Embedding a Deterministic BFT Protocol in a Block DAG

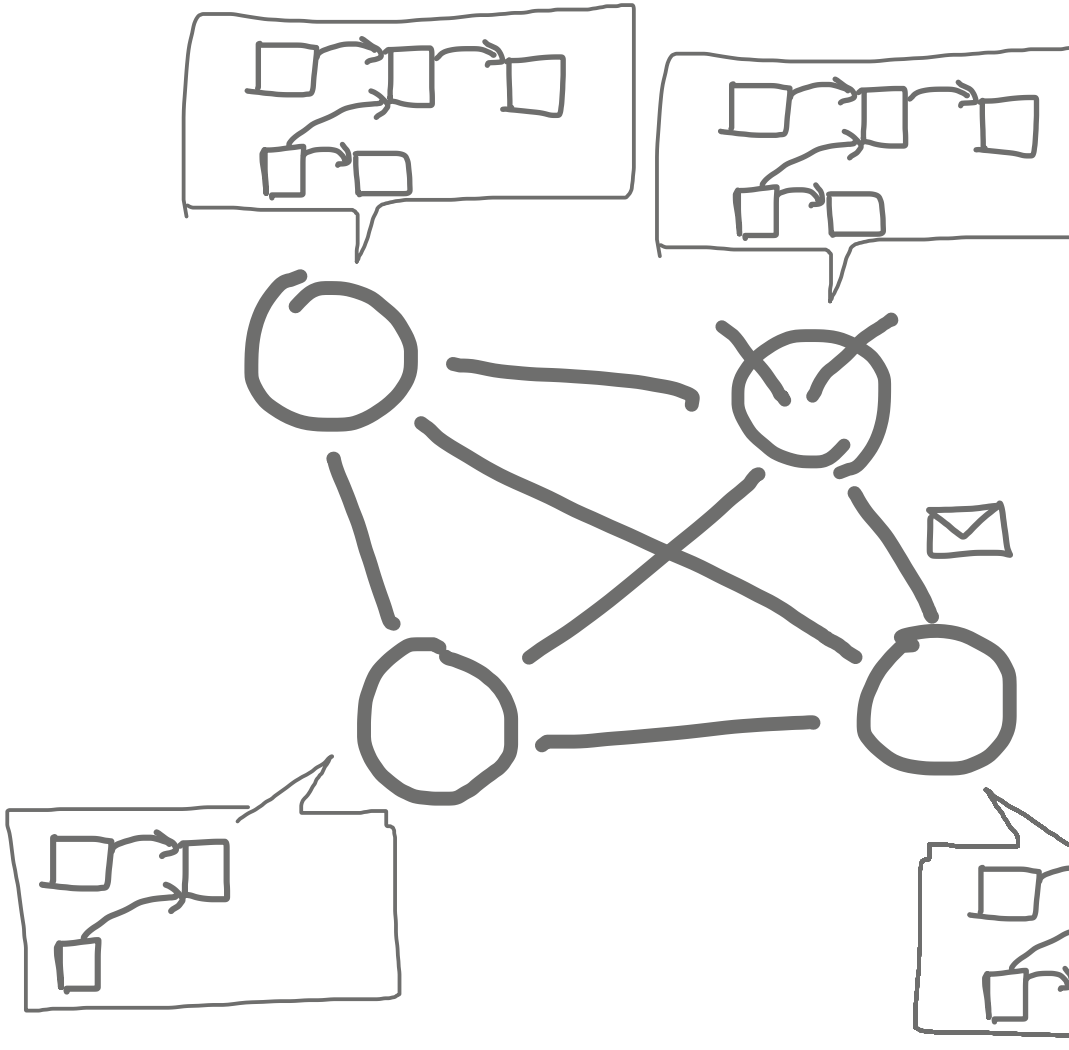
Maria A Schett
mail@maria-a-schett.net

George Danezis
g.danezis@ucl.ac.uk

UCL

PODC-2021-07

System Model



- byzantine < f for 3f+1
- eventually delivered
- "block DAG"

Background

Block DAG-"Blockchains":

Hashgraph; Aleph; Blockmania; Flare;

DAG Rider [PODC'21]; Narwhal & Tusk [arxiv];

Pedigree

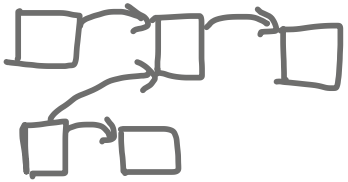
Peer review [SOSP'07]

Lamport's happened-before

Core Ideas

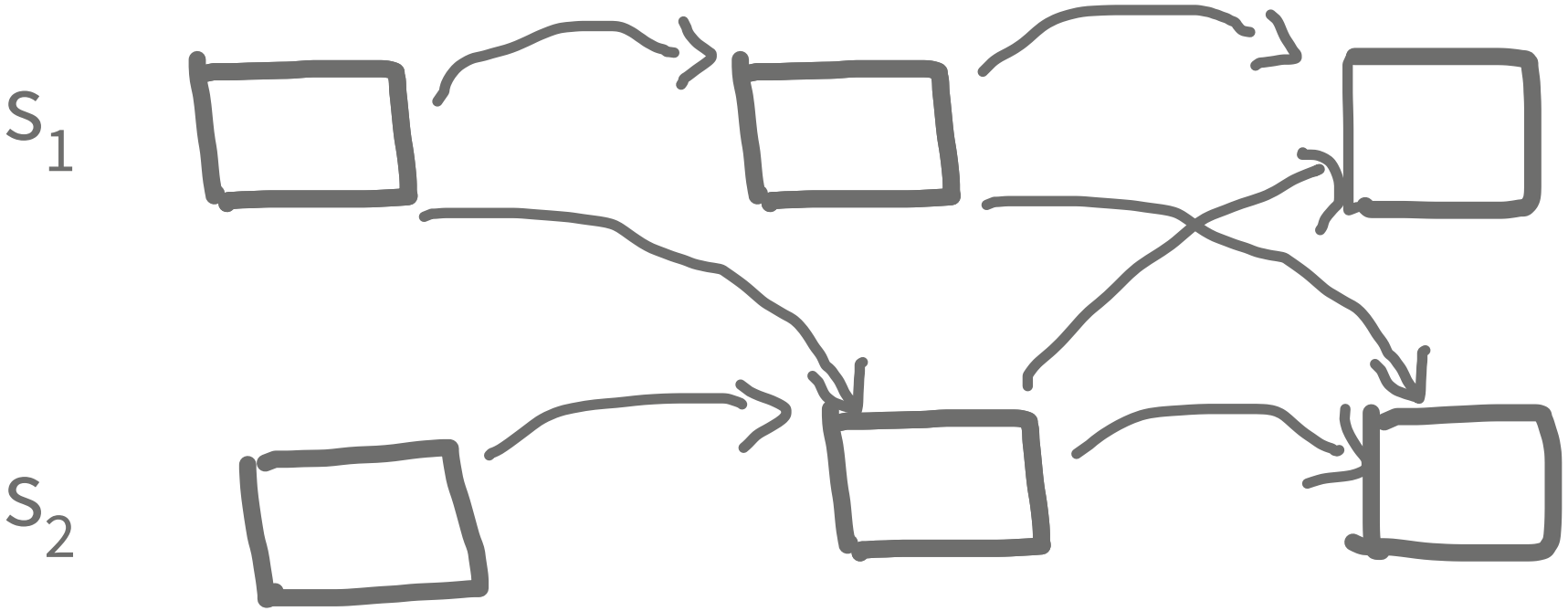
Every correct server ...

1 ... builds a **joint block DAG.**

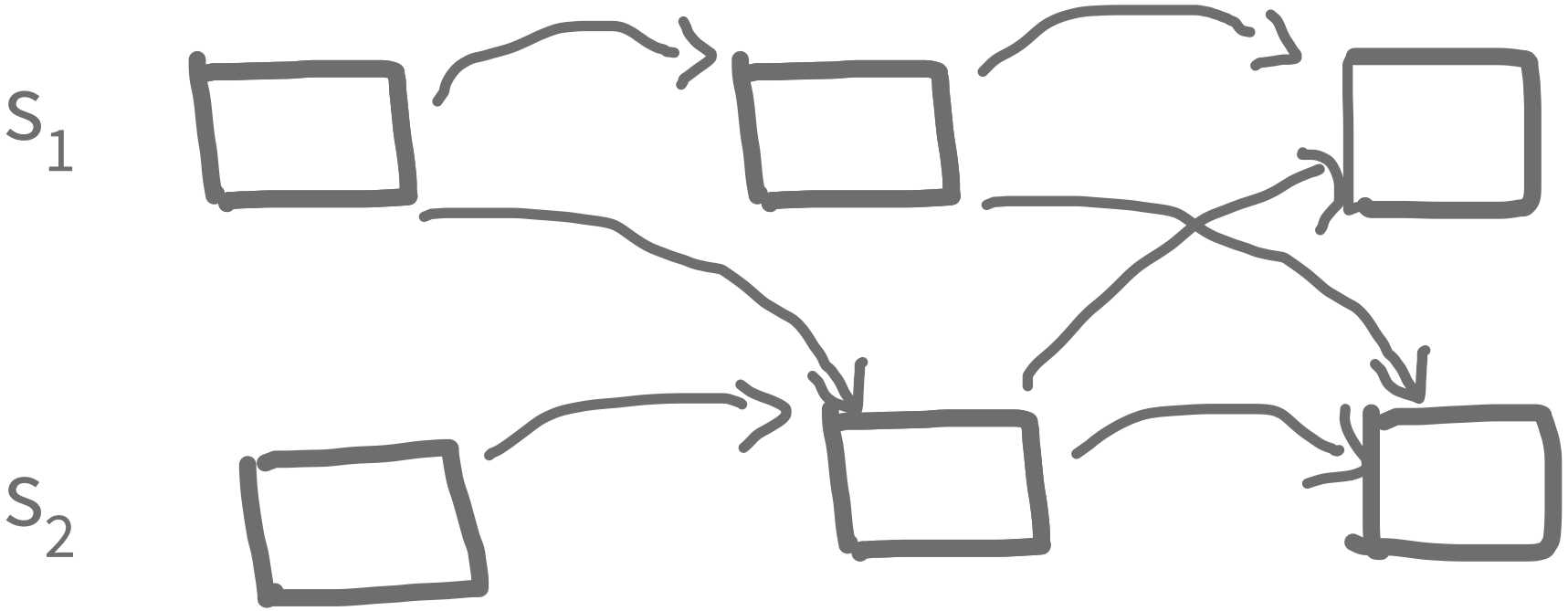


2 ... **interprets** determinstic BFT $\boxed{P}$ locally.

1 Build A Joint Block DAG

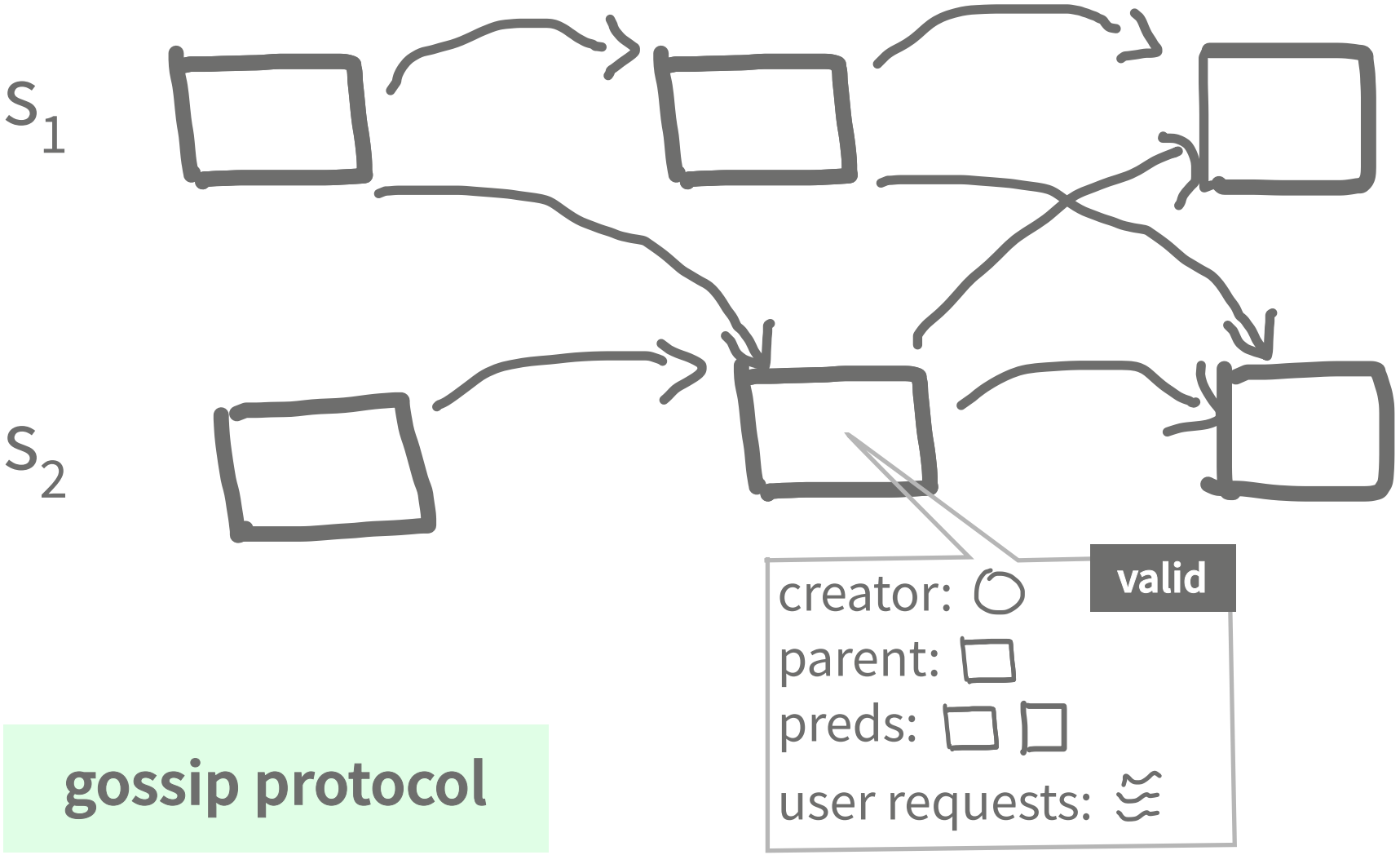


1 Build A Joint Block DAG



gossip protocol

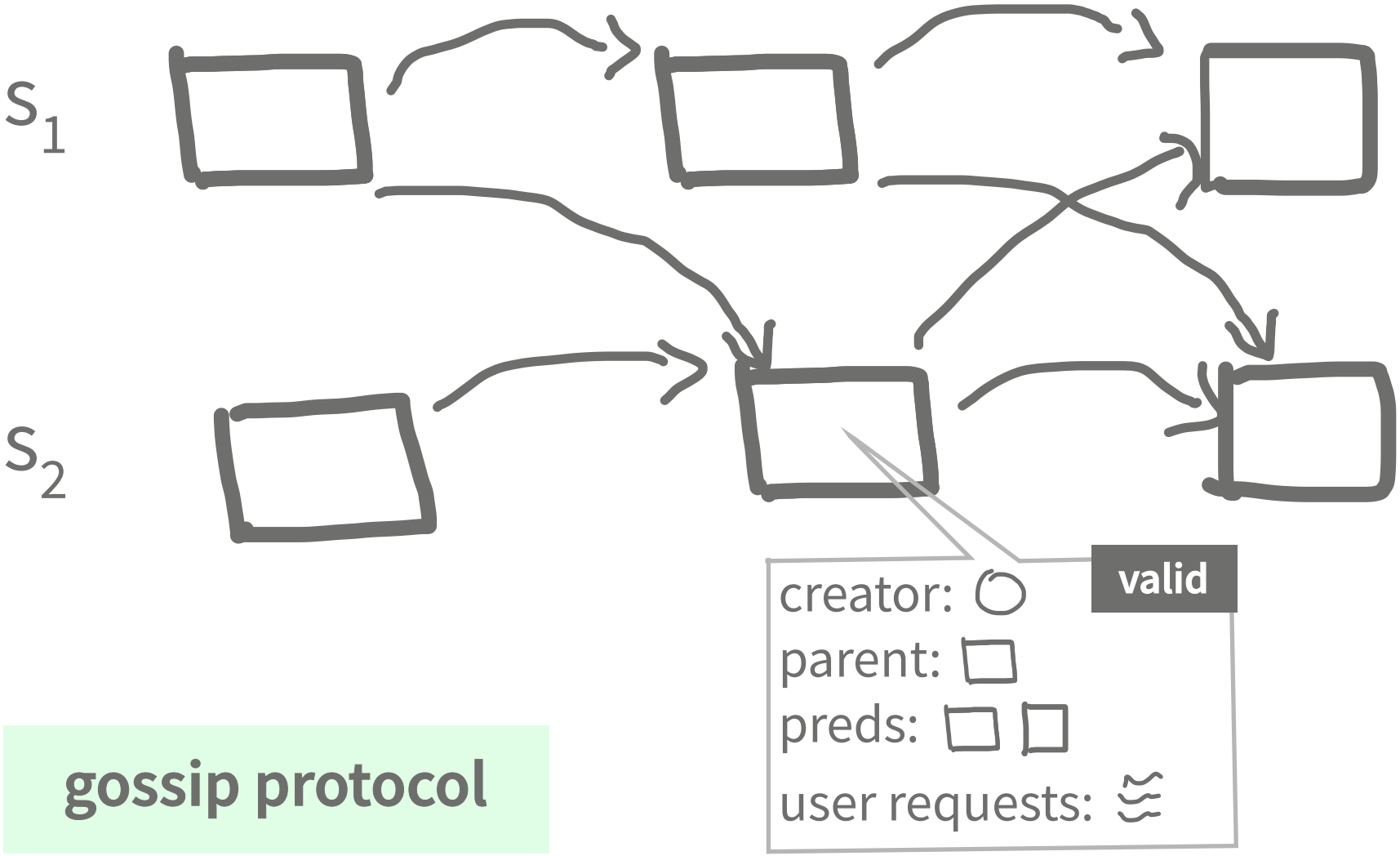
1 Build A Joint Block DAG



gossip protocol

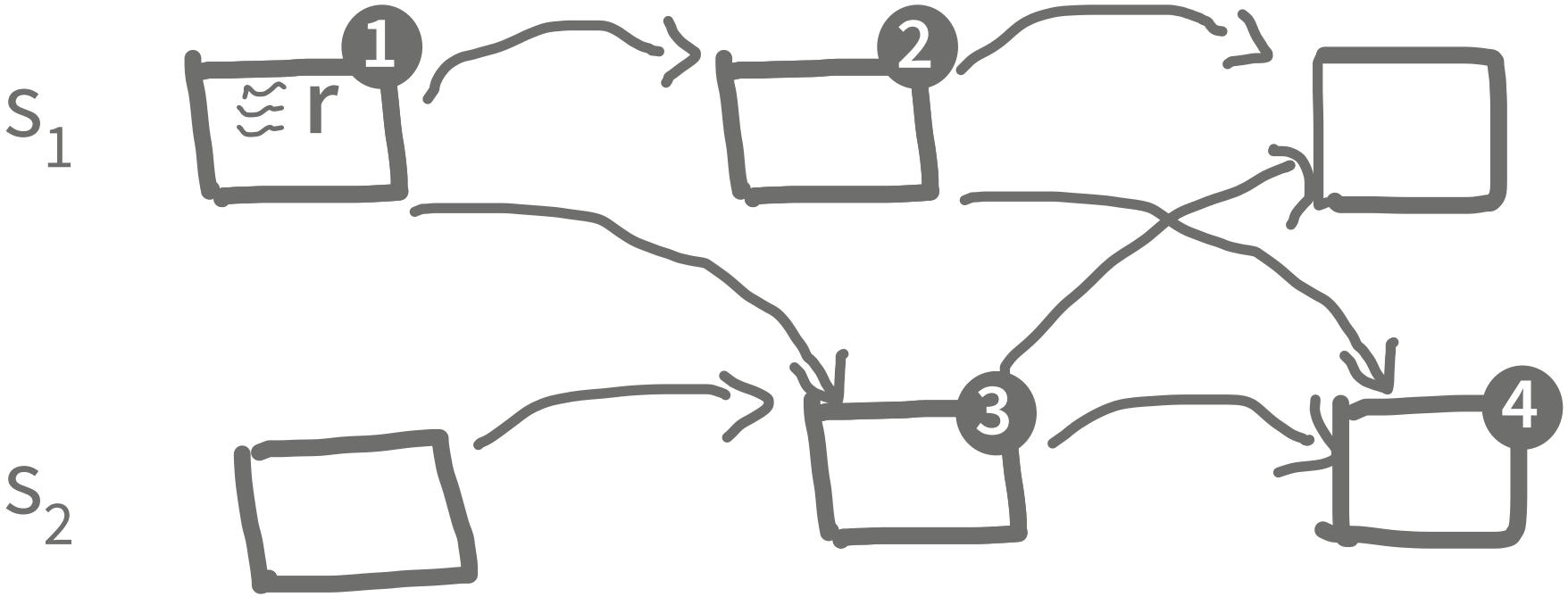
1 Build A Joint Block DAG

"log DAG"



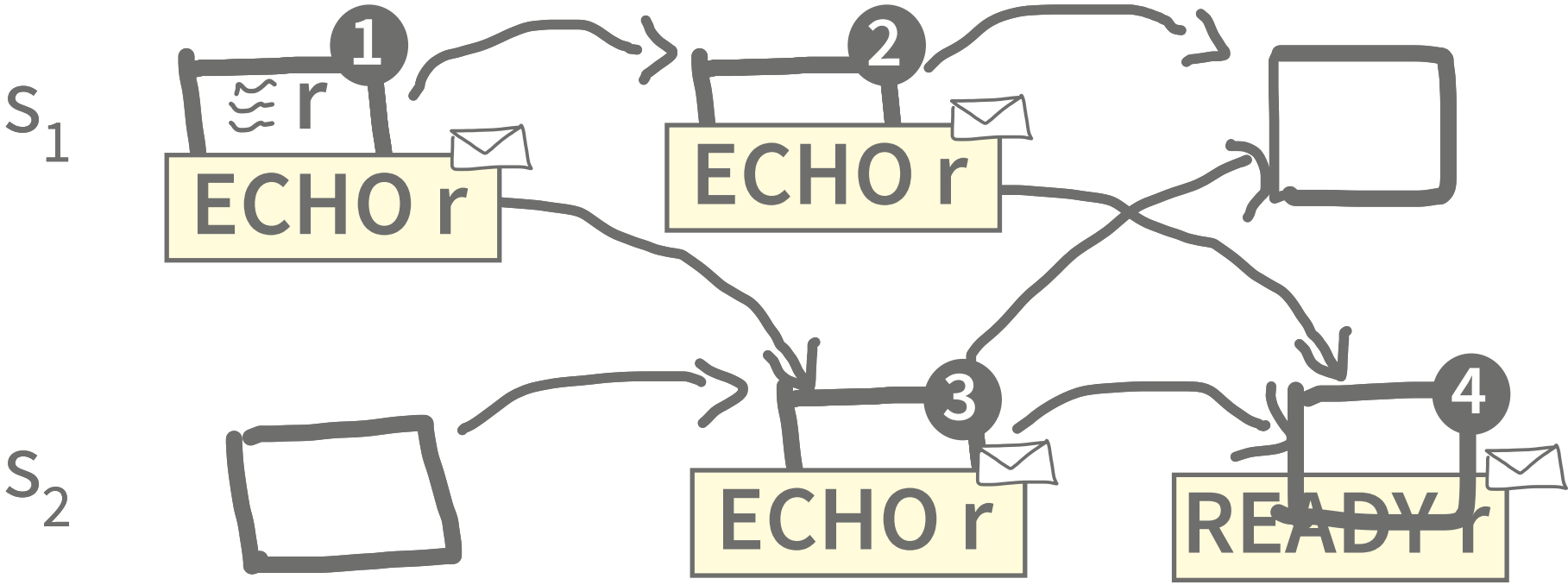
gossip protocol

2 Interpret $\mathcal{P}$:= reliable broadcast



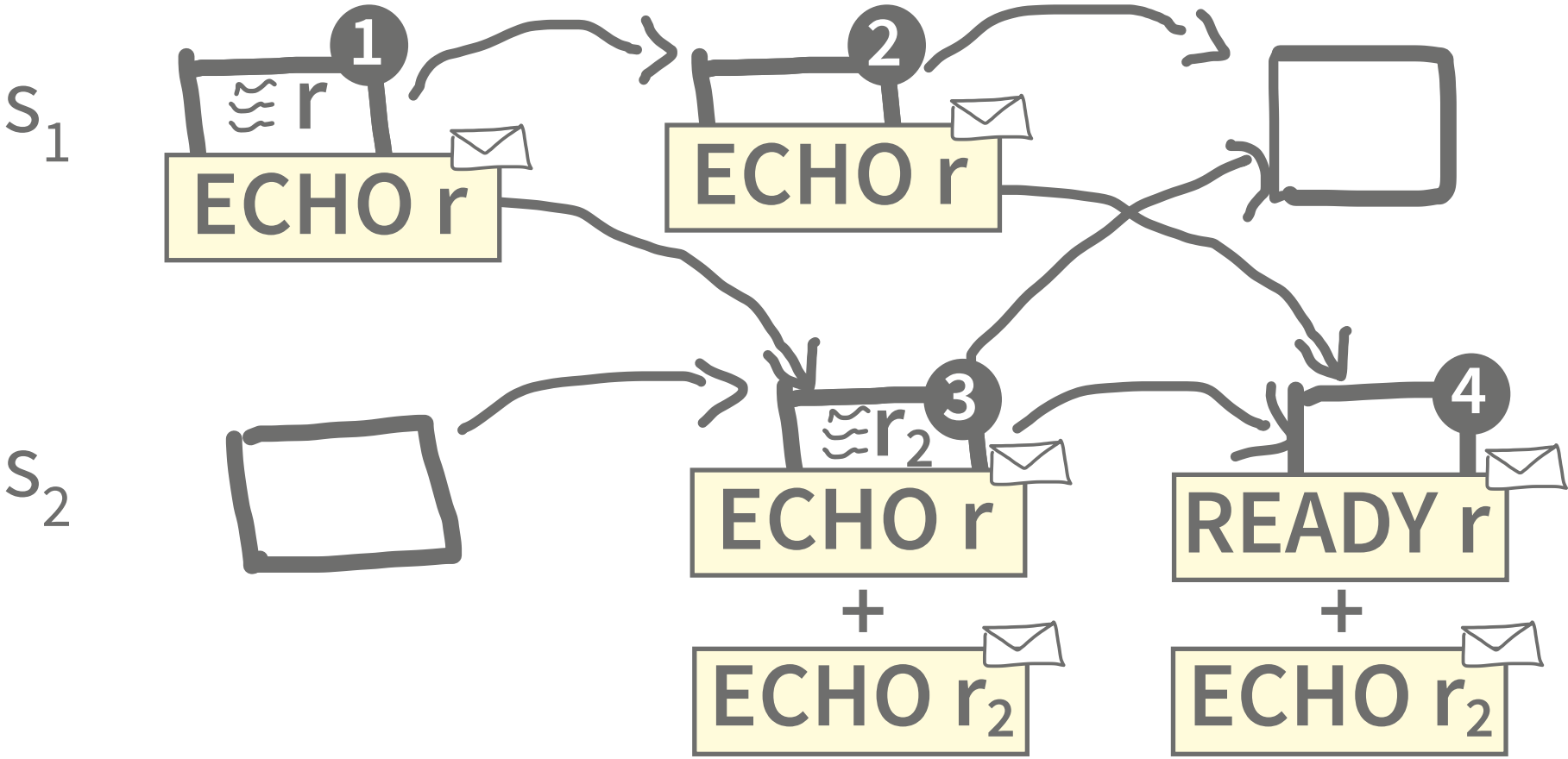
interpret protocol

2 Interpret P := reliable broadcast



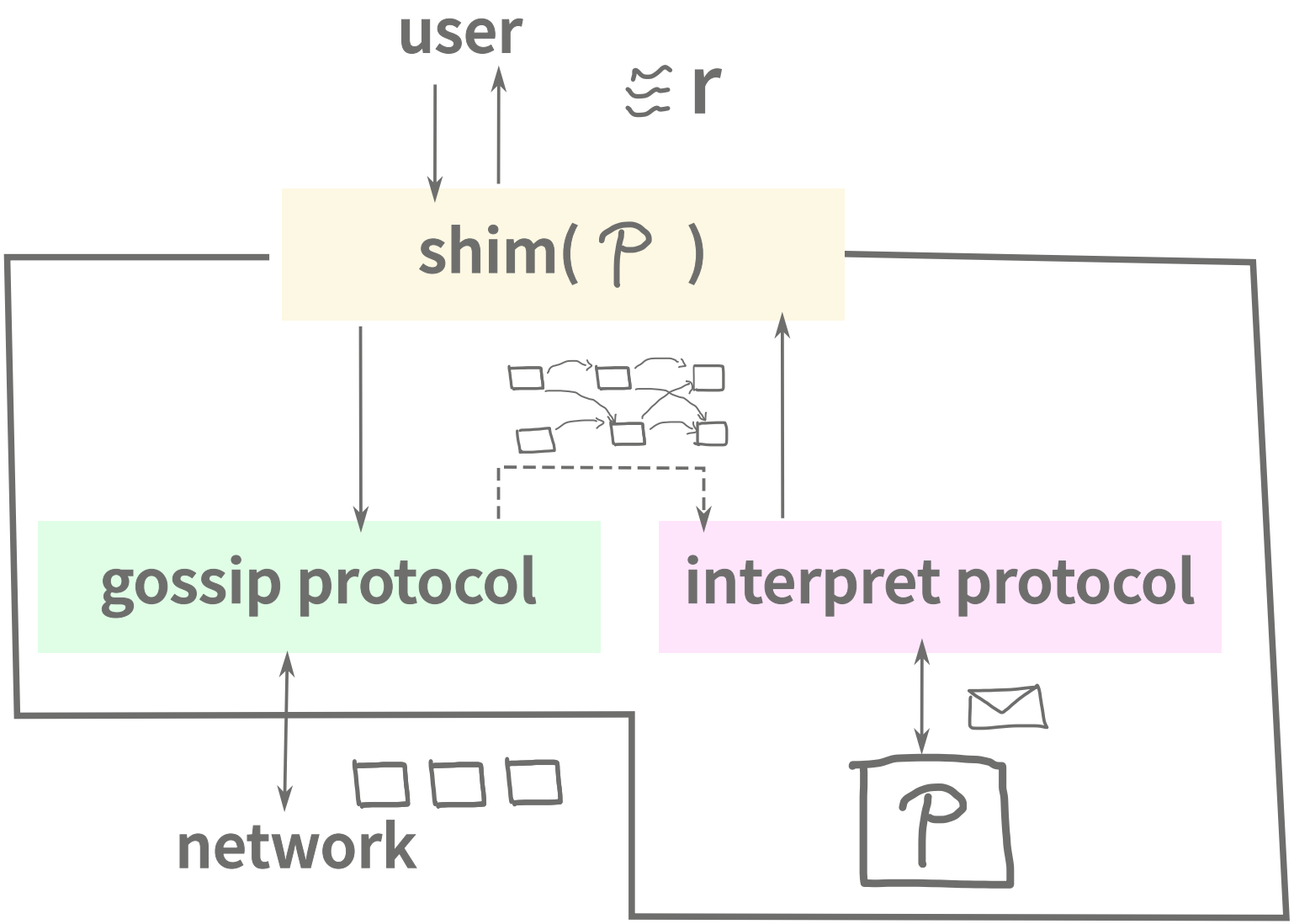
interpret protocol

2 Interpret $\mathcal{P}$:= reliable broadcast



interpret protocol

Block DAG Framework



Main Theorem

For every correct server

if protcol $\mathcal{P}$ has safety or liveness property $\mathbb{P}$

then shim($\mathcal{P}$) preserves $\mathbb{P}$.

Proof Idea: block DAG is a

reliable point-to-point link

Future Work

- **from deterministic** to **randomized** protocols
  - → write coin-flip in block
  - → shared coin in BFT setting
- **crash-and-recover** & **reconfiguration**
  - → replay protocol, but garbage collection
  - → pre-defined epochs

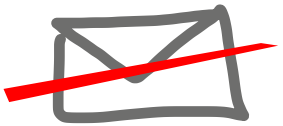
Conclusion

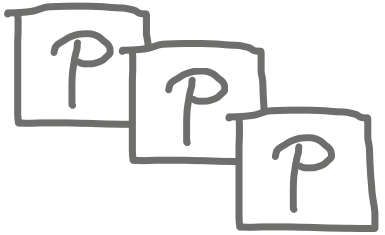
- framework to

  interpret a deterministic BFT protocol

  embedded in a block DAG

Why?

→ message compression 

→ parallel instances of 

Chat more? mail@maria-a-schett.net g.danezis@ucl.ac.uk