

# Solana Alpenglow Consensus

Quentin Kniep  
Kobi Sliwinski  
Roger Wattenhofer



Kobi



Quentin

# A New Protocol



## THE WALL STREET JOURNAL.

### Computer Science

RANK	COLLEGE	COUNTRY
1	ETH Zurich*	Switzerland
2	California Institute of Technology	United States
3	University of Oxford	United Kingdom
4	Massachusetts Institute of Technology	United States
5	Georgia Institute of Technology	United States

Eidgenössische Technische Hochschule

ETH Zurich

The blockchain world thrives on innovation, and Anza is the first to be forming a new research team. Founded by Professor Roger Wattenhofer and two of his recent PhD students, Kobi Sliwinski and Quentin Kniep, from ETH Zurich (Eidgenössische Technische Hochschule Zürich) in Switzerland, the team will work on the fundamental aspects of the protocol to bring Solana to the next level. This includes designing a more performant and provably correct turbine-based consensus algorithm, as well as researching improvements to latency, resilience, and economics.



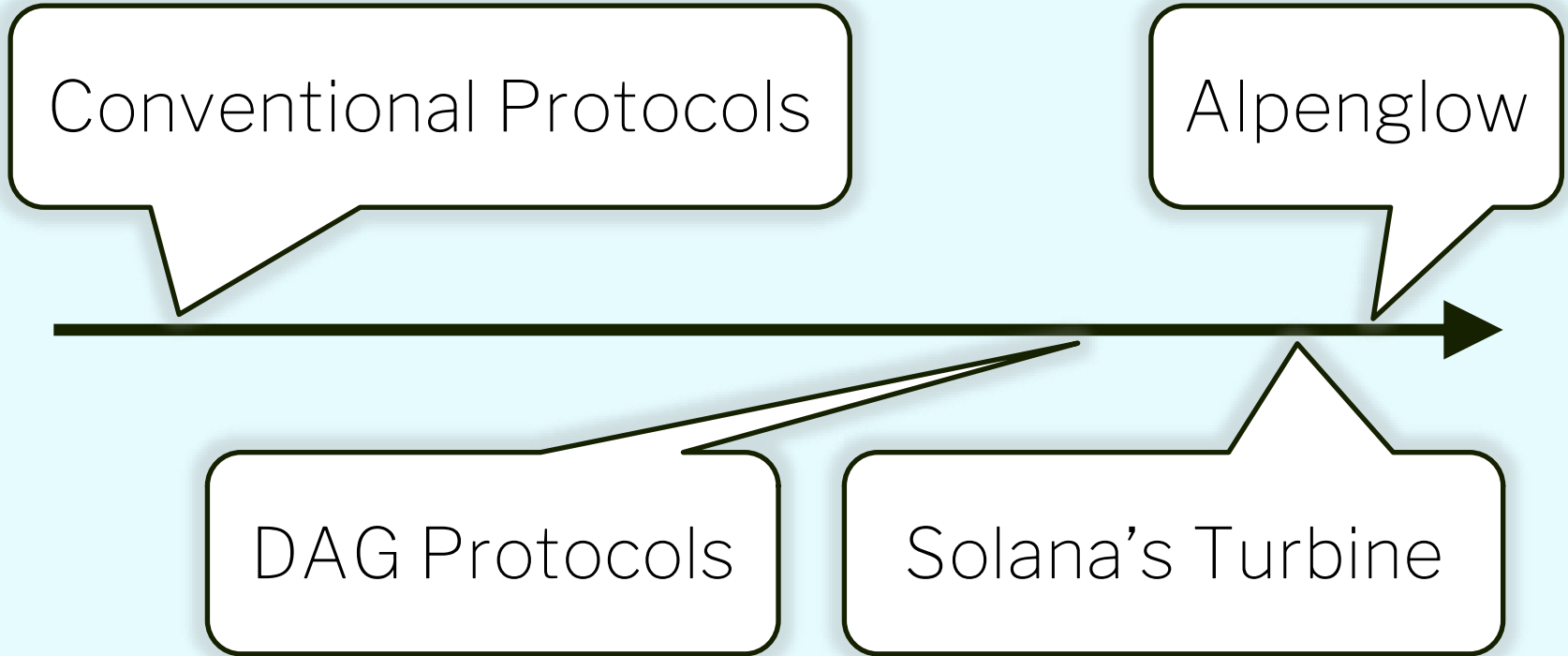
Alpenglow

Zurich, Switzerland

INCREASE BANDWIDTH

REDUCE LATENCY

# BANDWIDTH



INCREASE BANDWIDTH  
*optimized*

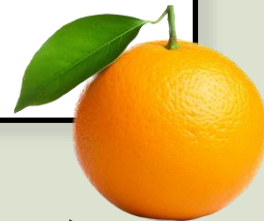
REDUCE LATENCY

# LATENCY



Alpenglow Finality  
150ms (100-380ms)

Optimistic Confirmation  
500-600ms



Best Known  
> 400ms

Solana Finality  
12.8s



INCREASE BANDWIDTH  
*optimized*



REDUCE LATENCY  
*so much better!*



20 + 20

Security!



20%  + 20% 

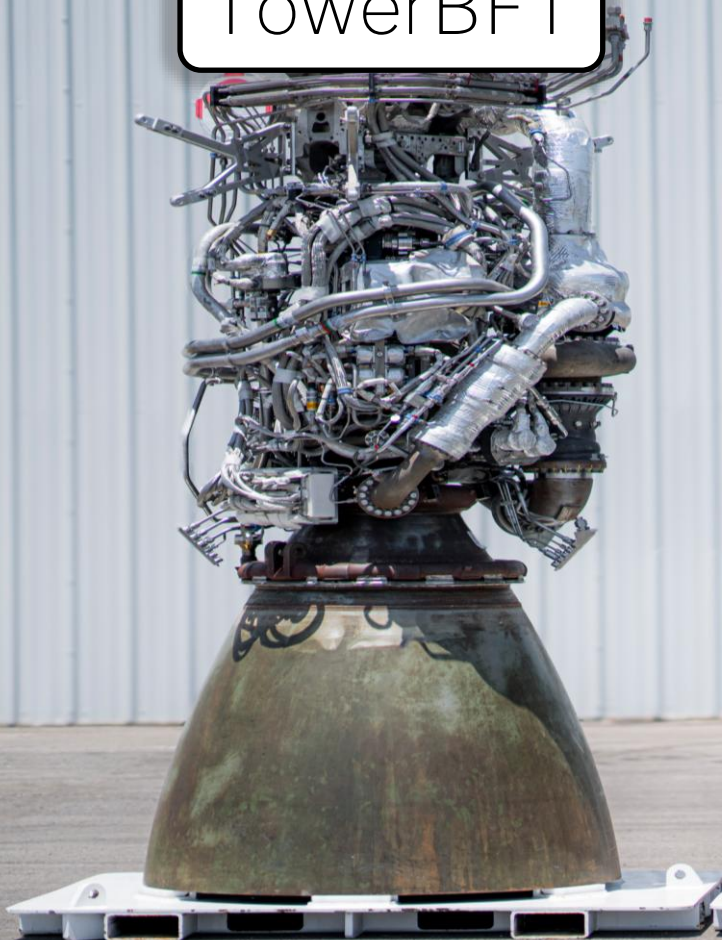
Security!

**KEEP IT**

**SIMPLE**

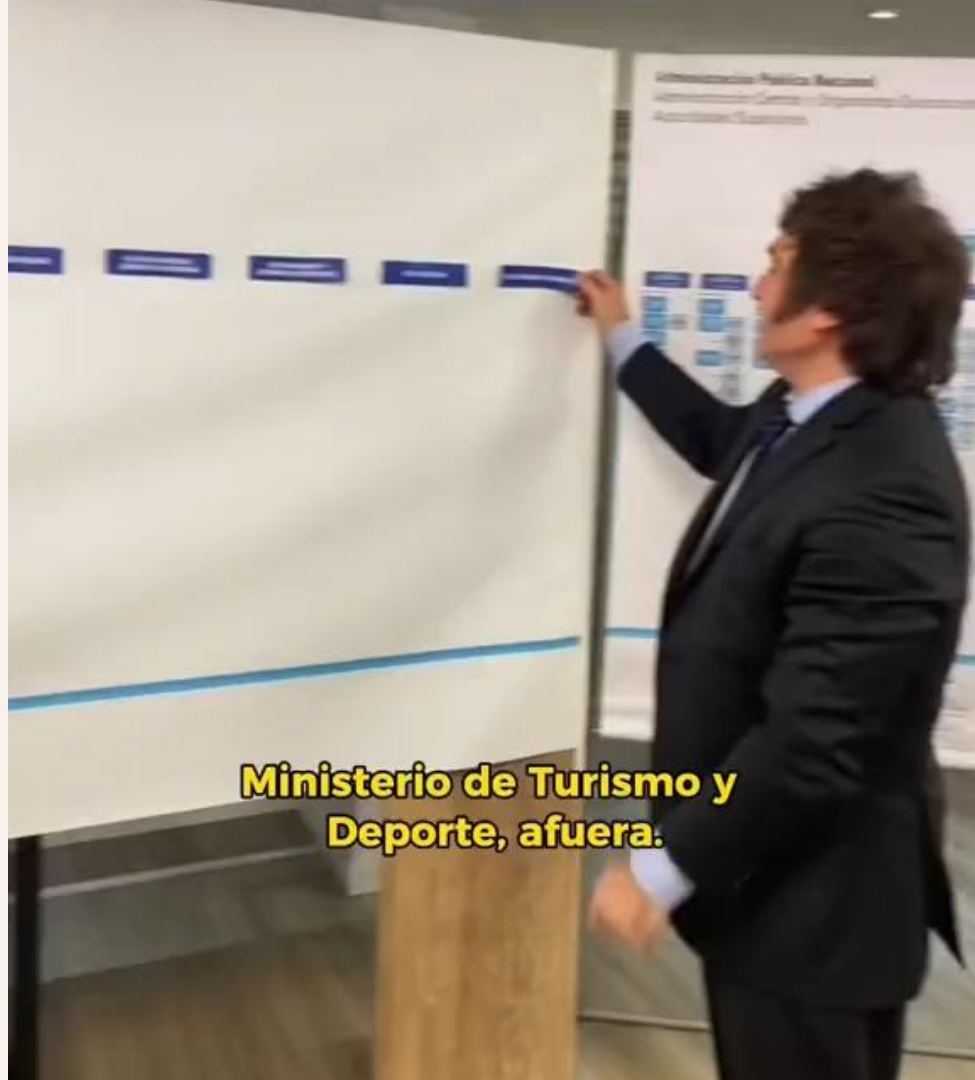


TowerBFT



Alpenglow





**Ministerio de Turismo y  
Deporte, afuera.**

Proof of History


Gossip

Turbine

Tower

# Rotor

## Sing a song of Simplex\*

Victor Shoup<sup>1</sup>   
Offchain Labs  
victor@shoup.net

February 5, 2025

**Abstract.** We flesh out some details of the recently proposed Simplex atomic broadcast protocol and modify it so that leaders disperse blocks in a more communication-efficient fashion. The resulting protocol, called *DispersedSimplex*, maintains the simplicity and excellent — indeed, optimal — latency characteristics of the original Simplex protocol. We also present several variations, including a variant that supports “signature-free” blocks, and variants that incorporate very recently developed data dissemination techniques to disseminate blocks even more efficiently, and variants that are “signature-free” to account not just network latency but also network bandwidth limitations and dissemination costs. Based on these estimates, we argue that despite its simplicity, *DispersedSimplex* should, in principle, perform in practice as well as or better than any other state-of-the-art atomic broadcast protocol, at least in terms of common-case throughput and latency.



## 1 Introduction

## Simplex Consensus: A Simple and Fast Consensus Protocol

Benjamin Y Chan\*  
Cornell University  
byc@cs.cornell.edu

Rafael Pass†  
Tel-Aviv University and Cornell Tech  
rafaelp@tau.ac.il

June 1, 2023

### Abstract

We present a theoretical framework for analyzing the efficiency of consensus protocols, and apply it to analyze the efficiency of state-of-the-art algorithms.

## BANYAN: Fast Rotating Leader BFT

Yann Vonlanthen  
yvonlanthen@ethz.ch  
ETH Zurich  
Switzerland

Massimo Albarello  
massimo@onfabric.io  
ETH Zurich  
Switzerland

Jakub Sliwinski  
jsliwinski@ethz.ch  
ETH Zurich  
Switzerland

Roger Wattenhofer  
wattenhofer@ethz.ch  
ETH Zurich  
Switzerland

assuring that resource accesses are consistent across all participants (called replicas), thus guaranteeing deterministic and secure execution of the computation.

Byzantine agreement (BA) protocols typically provide consensus on a single decision, while state machine replication protocols (SMR) focus on making efficient use of this costly primitive. In practice, the most widely used consensus protocols (such as Bitcoin [51], Ethereum [16], and Algorand [35]) make use of an elected leader to propose a batch of transactions, called a block. A total order across leaders is then obtained by chaining blocks.

It presents BANYAN, the first rotating leader state machine replication (SMR) protocol that allows transactions to be confirmed in just a single round-trip time in the Byzantine-tolerance (BFT) setting. Based on minimal alterations to the Internet Computer Consensus (ICC) protocol with negligible communication overhead, we introduce a novel quorum mechanism that enables optimal block confirmation at each step. Crucially, the modes of operation at each step are deterministic.

Crucially, leaders can misbehave and cause forks. Properly, leaders can misbehave and cause forks.

Votor

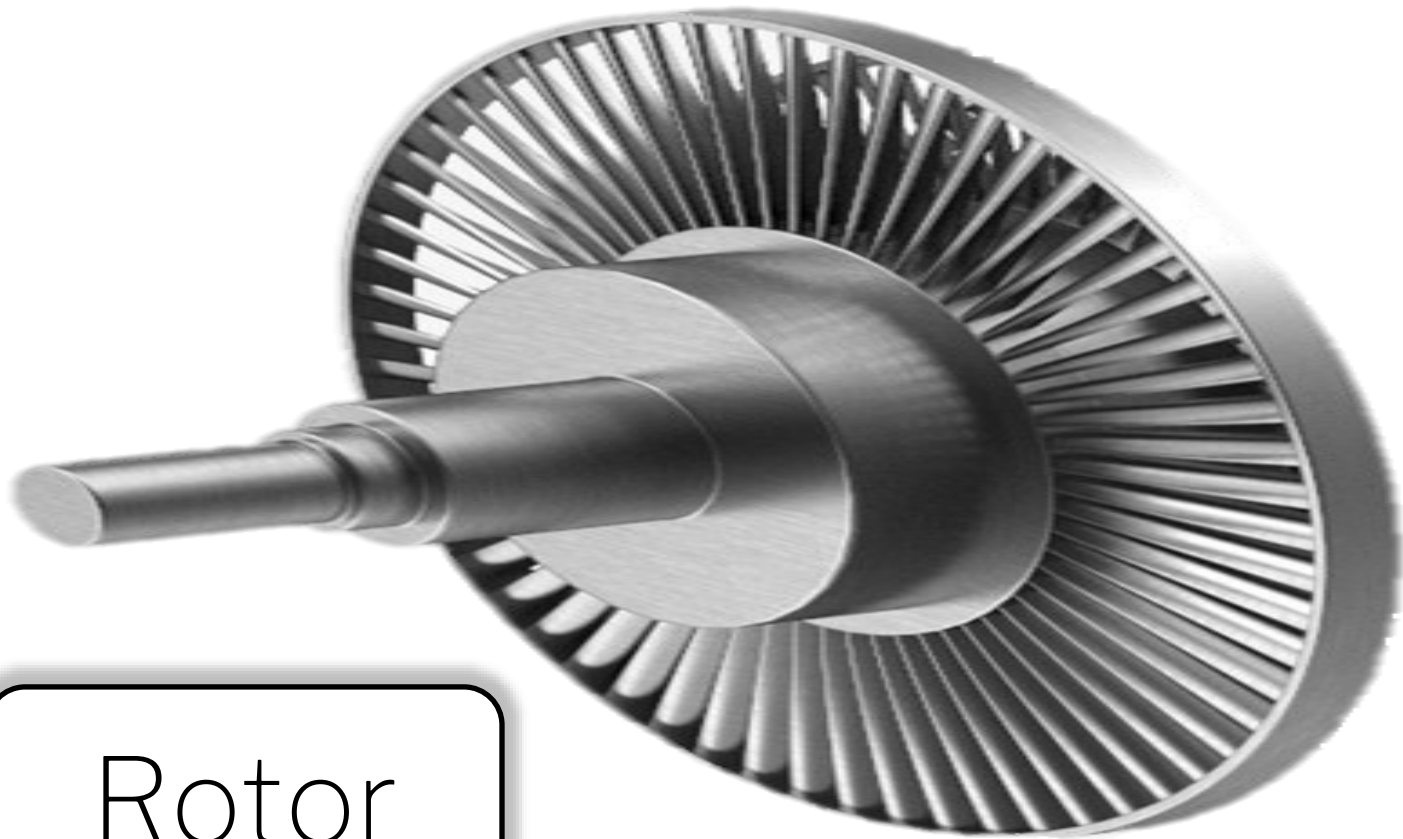
Rotor

**ROTOR**



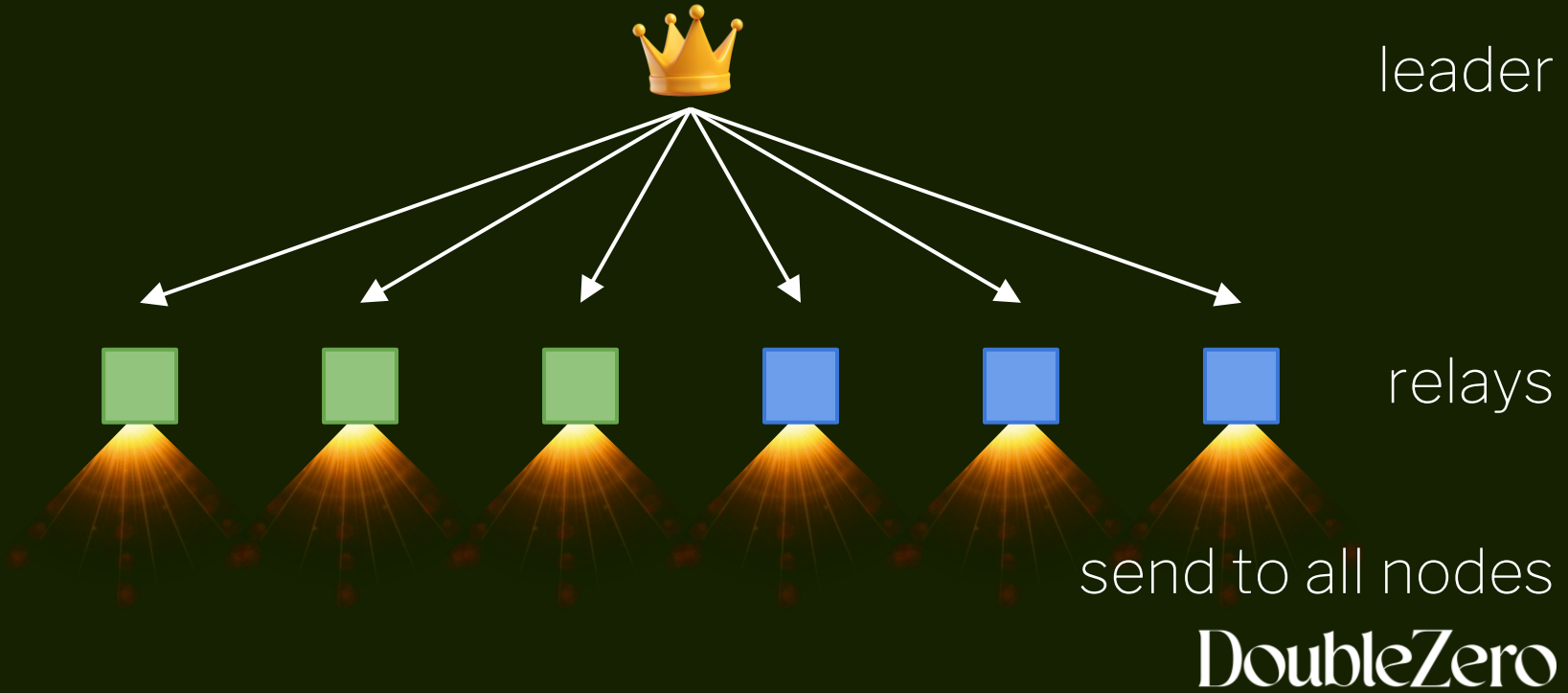


Turbine

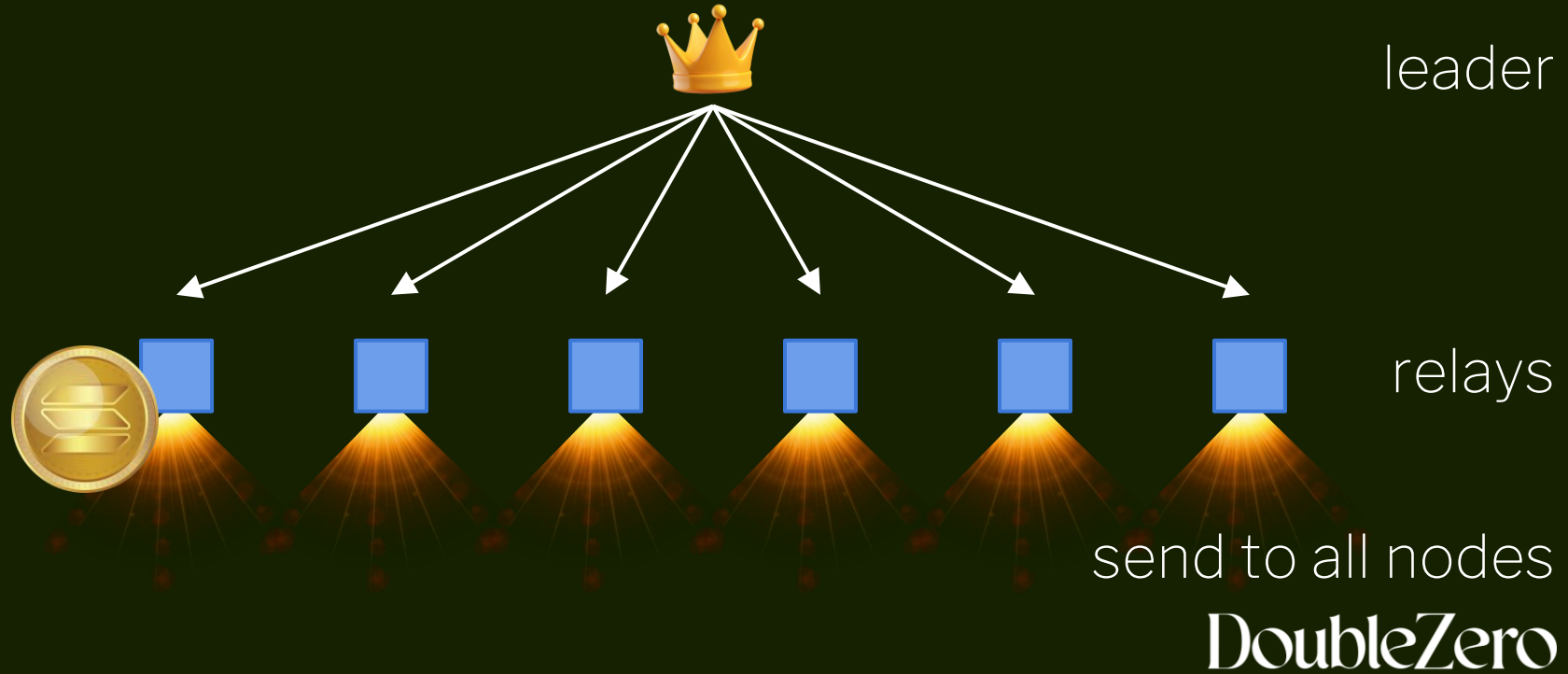


Rotor

# ROTOR: SINGLE LAYER



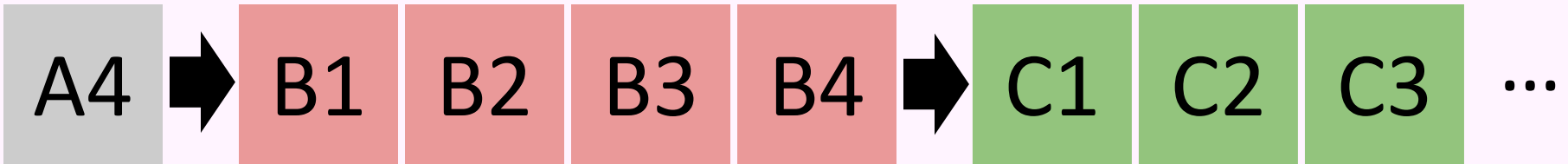
# ROTOR: SINGLE LAYER



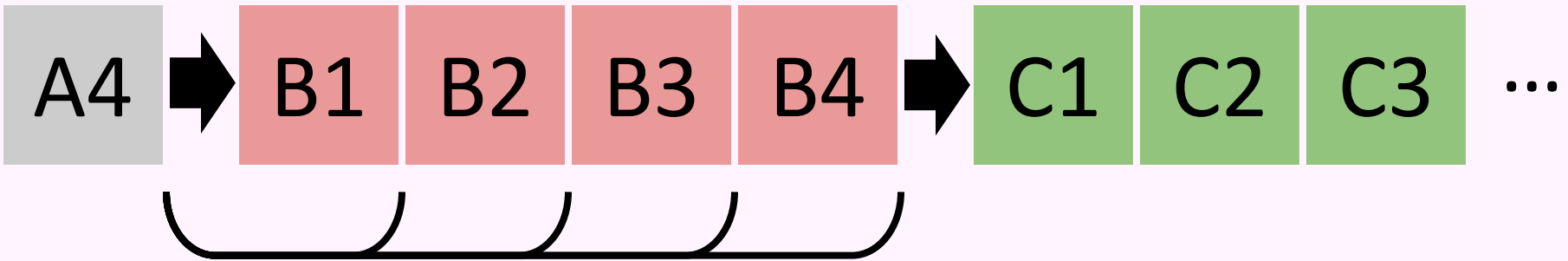
 **VOTOR**



# WHAT WE WANT




# PROOF-OF-HISTORY → TIMEOUTS




# FINALIZATION



60% 

mixed

80% 



one more  
round of  
voting



BLS  
cert.

**Algorithm 1** Votor, event loop, single-threaded

```

1: upon Block( $s$ , hash, hashparent) do
2:   if TRYNOTAR(Block( $s$ , hash, hashparent)) then
3:     CHECKPENDINGBLOCKS()
4:   else if Voted  $\notin$  state[ $s$ ] then
5:     pendingBlocks[ $s$ ]  $\leftarrow$  Block( $s$ , hash, hashparent)

6: upon Timeout( $s$ ) do
7:   if Voted  $\notin$  state[ $s$ ] then
8:     TRYSKIPWINDOW( $s$ )

9: upon BlockNotarized( $s$ , hash( $b$ )) do
10:  state[ $s$ ]  $\leftarrow$  state[ $s$ ]  $\cup$  {BlockNotarized(hash( $b$ ))}
11:  TRYFINAL( $s$ , hash( $b$ ))

12: upon ParentReady( $s$ , hash( $b$ )) do
13:  state[ $s$ ]  $\leftarrow$  state[ $s$ ]  $\cup$  {ParentReady(hash( $b$ ))}
14:  CHECKPENDINGBLOCKS()
15:  SETTIMEOUTS( $s$ )

16: upon SafeToNotar( $s$ , hash( $b$ )) do
17:  TRYSKIPWINDOW( $s$ )
18:  if ItsOver  $\notin$  state[ $s$ ] then
19:    broadcast NotarFallbackVote( $s$ , hash( $b$ ))  $\triangleright$  notar-fallback vote
20:    state[ $s$ ]  $\leftarrow$  state[ $s$ ]  $\cup$  {BadWindow}

21: upon SafeToSkip( $s$ ) do
22:  TRYSKIPWINDOW( $s$ )
23:  if ItsOver  $\notin$  state[ $s$ ] then
24:    broadcast SkipFallbackVote( $s$ )  $\triangleright$  skip-fallback vote
25:    state[ $s$ ]  $\leftarrow$  state[ $s$ ]  $\cup$  {BadWindow}

```

**Algorithm 2** Votor, helper functions

```

1: function WINDOWSLOTS( $s$ )
2:  return array with slot numbers of the leader window with slot  $s$ 

3: function SETTIMEOUTS( $s$ )  $\triangleright$  set timeouts for the window with slot  $s$ 
4:  for  $i \in$  WINDOWSLOTS( $s$ ) do
5:    schedule event Timeout( $i$ ) at time clock() +  $\Delta_{\text{timeout}}$  +  $(i - s + 1) \cdot \Delta_{\text{block}}$ 

6:  $\triangleright$  Check if a notarization vote can be cast.  $\triangleleft$ 
7: function TRYNOTAR(Block( $s$ , hash, hashparent))
8:  if Voted  $\in$  state[ $s$ ] then
9:    return false
10:  firstSlot  $\leftarrow$  ( $s$  is the first slot in leader window)
11:  if (firstSlot and ParentReady(hashparent)  $\in$  state[ $s$ ])
12:    or (not firstSlot and VotedNotar(hashparent)  $\in$  state[ $s - 1$ ]) then
13:    broadcast NotarVote( $s$ , hash)  $\triangleright$  notarization vote
14:    state[ $s$ ]  $\leftarrow$  state[ $s$ ]  $\cup$  {Voted, VotedNotar(hash)}
15:    pendingBlocks[ $s$ ]  $\leftarrow$   $\perp$ 
16:    TRYFINAL( $s$ , hash)
17:    return true

18: function TRYFINAL( $s$ , hash( $b$ ))
19:  if BlockNotarized(hash( $b$ ))  $\in$  state[ $s$ ] and VotedNotar(hash( $b$ ))  $\in$  state[ $s$ ]
20:    and BadWindow  $\notin$  state[ $s$ ] then
21:    broadcast FinalVote( $s$ )  $\triangleright$  finalization vote
22:    state[ $s$ ]  $\leftarrow$  state[ $s$ ]  $\cup$  {ItsOver}

22: function TRYSKIPWINDOW( $s$ )
23:  for  $k \in$  WINDOWSLOTS( $s$ ) do
24:    if Voted  $\notin$  state[ $k$ ] then
25:      broadcast SkipVote( $k$ )  $\triangleright$  skip vote
26:      state[ $k$ ]  $\leftarrow$  state[ $k$ ]  $\cup$  {Voted, BadWindow}
27:      pendingBlocks[ $k$ ]  $\leftarrow$   $\perp$ 

28: function CHECKPENDINGBLOCKS()
29:  for  $s$  : pendingBlocks[ $s$ ]  $\neq$   $\perp$  do  $\triangleright$  iterate with increasing  $s$ 
30:    TRYNOTAR(pendingBlocks[ $s$ ])

```

# Solana Alpenglow Consensus

## Increased Bandwidth, Reduced Latency

Quentin Kniep   Jakub Sliwinski   Roger Wattenhofer

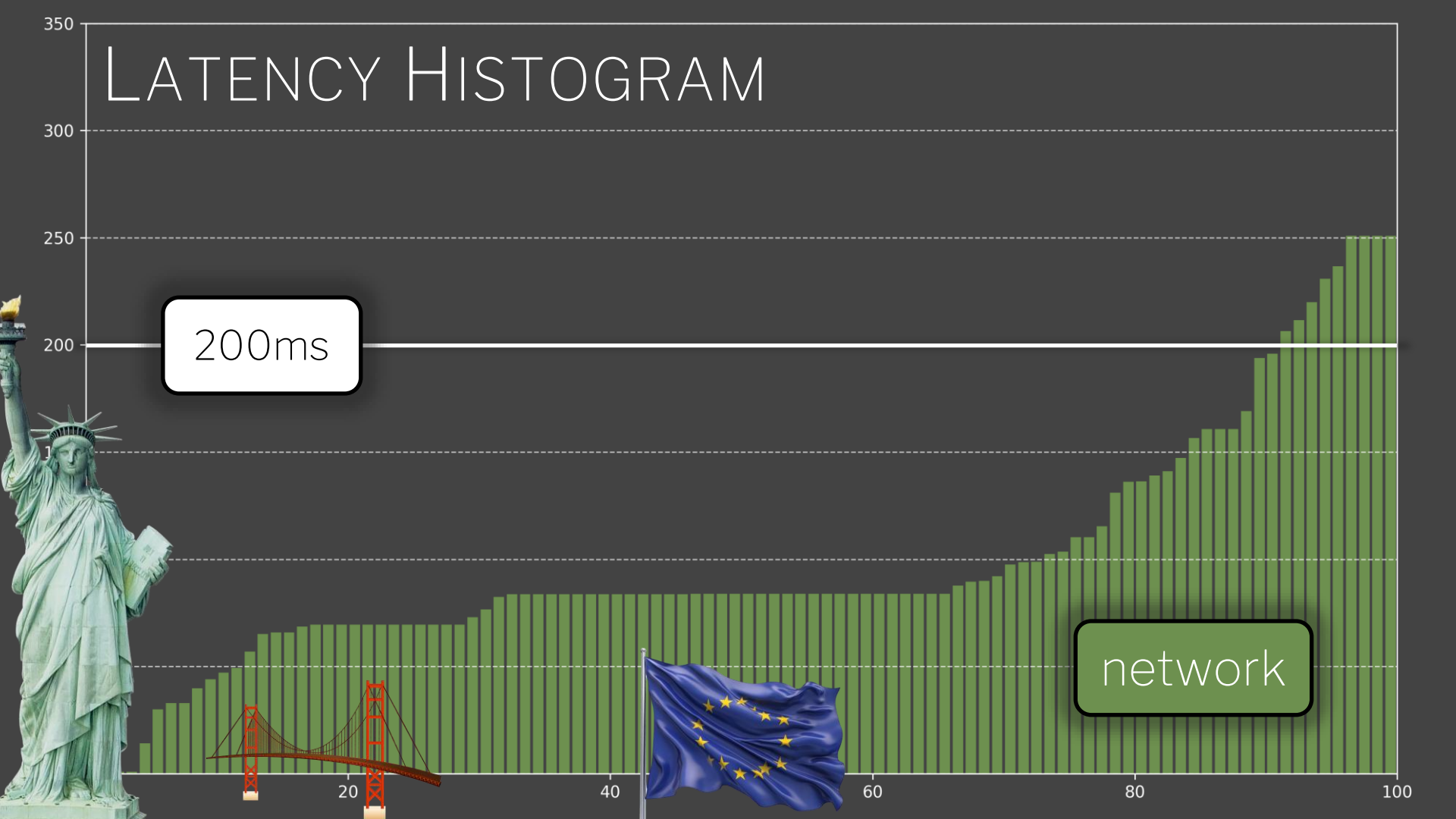


White Paper v1.0, May 19, 2025

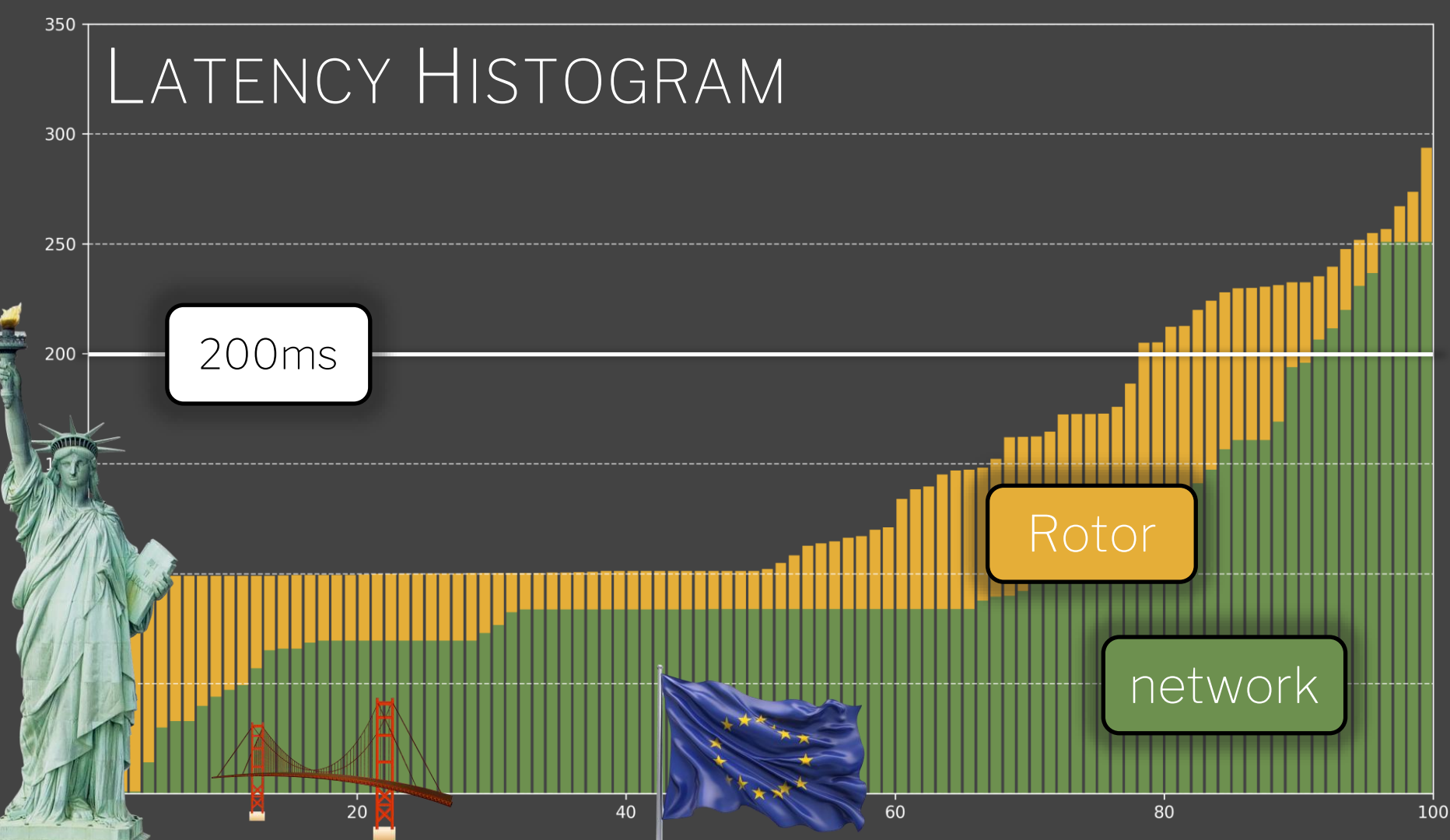
### **Abstract**

In this paper we describe and analyze Alpenglow, a consensus protocol tailored for a global high-performance proof-of-stake blockchain.

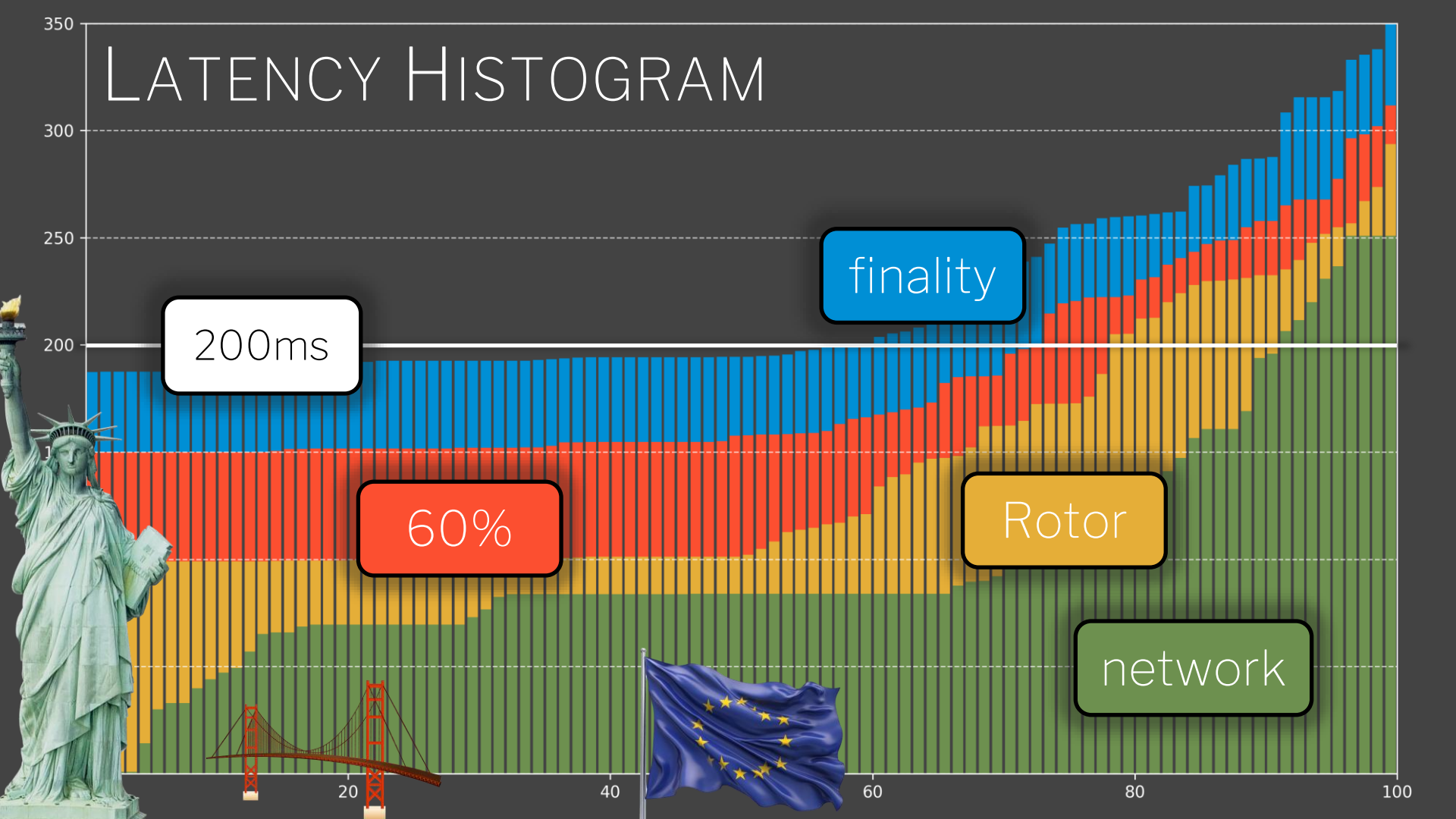
# LATENCY HISTOGRAM



# LATENCY HISTOGRAM



# LATENCY HISTOGRAM



200ms

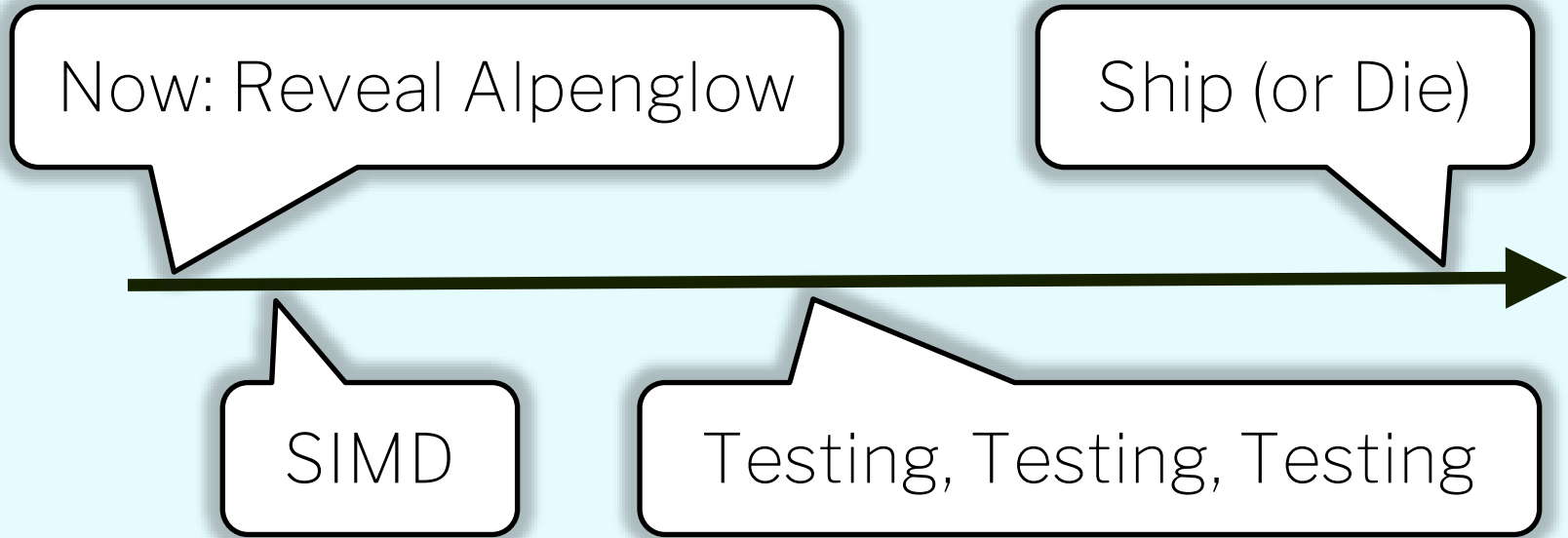
finality

60%

Rotor

network

# TIMELINE 2025





# Alpenglow

Quentin Kniep  
@qkniep

Kobi Sliwinski  
@DiscoKobi

Roger Wattenhofer  
@TheWattenhofer