Blockmania QED.

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

DS @ FM'19 @ Porto

2019 | 10 | 07

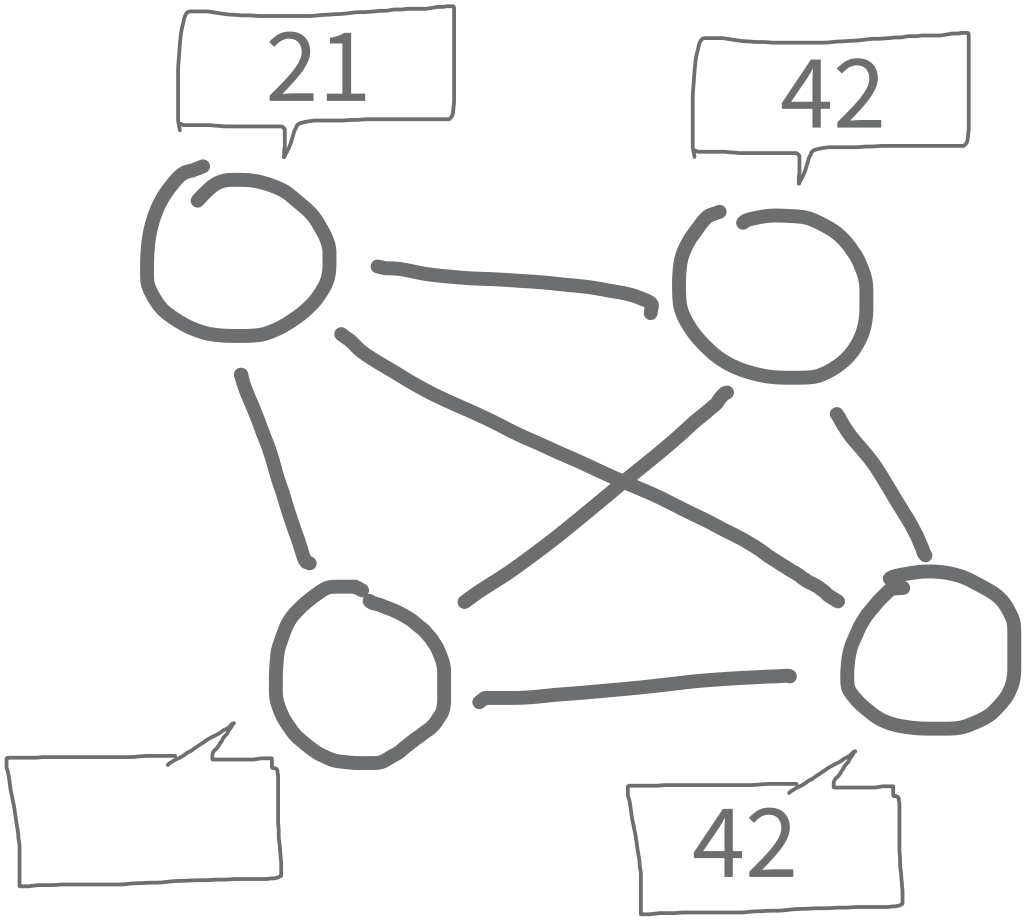
Goals (today)

(1) feedback: protocols & proofs

(2) suggestion: implementation

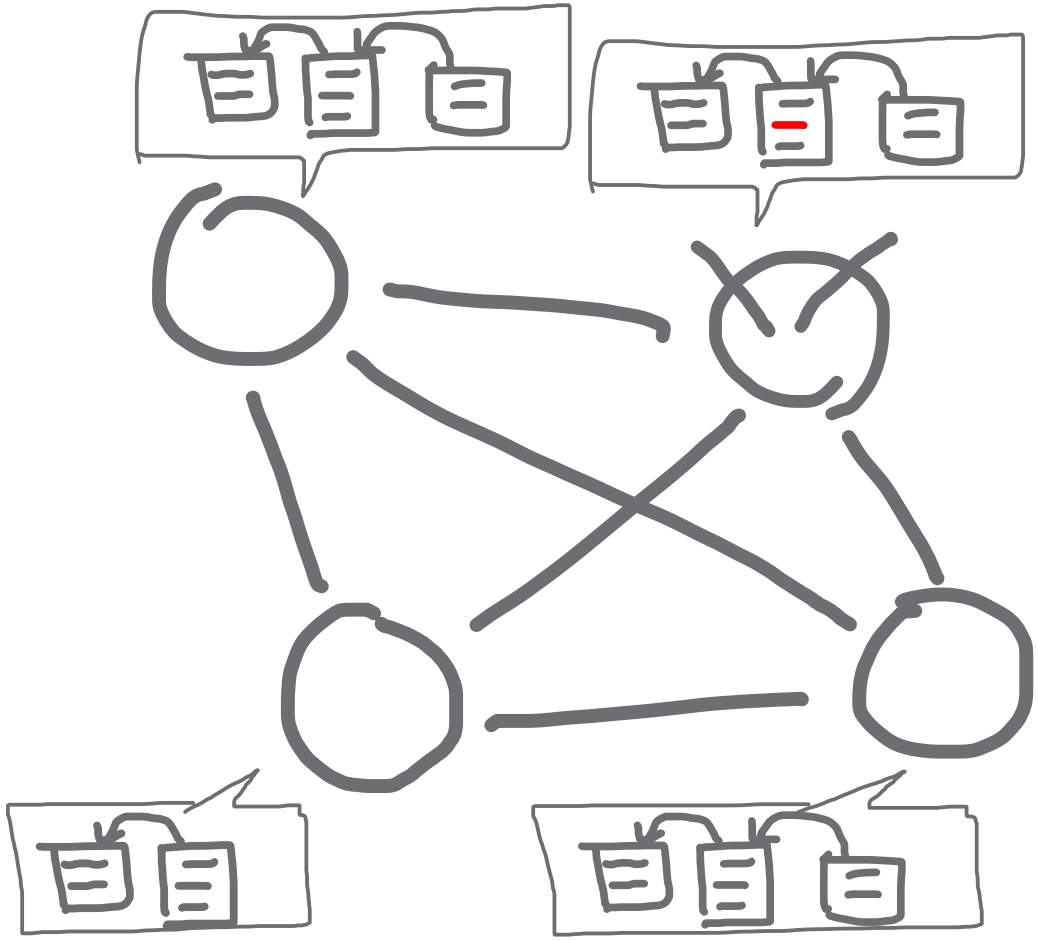
Background

shared ledger in distributed system



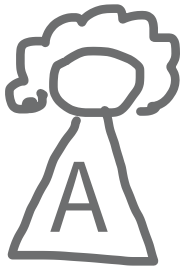
Background (2)

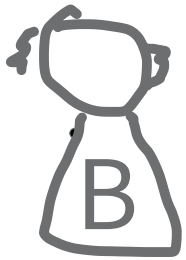
~~shared ledger in distributed system~~

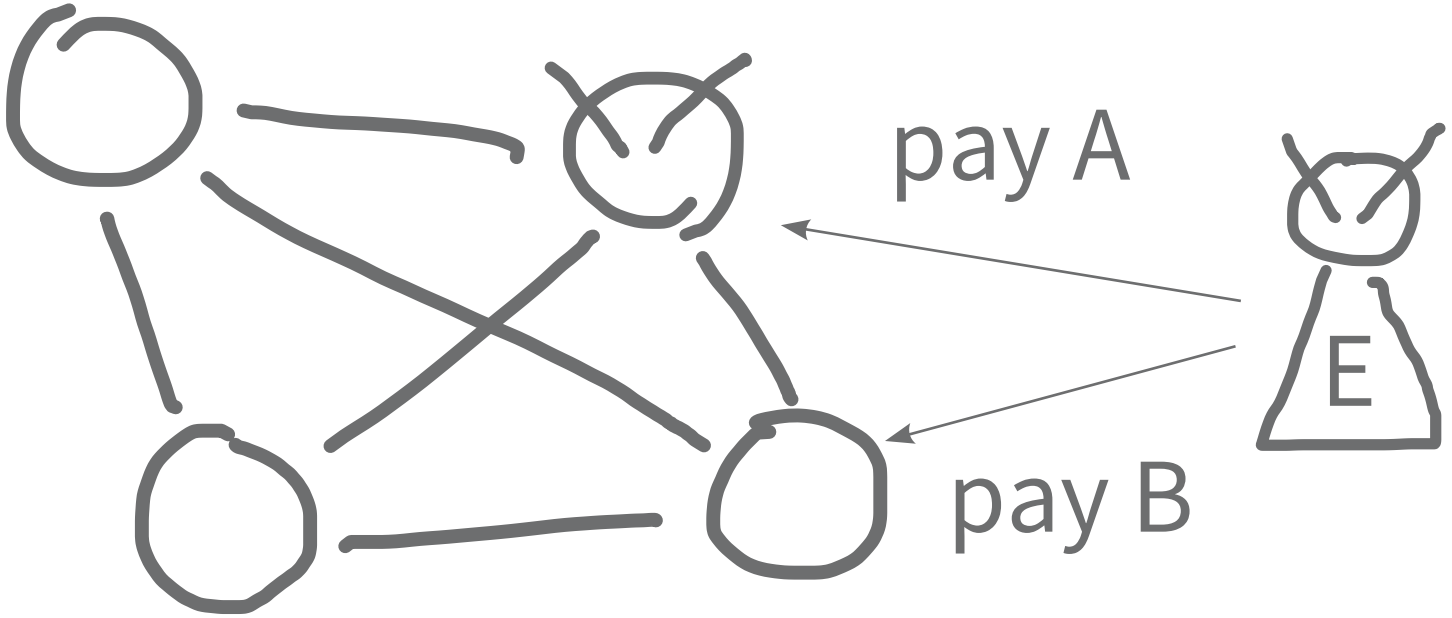


blockchain

Motivation: Consensus







credits: flat [CC], pizza by Pogrebnoj-Alexandroff [CC BY 2.5]

Challenges

a/synchronous

fail-stop: FLP [Fischer et al, 1985]

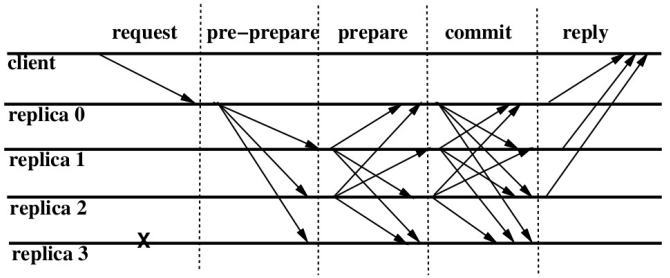
fail-arbitrary/
Byzantine [Lamport et al, 1982]

History

[Lamport et al, 1982] Byzantine

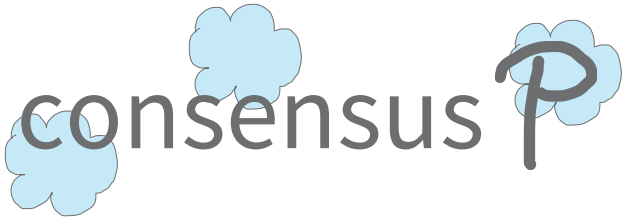
[Fischer et al, 1985] FLP

[Castro et al, 1999] PBFT



[Nakamoto, 2000] Bitcoin

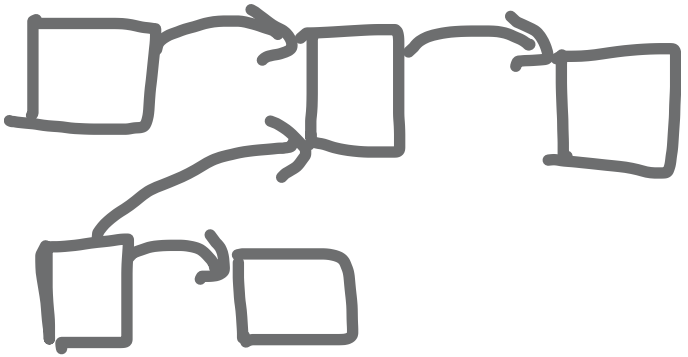
[now] 50+ consensus



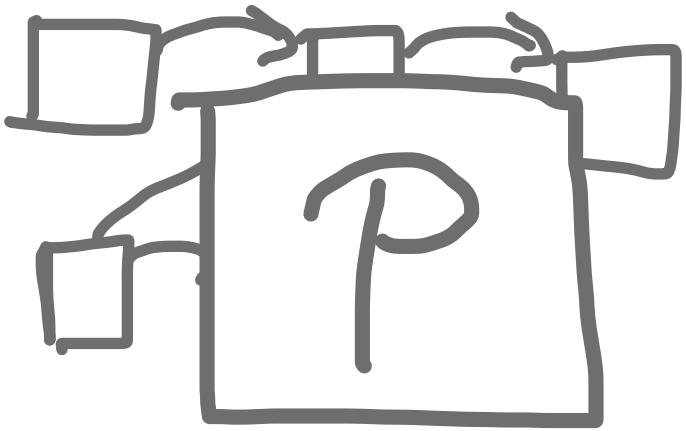
Blockmania [Danezis et al, 2018]

consensus protocol for 

1 block graph

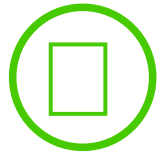


2 interpret--
don't execute



P ~ PBFT [Castro et al, 1999]

Blockmania QED.

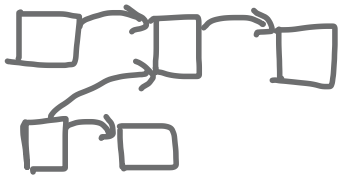
Goal 2: show Blockmania correct 

* safety: agreement, non-triviality, integrity
* liveness: termination

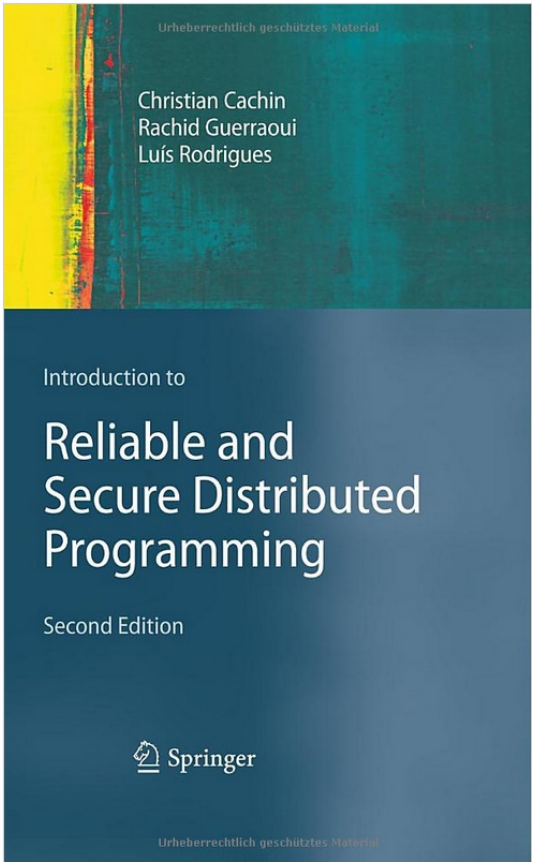
Goal 3: deconstruct Blockmania

Blockmania QED.

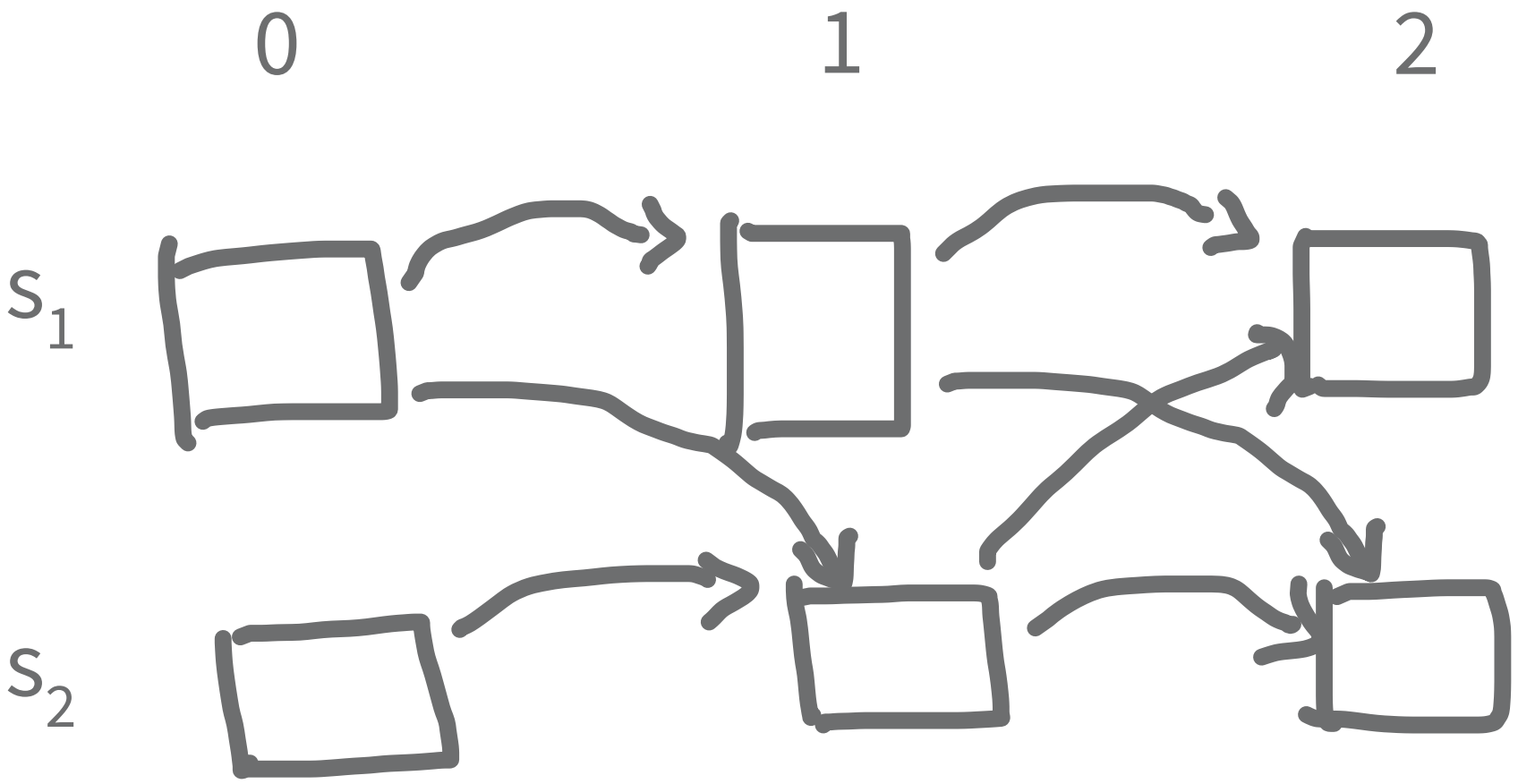
Goal 1: Blockmania formalize

* block graph 

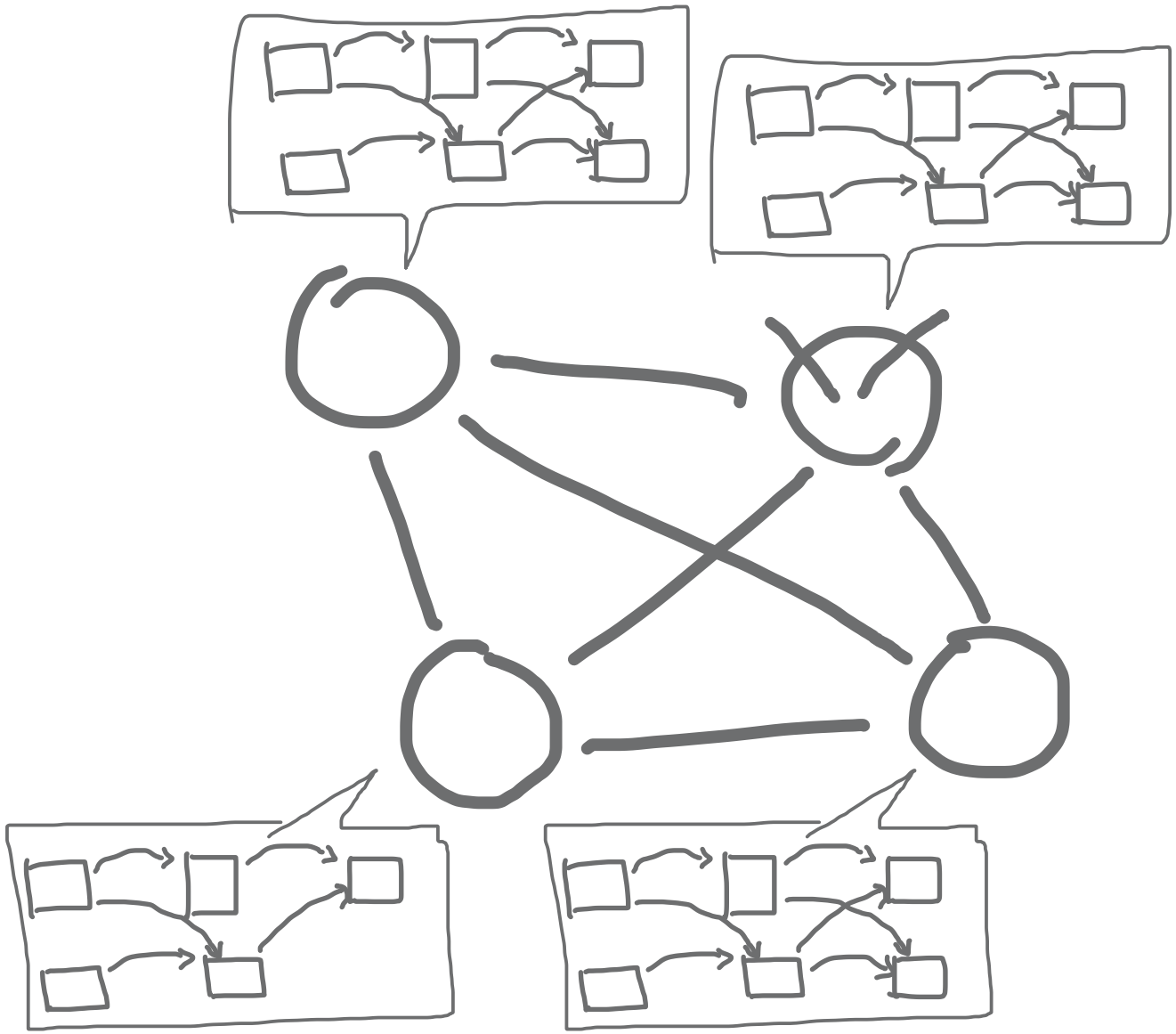
* pseudo-code 



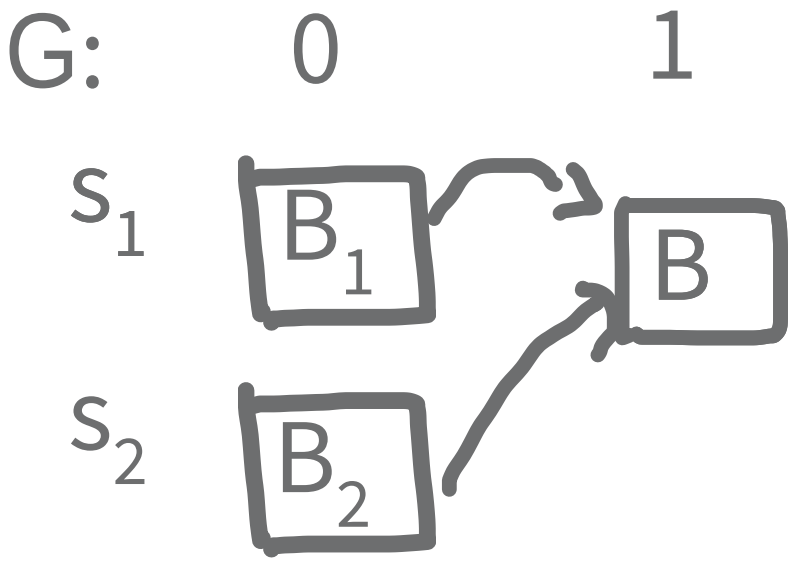
Block Graph



Joint Block Graph



Block Graph ~ DAG



B

server id: s_1

seq nr: 2

preds: [ref(B_1), ref(B_2)]
with B.parent = B_1

sign: σ

rqst: [tx_1 ... tx_m]

...and all blocks are valid.

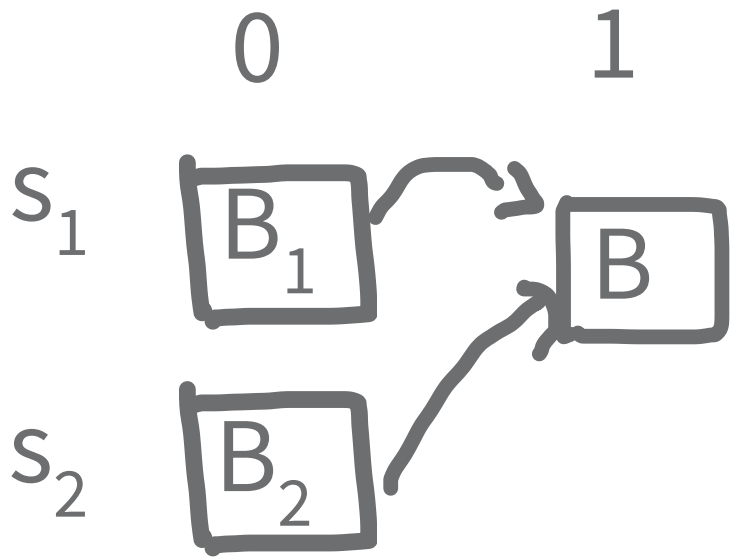
Valid Block

"s considers B valid"

(1) s verifies B.σ

(2) either B.k = 0
or B.parent.k = B.k-1

(3) s considers all B_i in B.preds valid

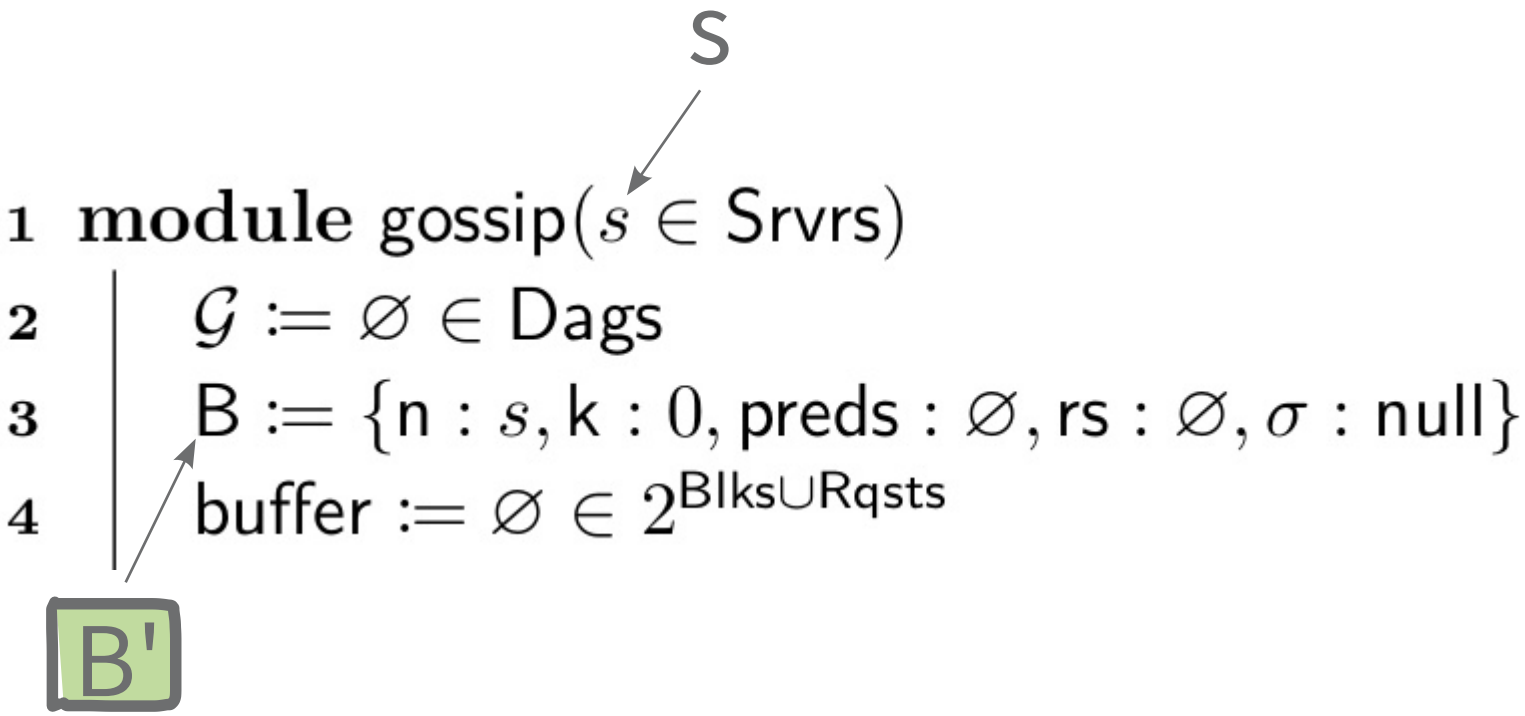


Joint Block Graph

Lemma.
correct s receives & considers B valid

□

every correct server will eventually receive B


s
1 module gossip($s \in$ Srvrs)
2 $\mathcal{G} := \varnothing \in$ Dags
3 B := {n : s, k : 0, preds : $\varnothing$, rs : $\varnothing$, σ : null}
4 buffer := $\varnothing \in 2^{\text{Blks} \cup \text{Rqsts}}$
B'

s builds B'

s receives & considers B valid

```
9   when valid(s, B) for some B ∈ buffer
10      buffer := buffer \ {B}
11      G := insert(G, B)
12      (interpret_block(B))
13      B.preds := B.preds ∪ {ref(B)}
```

s inserts ref(B) to B'

```
17 |  when full(B) or C_B
18 |  |  B.σ := sign(s, B)
19 |  |  broadcast B
```

s broadcast B' with ref(B)

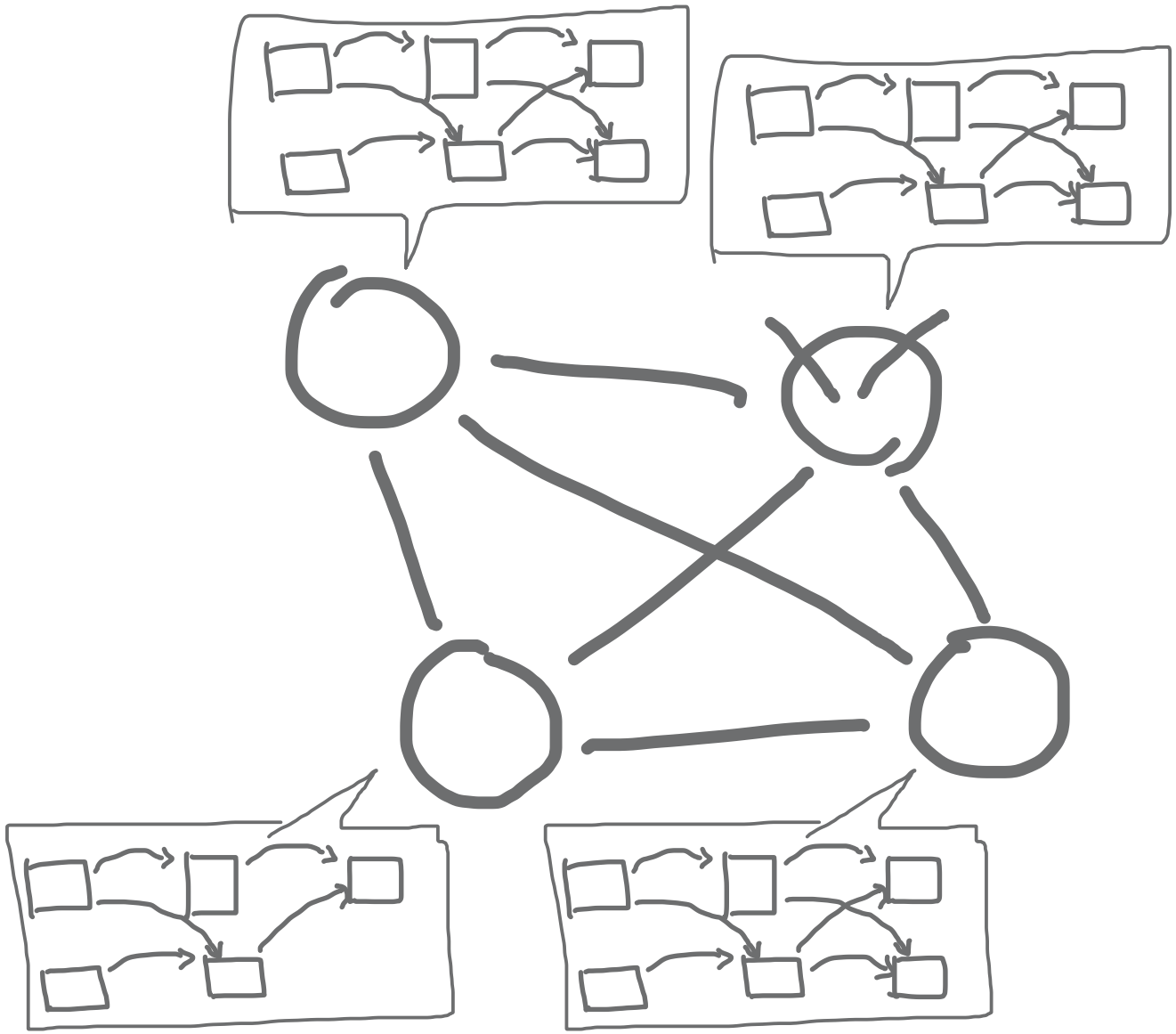
creator of  is 

```
21  when received FWD B from s' and B ∈ G
22      send B to s'
23  when B ∈ buffer after time Δ_B
24      for B' ∈ B.preds where B' ∉ buffer
25          send FWD B' to B.n
```

s' asks s for 

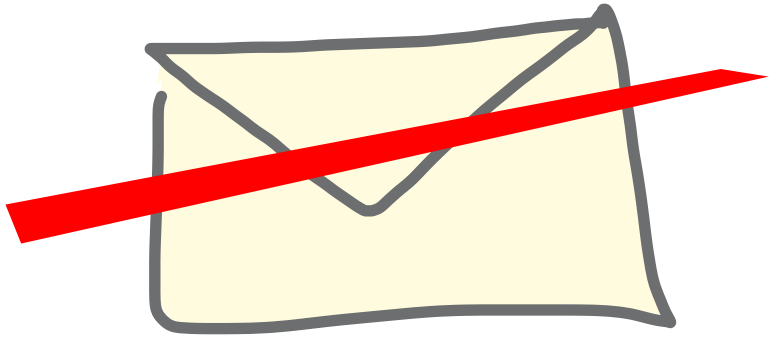
Future Work

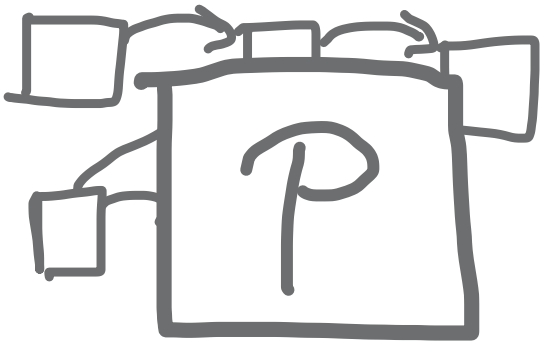
Joint Block Graph



Interpret--Don't Execute!

if $\mathcal{P}$ deterministic





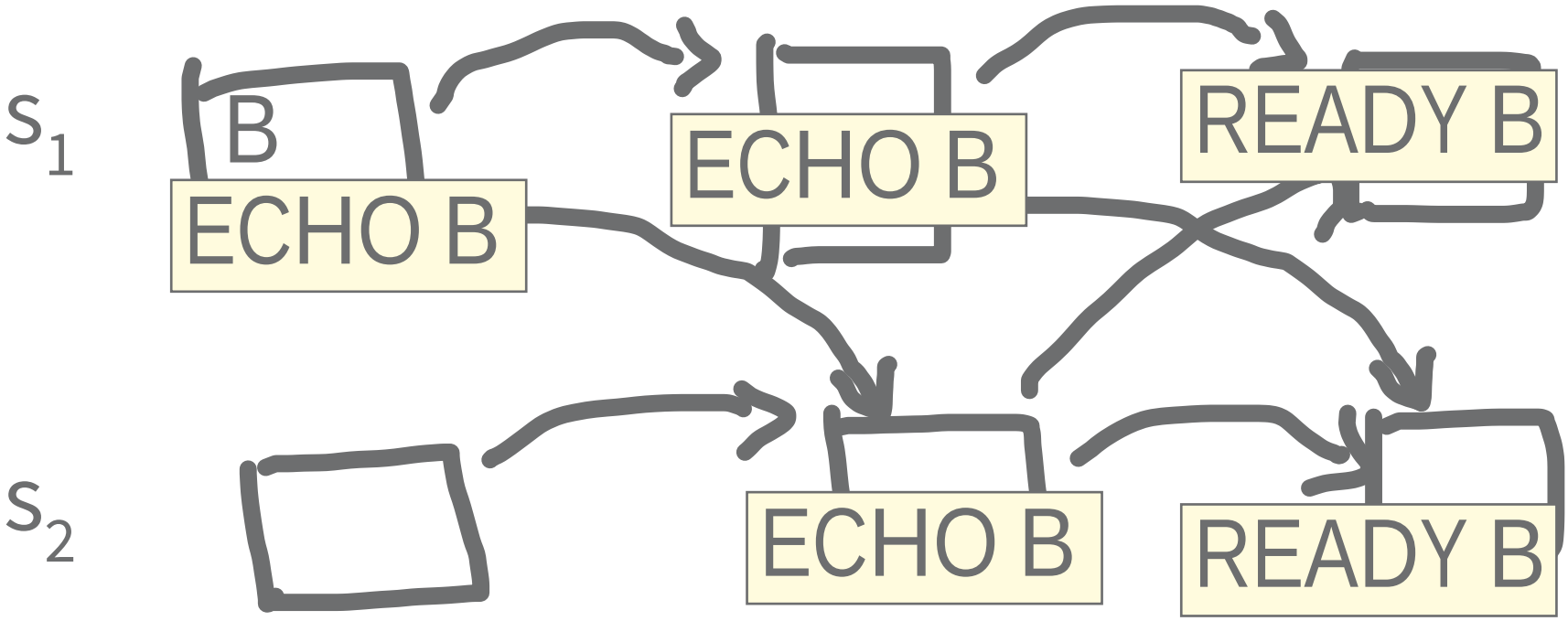
$\mathcal{P}$ = Bracha Broadcast [Bracha et al, 1985]

ECHO B

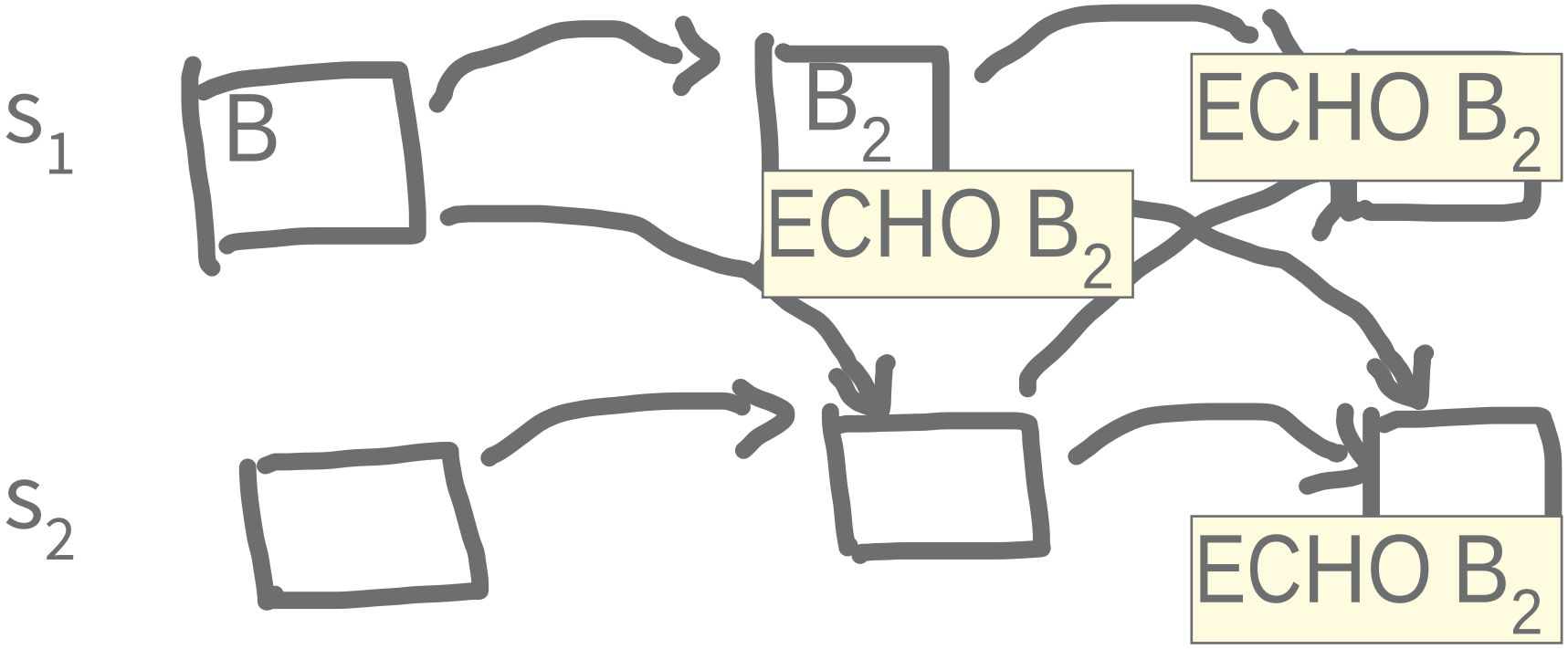
READY B

12 | | interpret_block(B)

Interpret



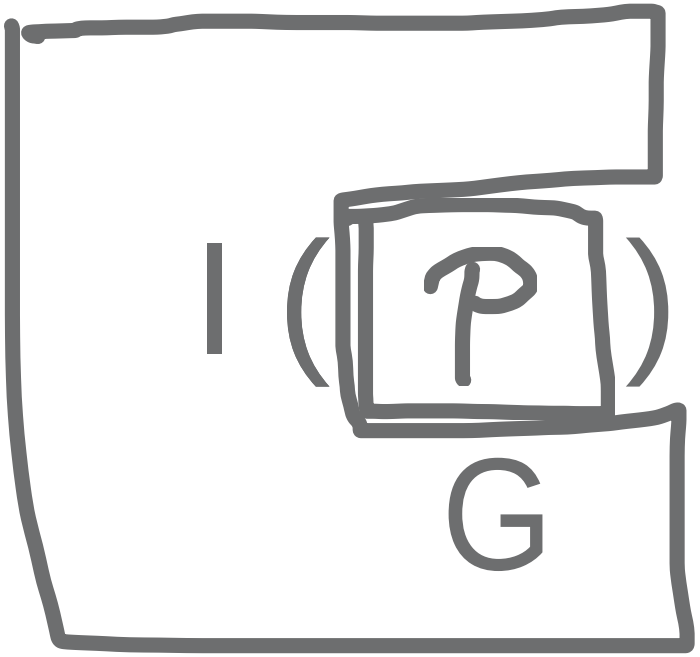
Interpret

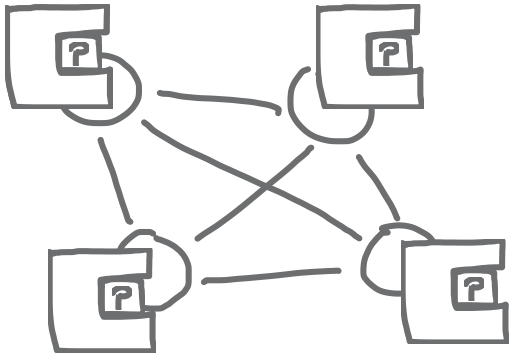


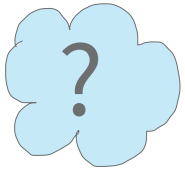
Endgoal

For executing $\mathcal{P}$ holds P □
for interpreting $\mathcal{P}$ holds P.

Implementation





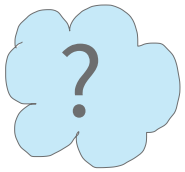


discrete event/network simulators

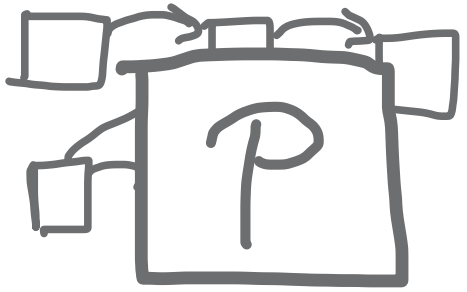
protocol $\mathcal{P}$

Thank you!

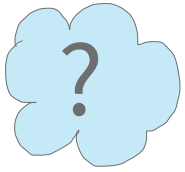
(1) feedback: protocols & proofs



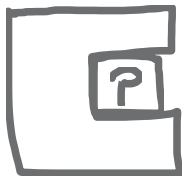
ad-hoc □ framework



(2) suggestion: implementation



discrete event/network simulators
protocol $\mathcal{P}$



Bibliography

[Lamport et al, 1982] Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems 4/3.

[Bracha et al, 1985] Bracha, G., and Toueg, S. (1985). Asynchronous Consensus and Broadcast Protocols. J. ACM 32.

[Fischer et al, 1985] Fischer, M.J., Lynch, N.A., and Paterson, M.S. (1985). Impossibility of distributed consensus with one faulty process. Journal of the ACM (JACM) 32.

[Castro et al, 1999] Castro, M., and Liskov, B. (1999). Practical Byzantine Fault Tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation.

[Nakamoto, 2000] Nakamoto, S. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System.

[Danezis et al, 2018] Danezis, G., and Hrycyszyn, D. (2018). Blockmania: from Block DAGs to Consensus.