Episode 10: Storage & Networks

Storage

storage

- (1) Background: ACID & 2PC
- (2) Sagas ⇒ Distributed Saga
- (3) DB patterns (VLDB '22)

Network

n/w

- (1) RPCs over the N/W
- (2) LB ⇒ client LB ⇒ Service Mesh
- (3) Head Of line Blocking (HOL) & Queuing
- (4) R2P2 (deployments)

ACID

Atomicity

"all or nothing"

Consistency

preserve system specific Consistency semantics / invariants

Isolation

individual transactions do not effect or see intermediate state of other transactions

Durable

logs to ensure commits live forever

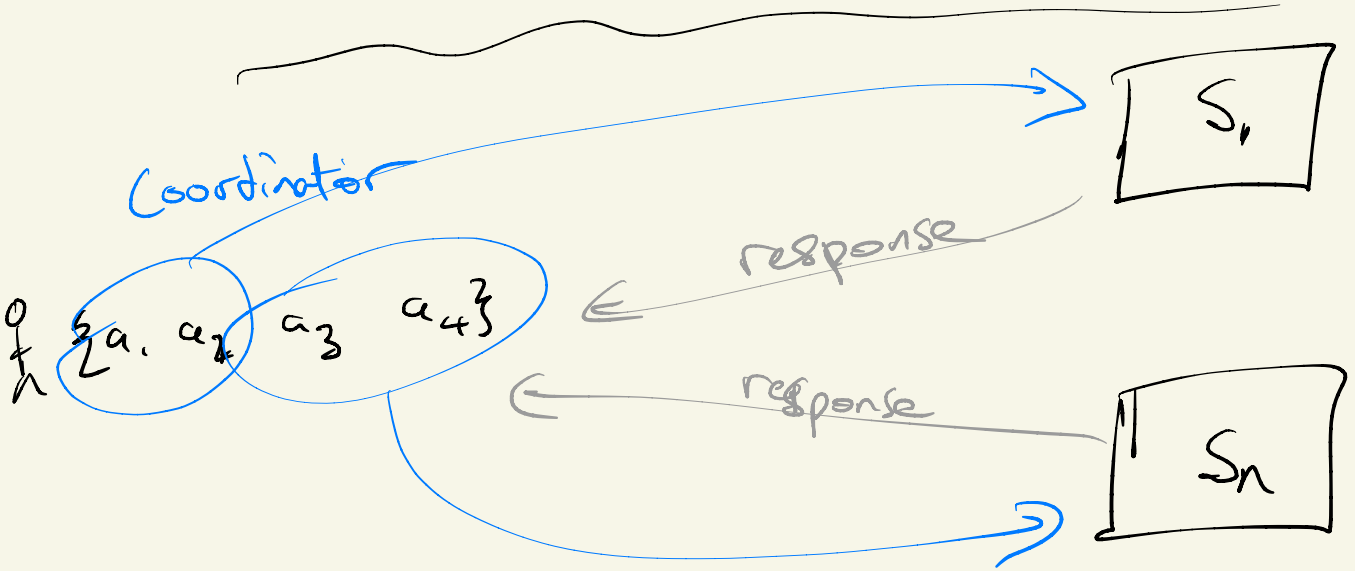
Transaction

$Tx = \{a_1, a_2, a_3, \ldots, a_n\}$ ⟸ ACID

nothing

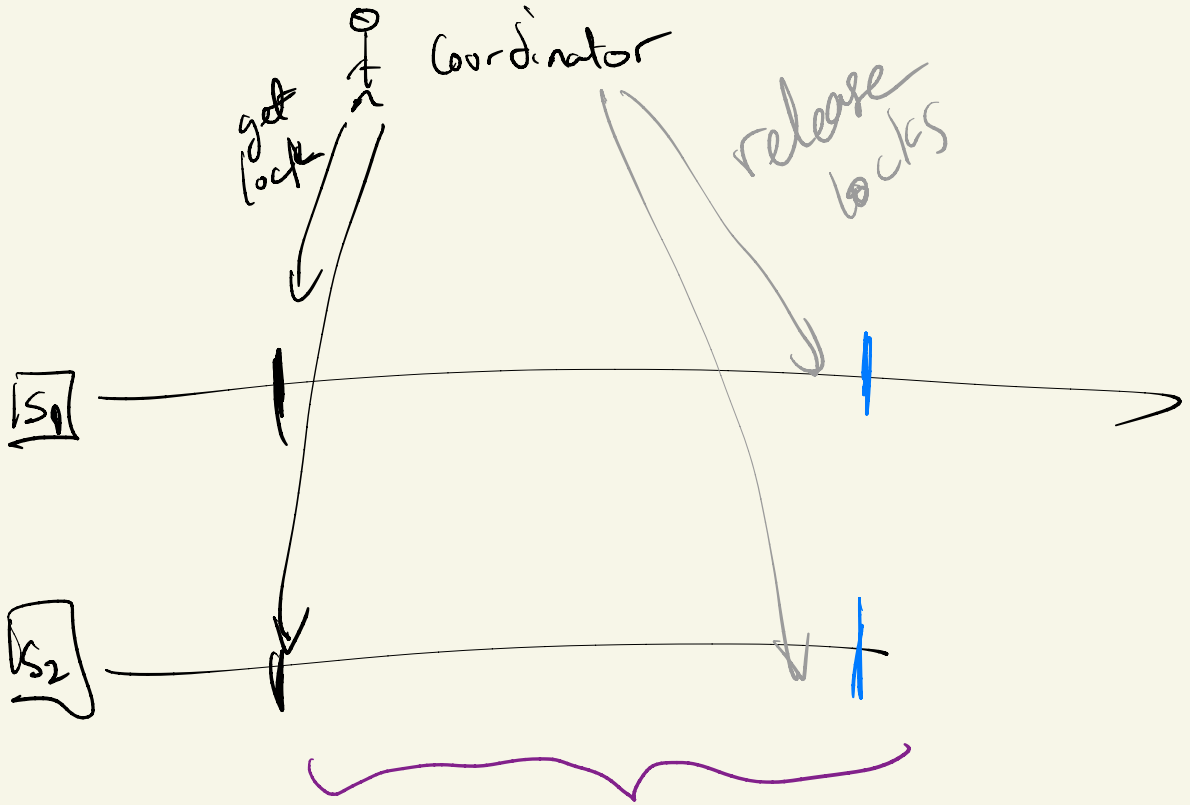
all actions have been executed

Two Phase Commit



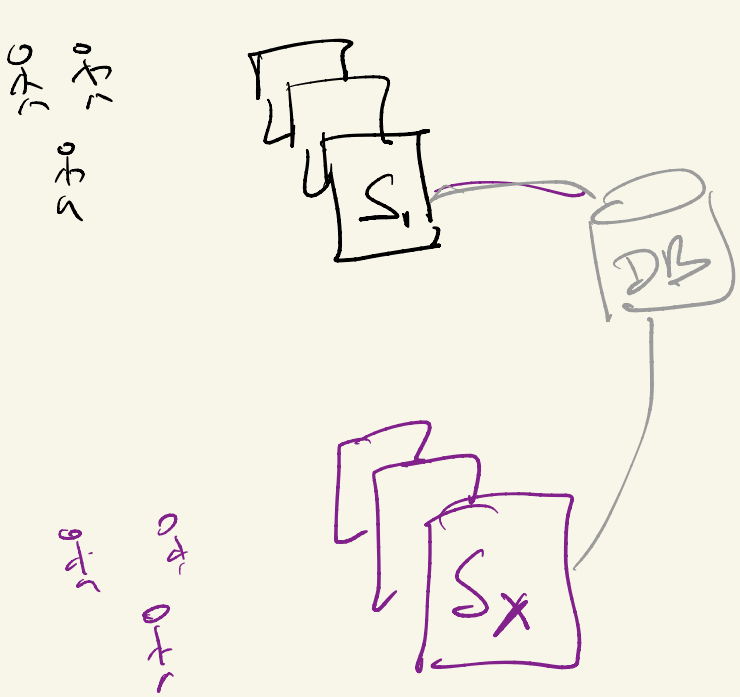
phase 1 (prepare T) : asks servers to vote

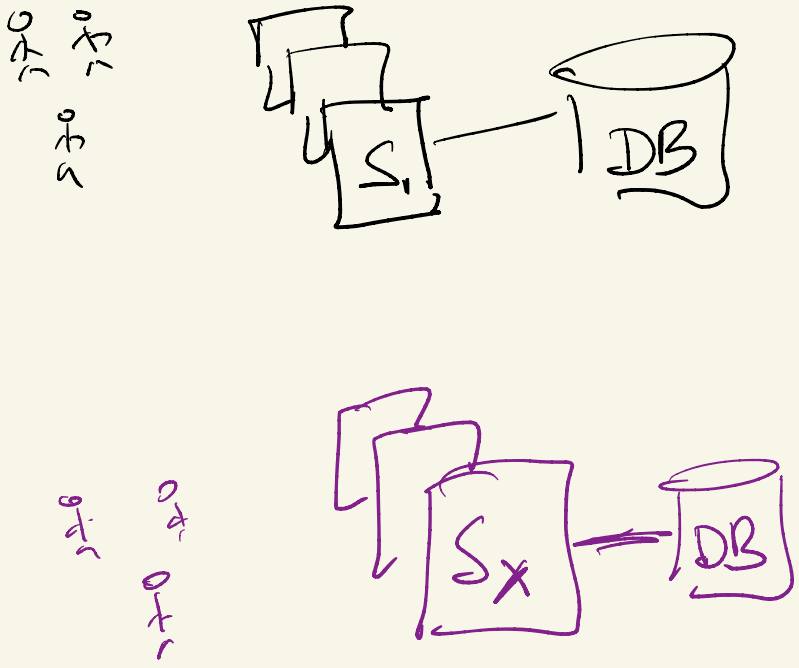
phase 2 (commit/abort) : if all servers vote Yes
Then Commit
else
Abort


Coordinator
get lock
release locks
S_1
S_2
Servers unavailable
for other transactions

Databases in Microservices

Most Common





One DB for company

one DB per service

Based on benefits this would be popular

single DB

Pro

- from complexity; easier to manage
- easier to manage consistency

Con

- requires coordination
- Performance

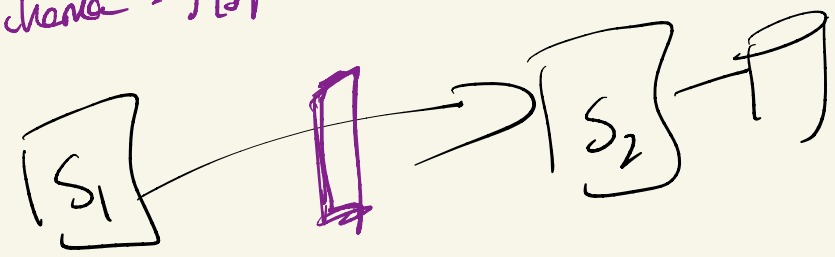
many DBs

Con

- DB schema evolves quick & RPC interface will also

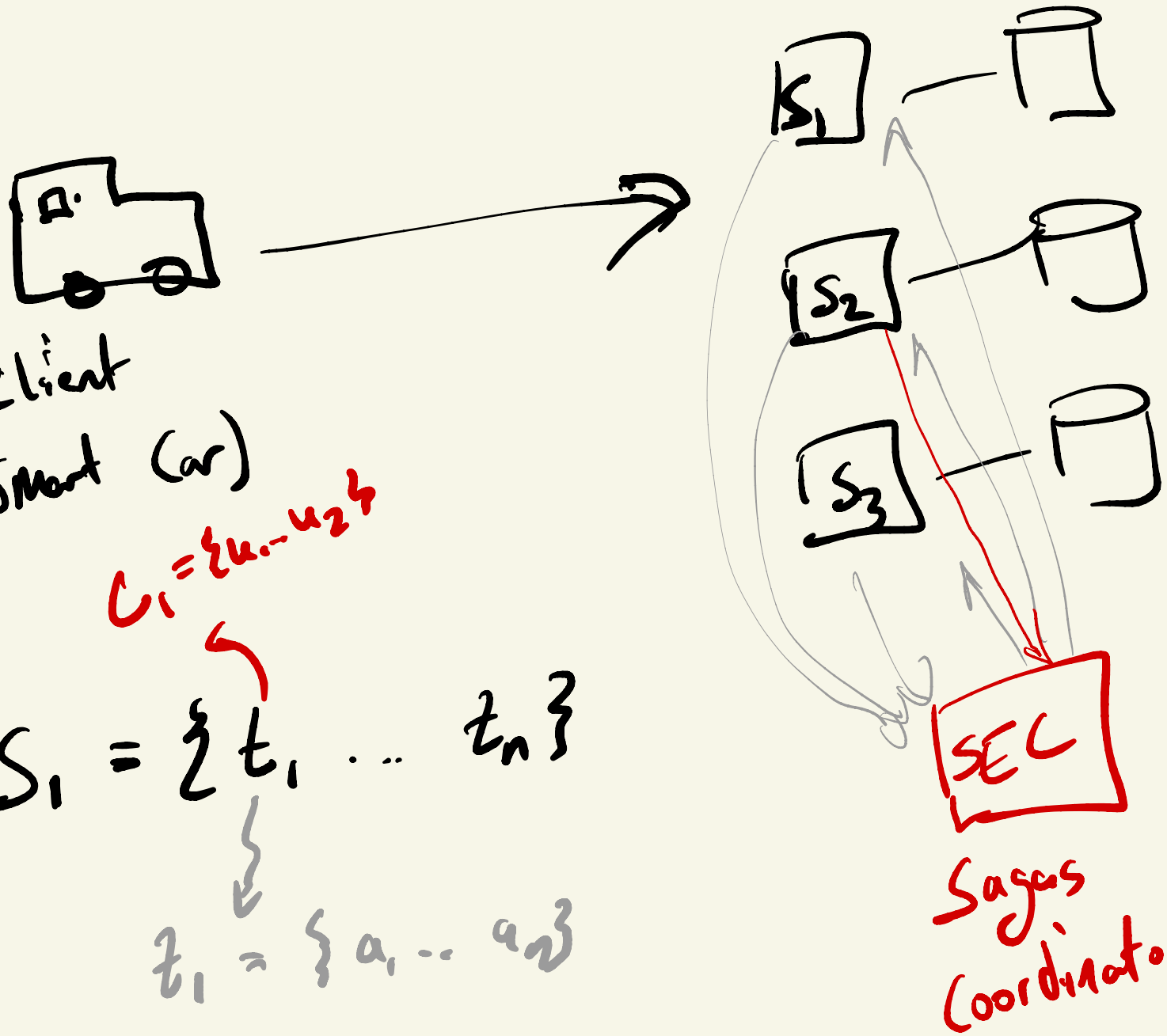
schema 1.1 ⇒ 3.2

schema ⇒ 1.1



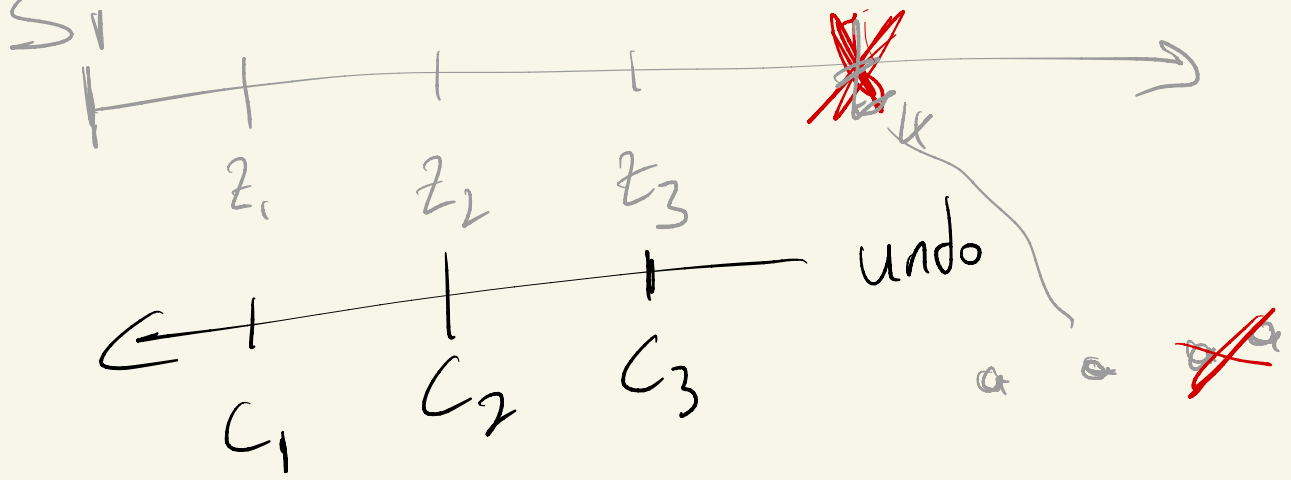
translation

Complexity


IDEAL: ACID Semantics
client
(Smart car)
$C_1 = \{u_1 \dots u_2\}$
$S_1 = \{t_1 \dots t_n\}$
$t_1 = \{a_1 \dots a_2\}$
S_1
S_2
S_3
SEC
Sagas Coordinator

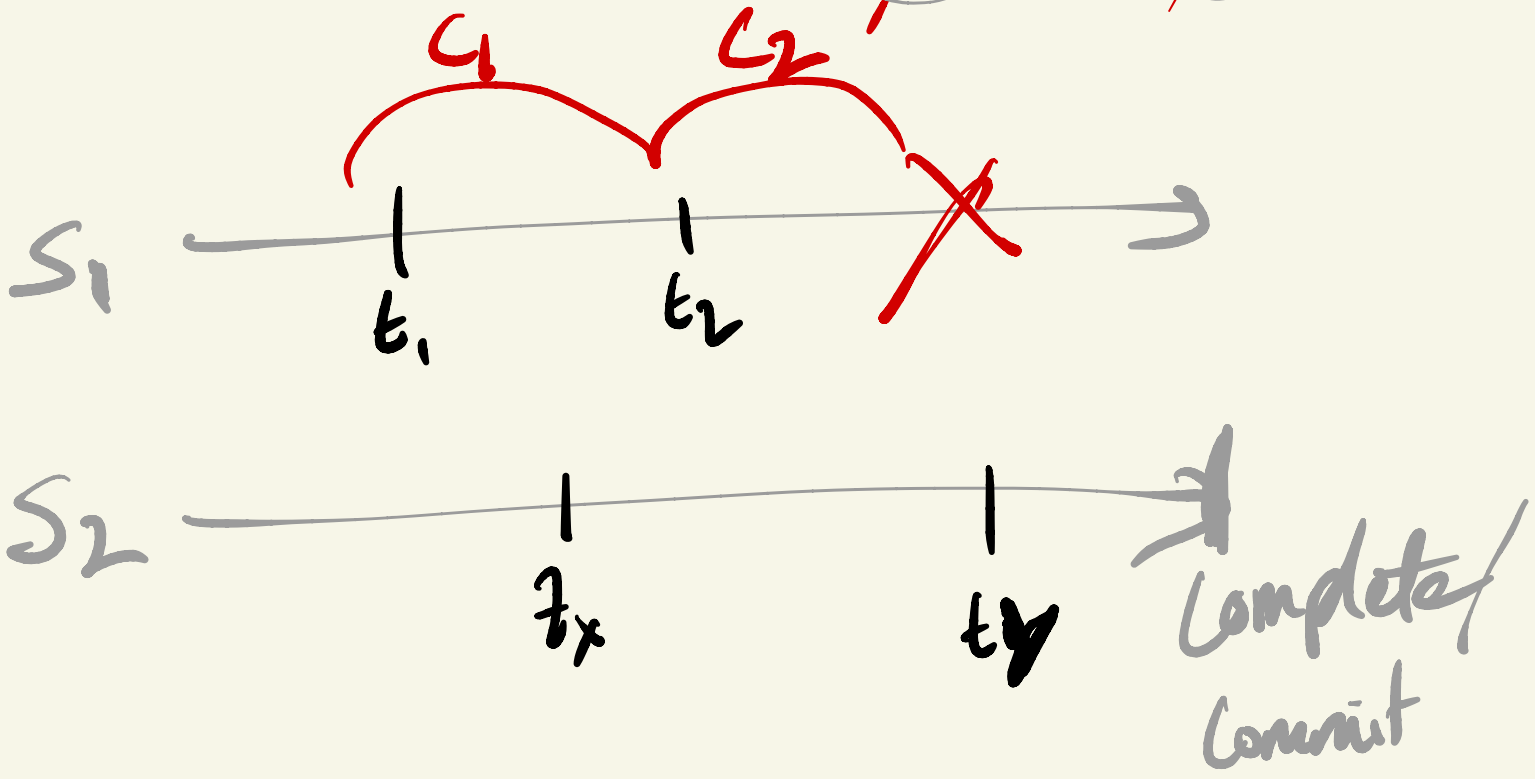
failure

Sagas fails ⇒ "Backward recovery" undo the committed transaction



Transaction fails ⇒ regular 2PC will ensure nothing is commit

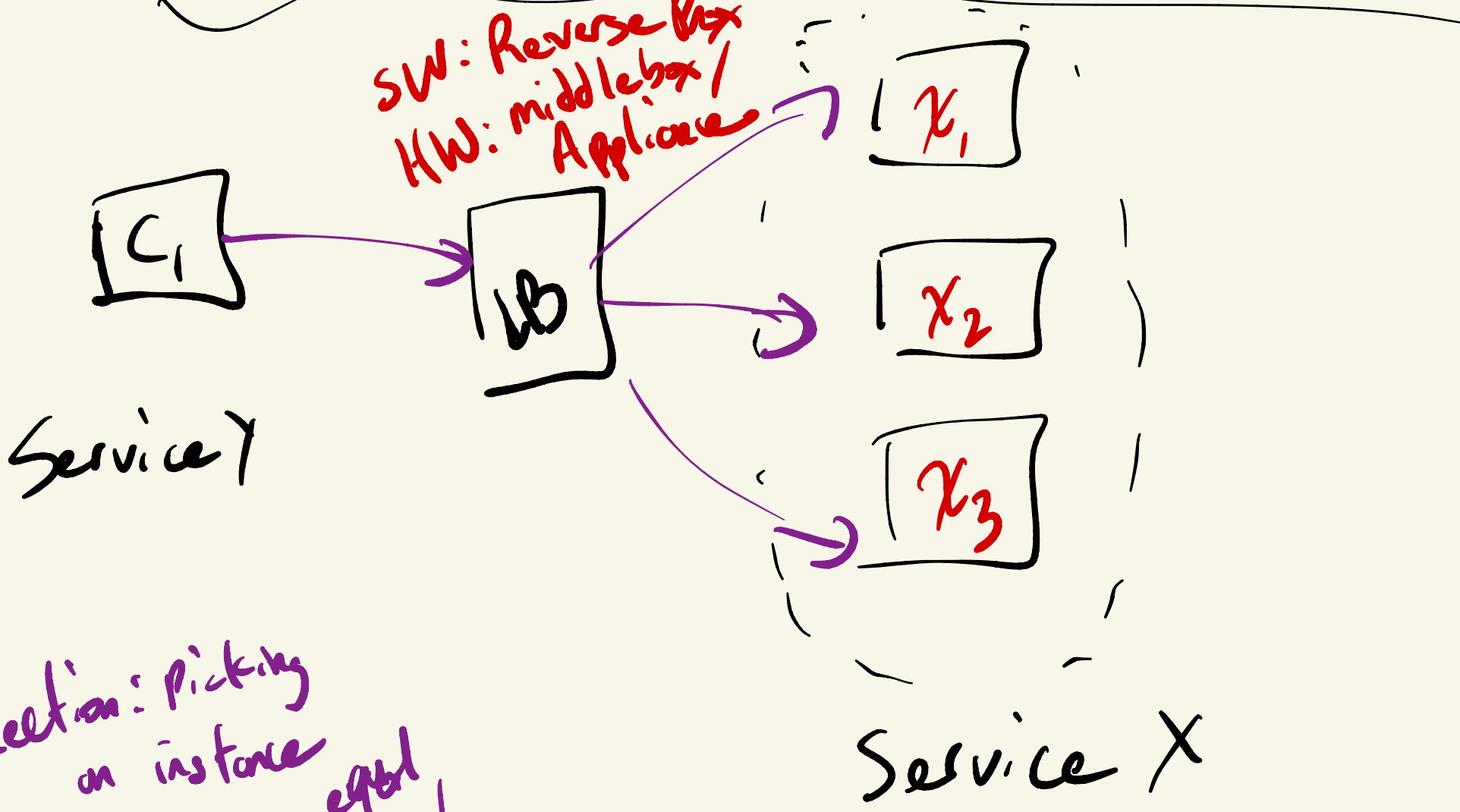
SAGA = ~~A~~ C ~~I~~ D



"local consistency"


Networking

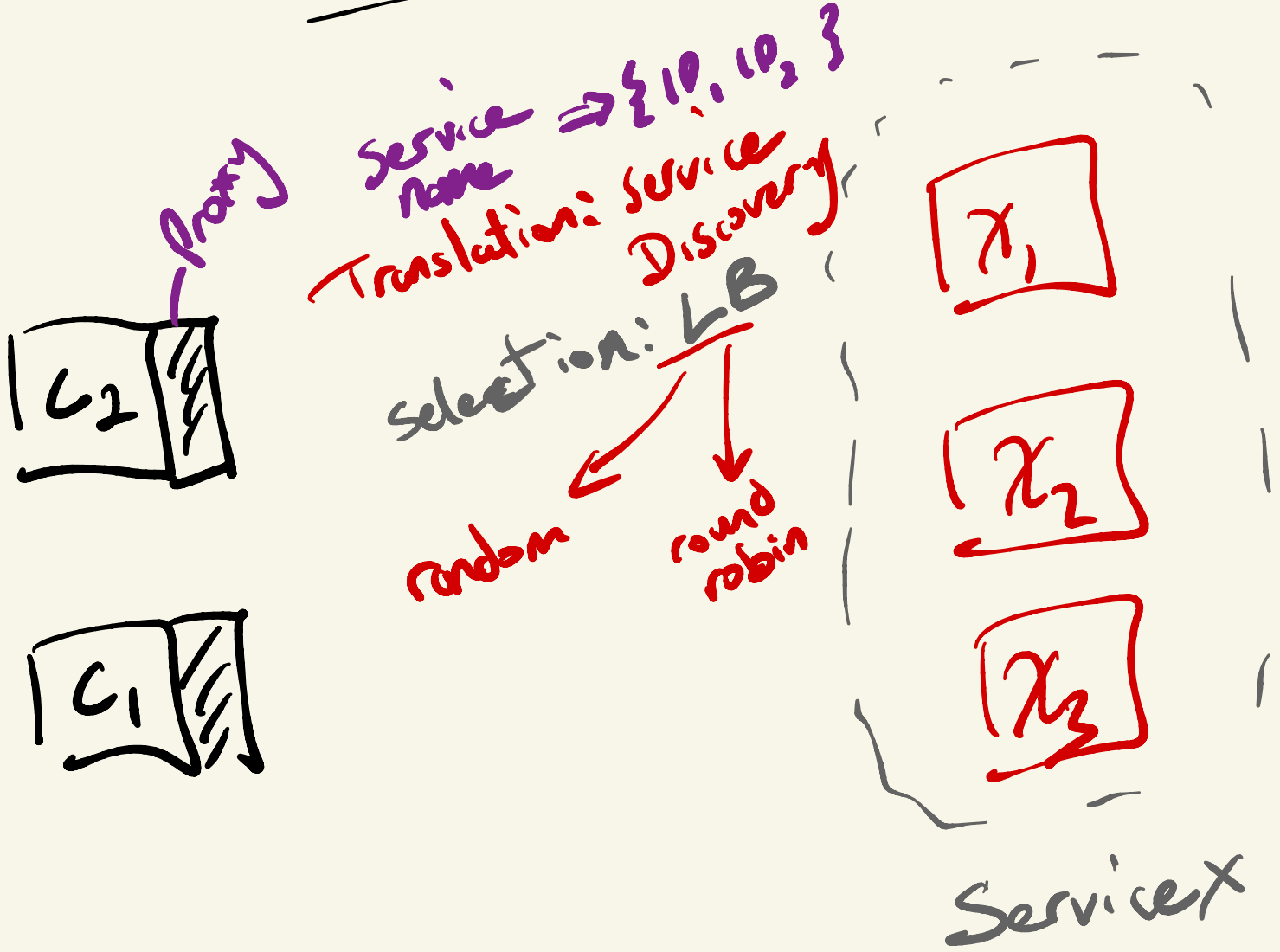
RPC (Remote Procedure calls)



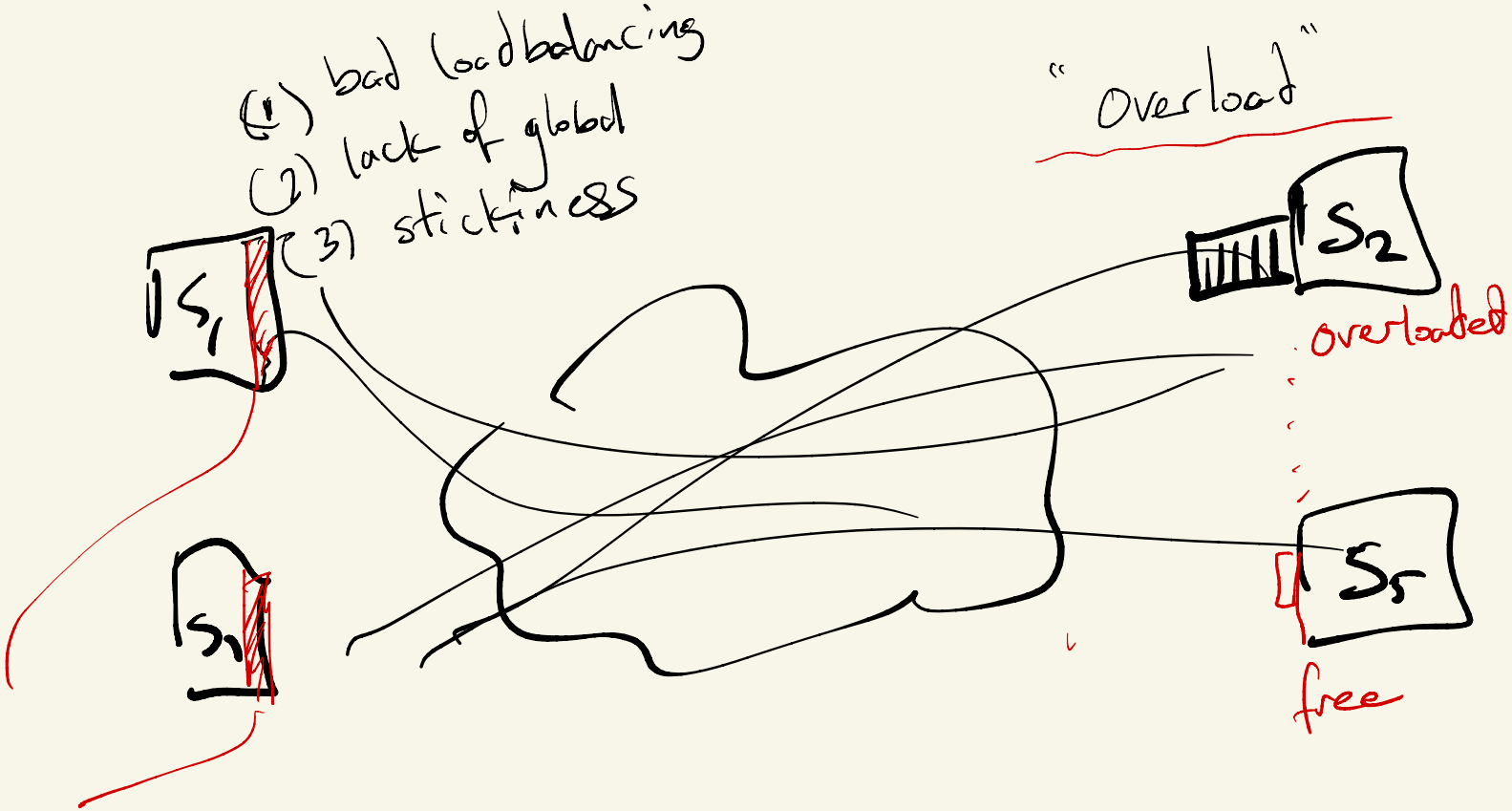
Selection: picking an instance

LB: ensure eqbl utilization / speed

Service Mesh



Break until 4:15



Implement s/w RPC in ServiceMesh

Proxy

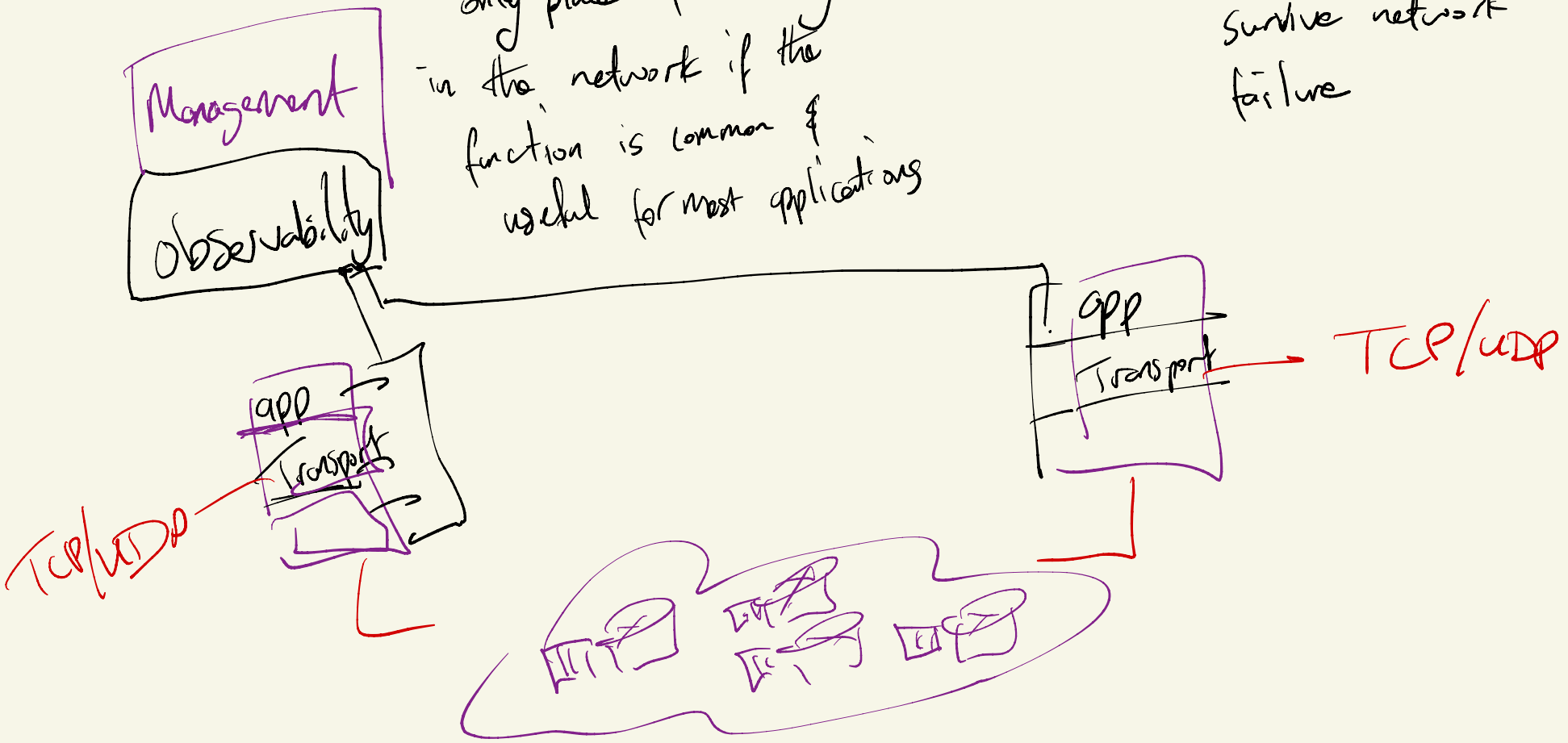
Visibility: for close to perfect they need load info from the servers

E2E

Where do you place functionality : in the network or at the end host

only place functionality in the network if the function is common & useful for most applications

functional should survive network failure



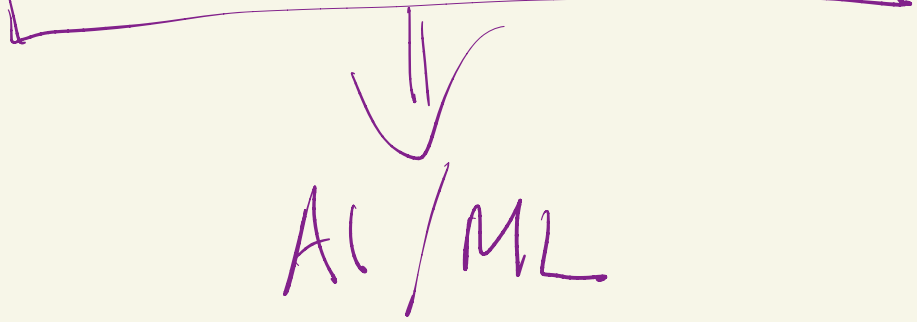
Operational Issues

(1) Observability/Management stack needs to change (new code & new heuristic)

(2) Network Routing needs to change (picking protocols & configuring)

Accelerators

Special purpose hardware that are great at certain types of computation : GPU : TPU : FPGA



AI / ML

(1) Scheduling
(2) Virtualization / security
(3) Profiling

Today "Other Resources"

* Storage
  * background
  * paradigms
  * Sagas
* Network
  * RPC / selection / LB ⇒ Appliance; ServiceMesh
  * BGP2
  * Provided Context
  * Limitations
* Accelerators!!