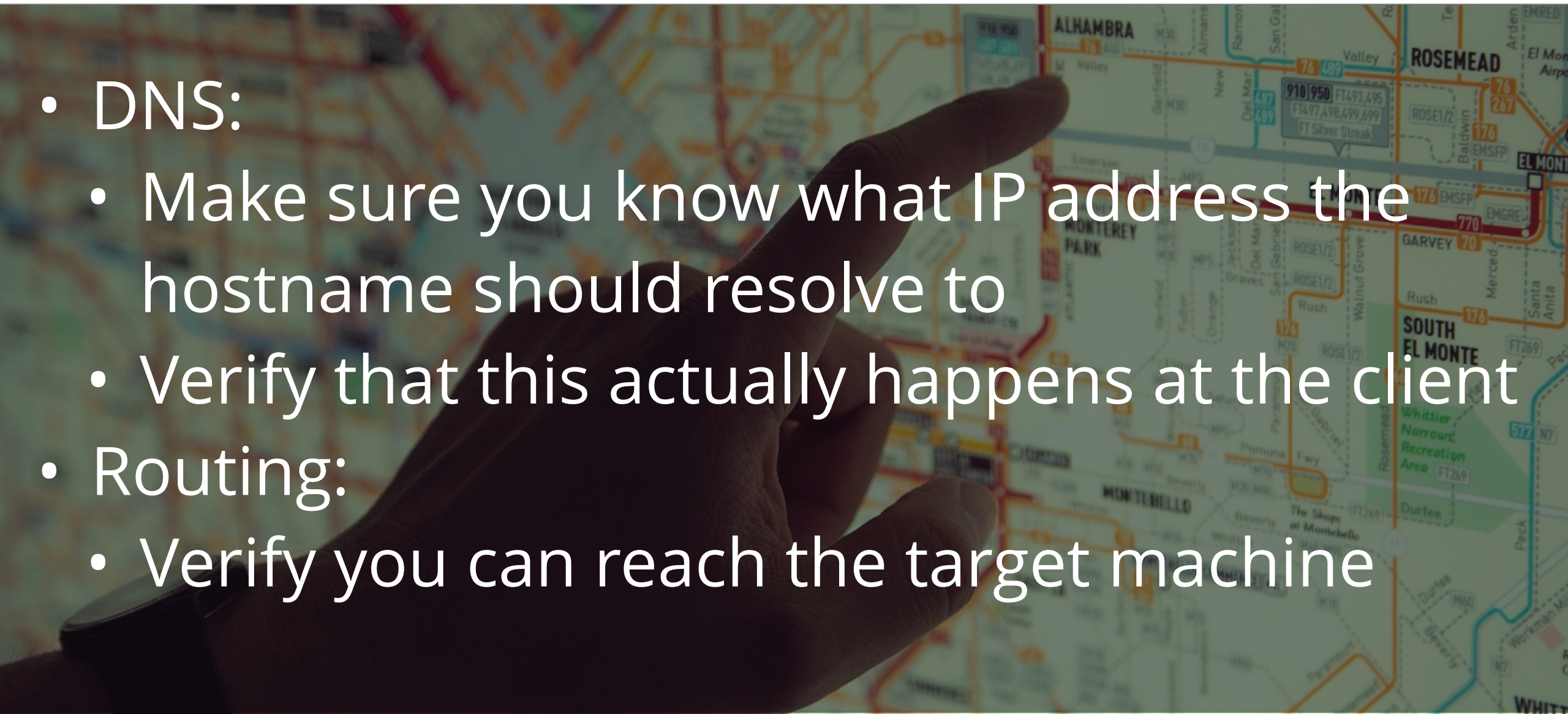


Step 3: Debug client side

- Focus on eliminating anything that could be wrong on the client side
 - Are we connecting to the right host?
 - Do we send the right message?
 - Do we receive a response?
 - Not much different from local debugging
- 

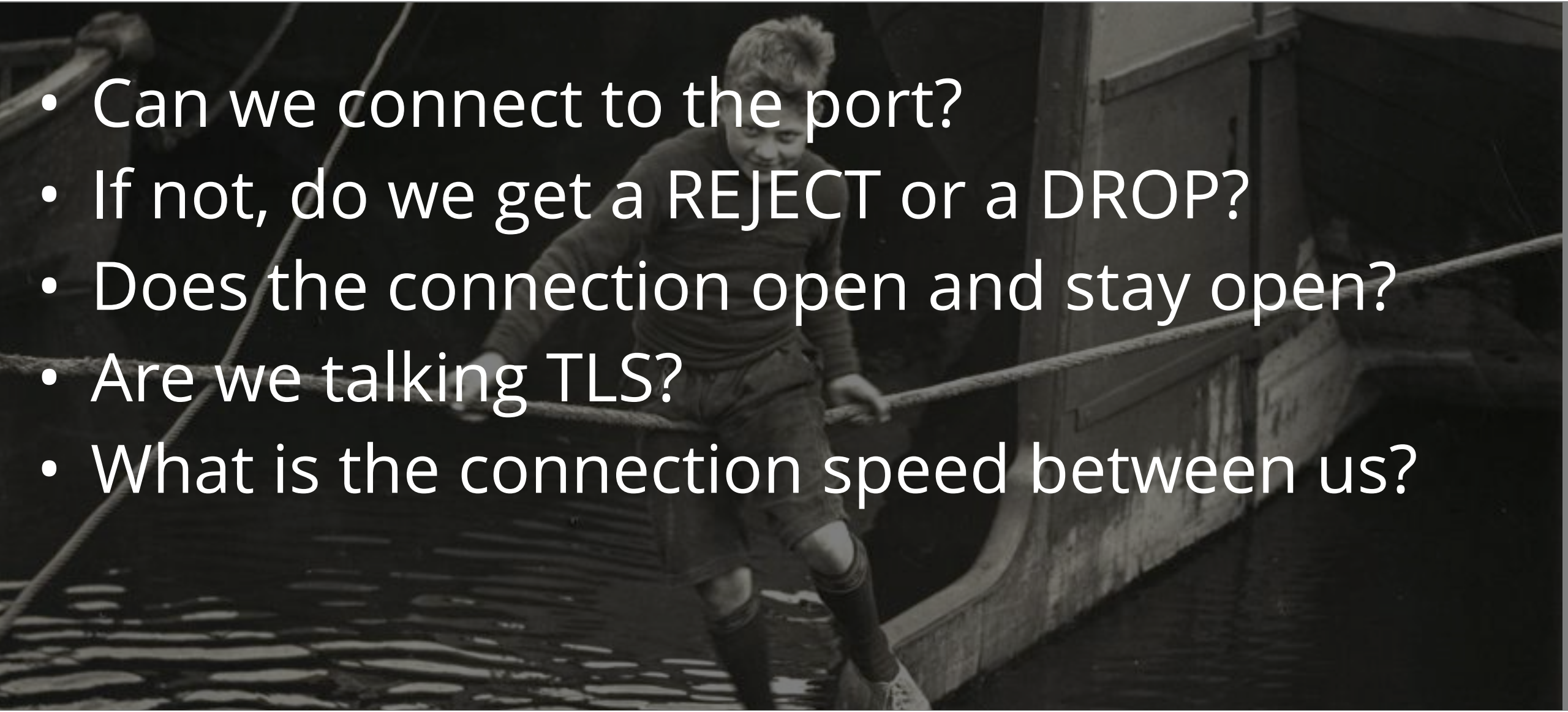
Step 4: Check DNS & routing

- DNS:
 - Make sure you know what IP address the hostname should resolve to
 - Verify that this actually happens at the client
- Routing:
 - Verify you can reach the target machine



Step 5: Check connection

- Can we connect to the port?
- If not, do we get a REJECT or a DROP?
- Does the connection open and stay open?
- Are we talking TLS?
- What is the connection speed between us?

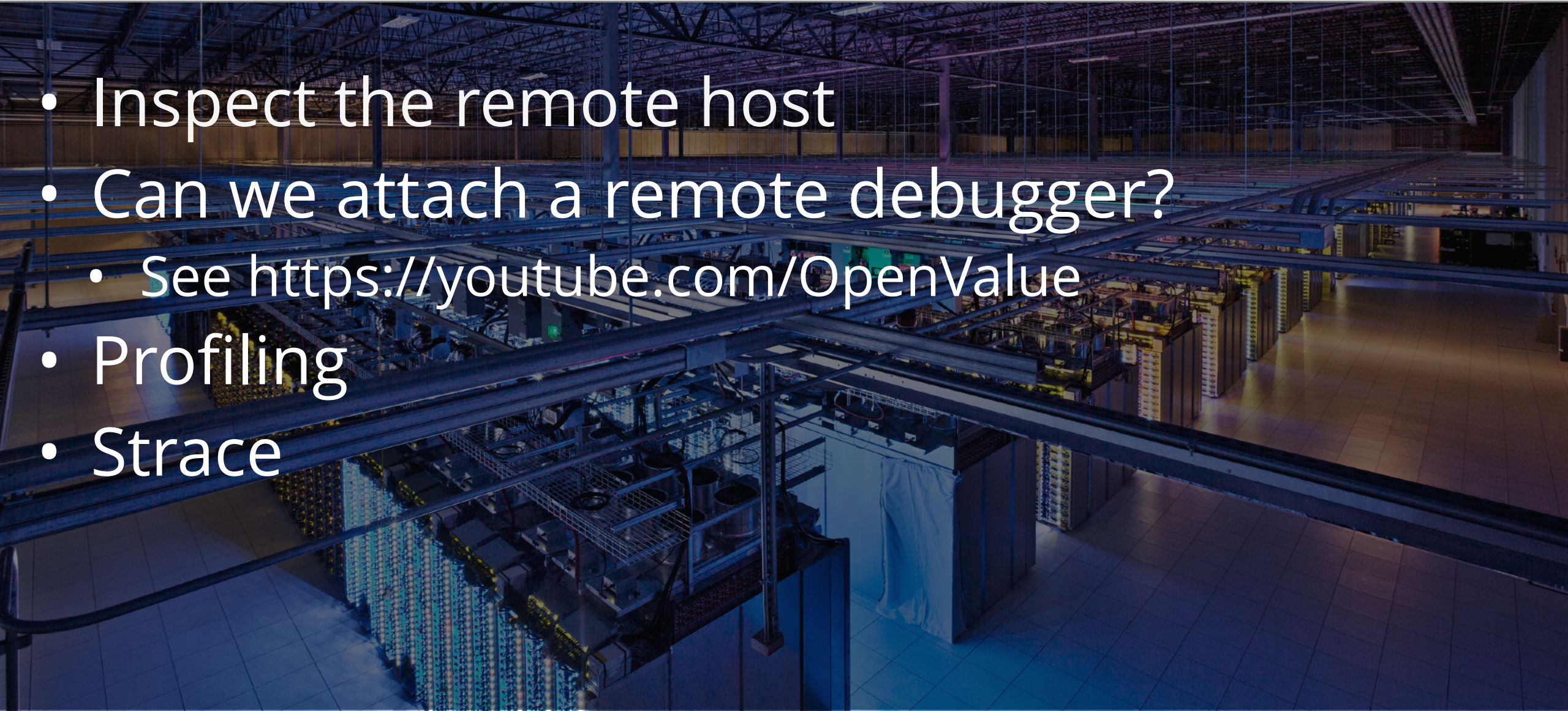


Step 6: Inspect traffic / messages


- Do we send the right request?
- Do we receive the right response?
- How do we know?
- How do we handle TLS?
- Are there any load balancers or proxies in between?

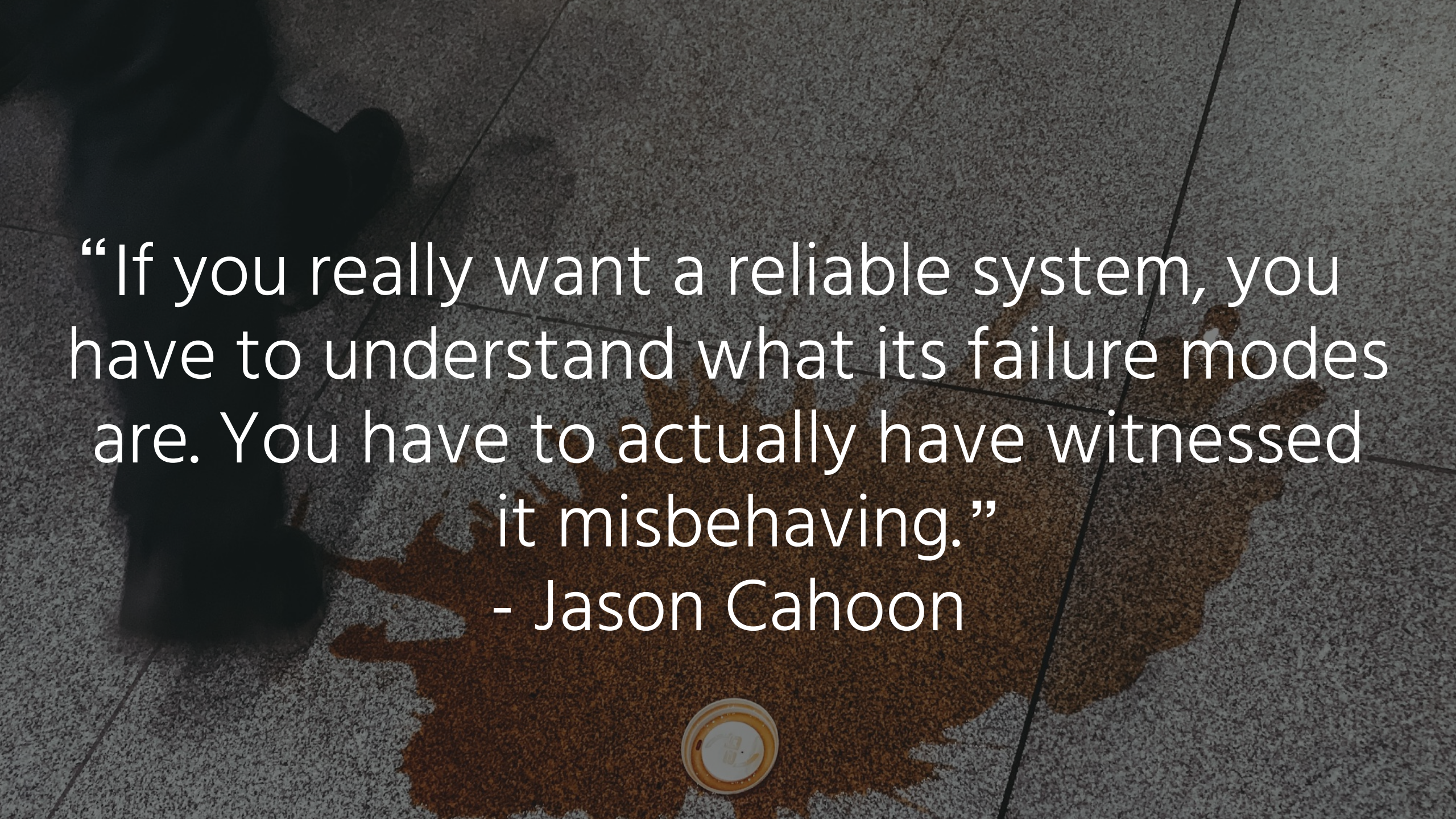
Step 7: Debug server side

- Inspect the remote host
- Can we attach a remote debugger?
 - See <https://youtube.com/OpenValue>
- Profiling
- Strace



Step 8: Wrap up & post mortem

- Document the issue:
 - Timeline
 - What did we see?
 - Why did it happen?
 - What was the impact?
 - How did we find out?
 - What did we do to mitigate and fix?
 - What should we do to prevent repetition?
- 
- A photograph of two young children sitting at a desk with a laptop. The child on the left is a boy with short brown hair, wearing a dark blue shirt, looking intently at the laptop screen with his mouth open. The child on the right is a girl with blonde braids, wearing a green and white striped shirt, pointing at the laptop screen with a wide, joyful smile. The background is slightly blurred, showing what appears to be an office or classroom setting.

A photograph of a tiled floor with a large, dark brown spill, likely a beverage. A white plastic bottle cap lies on the floor near the center of the spill. The background is dark and out of focus.

“If you really want a reliable system, you have to understand what its failure modes are. You have to actually have witnessed it misbehaving.”

- Jason Cahoon



Distributed systems war stories

A long, dimly lit subway tunnel. The left side is dark with a series of circular lights on the ceiling. The right side is illuminated with a warm, orange glow from a wall of small tiles. People are walking away from the camera towards a bright opening at the end of the tunnel. In the foreground on the right, a person is walking away with a large suitcase.

The time where it worked half of the time...

The one where two services didn't speak the same language...



Katharine
@katharineCodes

Hours. Hours debugging 2 microservices.
Why won't they talk to each other?
Why???

Because one was doing PUT, and server was expecting POST.



```
50:31.427 WARN 37761 --- [nio-9010-exec-4] .w.s.m.s.Default
Resolved [org.springframework.web.HttpRequestMethodNotSupportedException:
Method 'POST' not supported]
```


12:15 PM · Jul 13, 2021 · Twitter Web App



The one with expensive logging...

*"If Someone in Your Family
Has Cancer"*
Definition
Feelings
Treatment


Who?
What?
When?
Where?
Why?
How?



A classroom display featuring a list of WH questions (Who?, What?, When?, Where?, Why?, How?) on the left, a map of Europe in the center, and various small photographs and illustrations on the right.

The one at a school...





0 DAYS
SINCE IT
WAS DNS

(It's always DNS)

A scenic view of a Dutch tulip field with several windmills in the background under a blue sky with light clouds. The foreground is filled with rows of tulips in shades of red, yellow, and purple. In the background, several traditional Dutch windmills are visible, with the largest one on the left. The sky is a clear blue with some light, wispy clouds.









The one where only one country was
affected...



News

The one where breaking news broke
something else...

Summary: a structured approach to debugging distributed systems

-  Observe & document
-  Create minimal reproducer
-  Debug client side
-  Check DNS & routing
-  Check connection
-  Inspect traffic / messages
-  Debug server side
-  Wrap up & post mortem





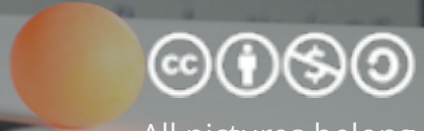
**THAT'S IT.
NOW GO KICK SOME ASS!**



Questions?

Thanks for your time.

Got feedback? Tweet it!



All pictures belong to their respective authors

