

CSE P 552

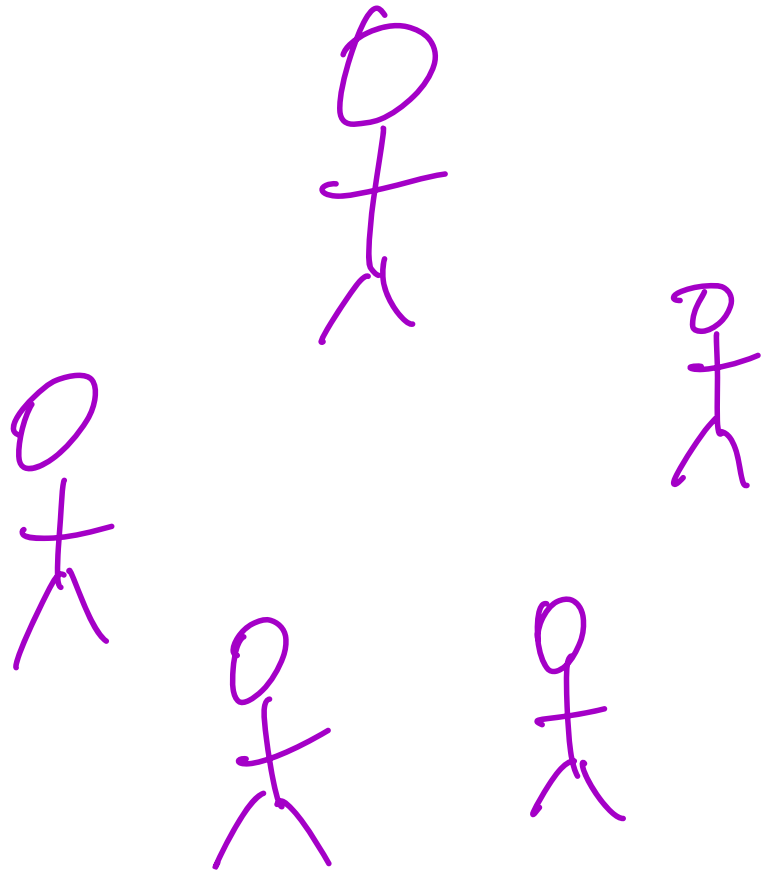
Distributed Systems

- applied vs theory

Muddy foreheads

n students

1 teacher



Muddy Distributed Systems

- local knowledge, global behavior
- lack of communication = lossy network
- topology of who can see what
- only missing knowledge is yourself
 - opposite of D.S.
- unreliable kindergarten — break protocol
- history is important — assuming good intent?

What is a distributed system?

- multiple machines, working together on some problem
- many isolated customers
- Some kind shared state
- give the appearance of only one machine
- Communication — lossy (synchronization)
- machines can fail
- Centralized controller?

Why distributed systems?

- more efficient, faster
- More reliable
- "scalable" horizontal = increase # of machines
 - vertical scaling = get bigger machine
- Want to share state
- independence of subsystems
- put data closer to users

- concurrency
- asynchronous
- partial failure

- failure is normal

Fault Model

- design protocols that are fault tolerant
- fault model = set of faults we plan to tolerate
 - usually not all possible faults
- Byzantine fault = malicious nodes
 - node not following the protocol

Our fault model

- Standard SS2 fault model
- Asynchronous network w/ fail stop nodes
- Network :
 - asynchronous (delay)
 - messages can be dropped, duplicated, reordered
- nodes: can fail by stopping

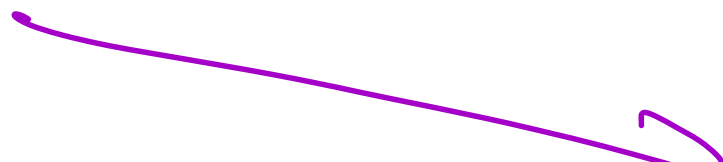
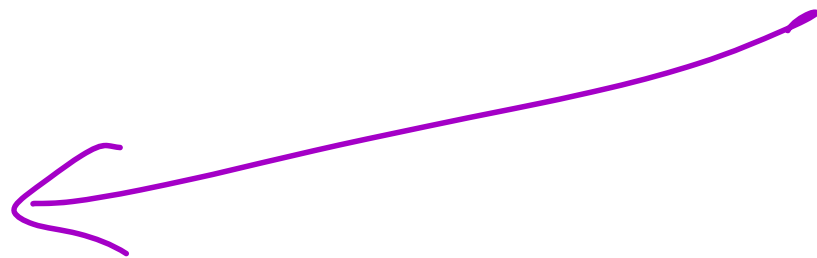
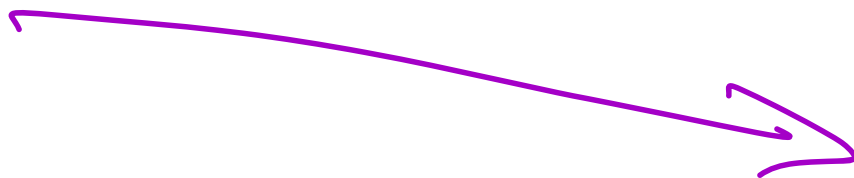
What can nodes know in SFM?

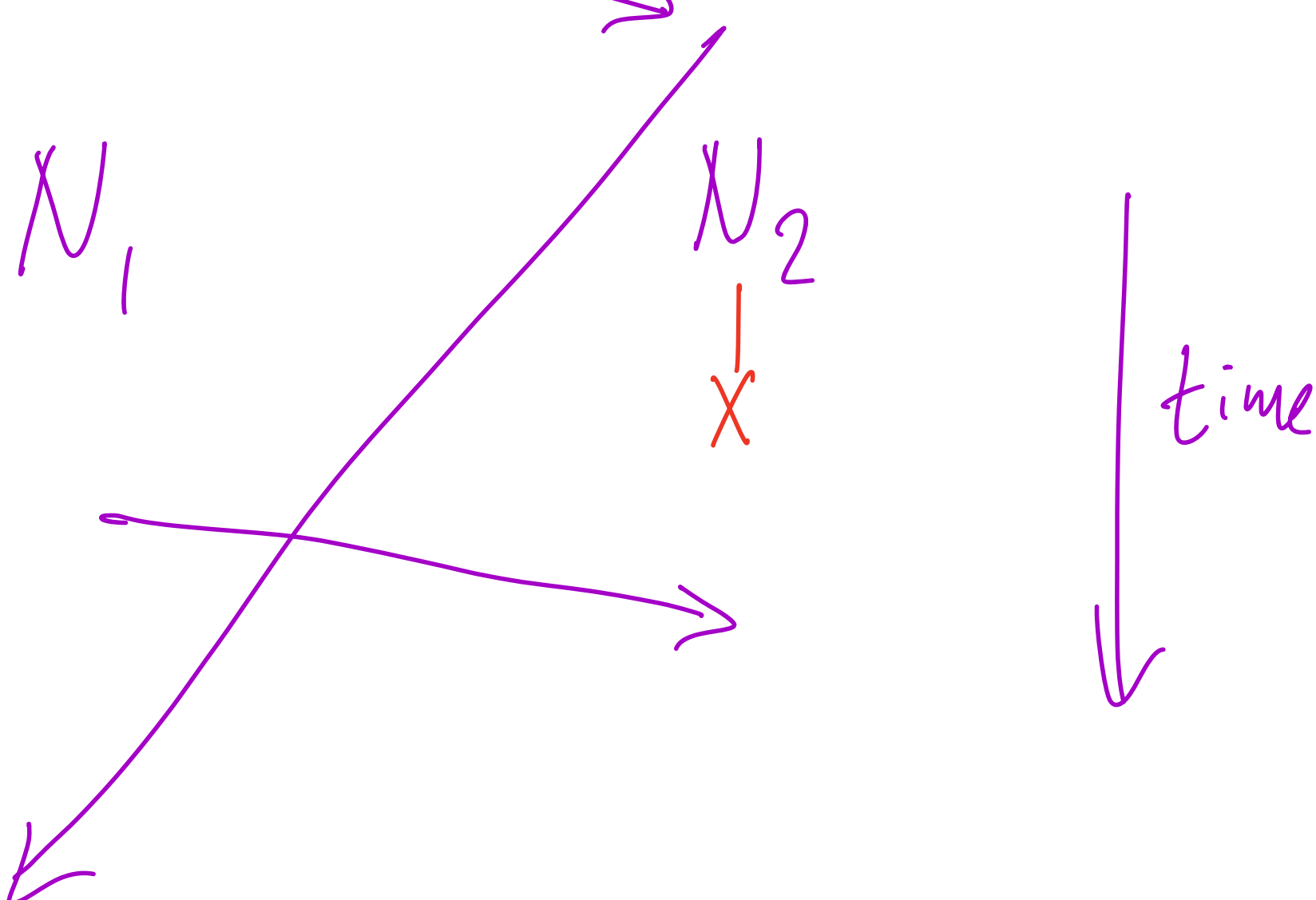
- they received a message (it was sent)
- other nodes are following the protocol
- is it possible to know if another node is failed or not?
- no

N_1

N_2

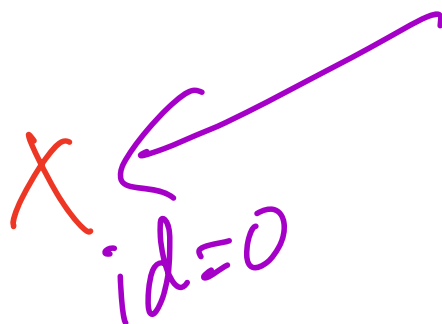
I





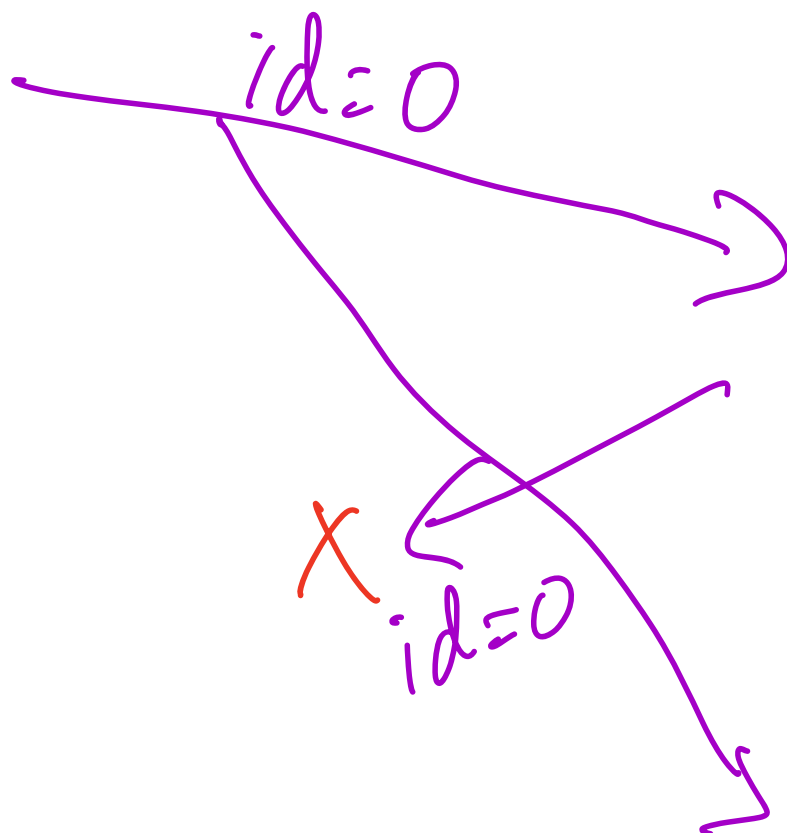
N_1

N_2



N_1

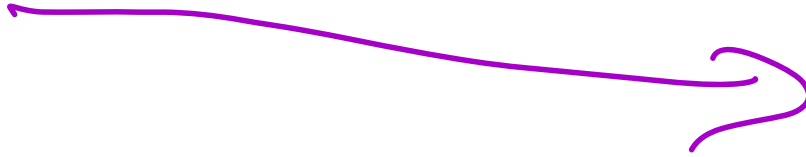
N_2



N_1

N_2

X



Networks vs Distributed Systems

- network delivers messages
- D.S. has our machines

Lab 1: Remote Procedure Call

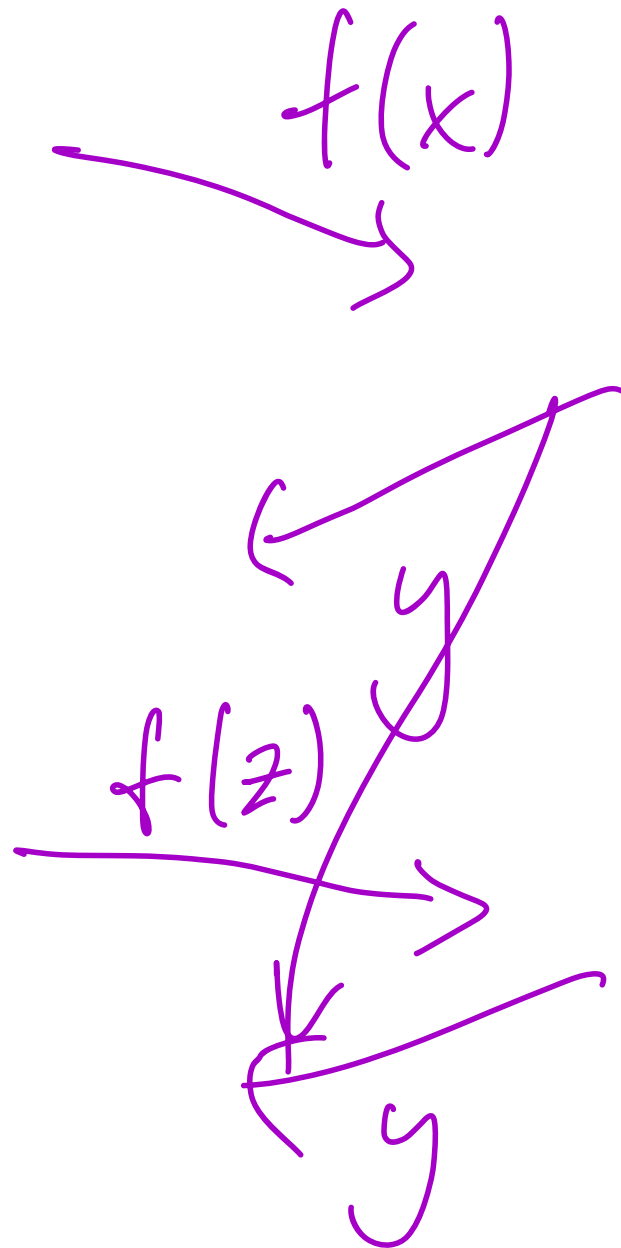
- RPC, R

- TCP solve all our problems?

- NO

Client

Server



Naive RPC

Naive RPC w/ identifiers

At most once RPC

- server remembers

$\text{Map} \left\langle \begin{array}{l} (\text{client}, \text{id}) \\ \text{resp} \end{array} \right\rangle$

At least once RPC

- client retry

Exactly once RPC

C

S

$f(x), 0$

$y, 0$
 x

$f(x), 0$

$y, 0$

Optimized ED RPC

old: $\text{Map} \langle (\text{client}, \text{id}), \text{resp} \rangle$

new: $\text{Map} \langle \text{client}, (\text{id}, \text{resp}) \rangle$

— incoming id stored id

=

retransmit resp

<

ignore

>

execute request