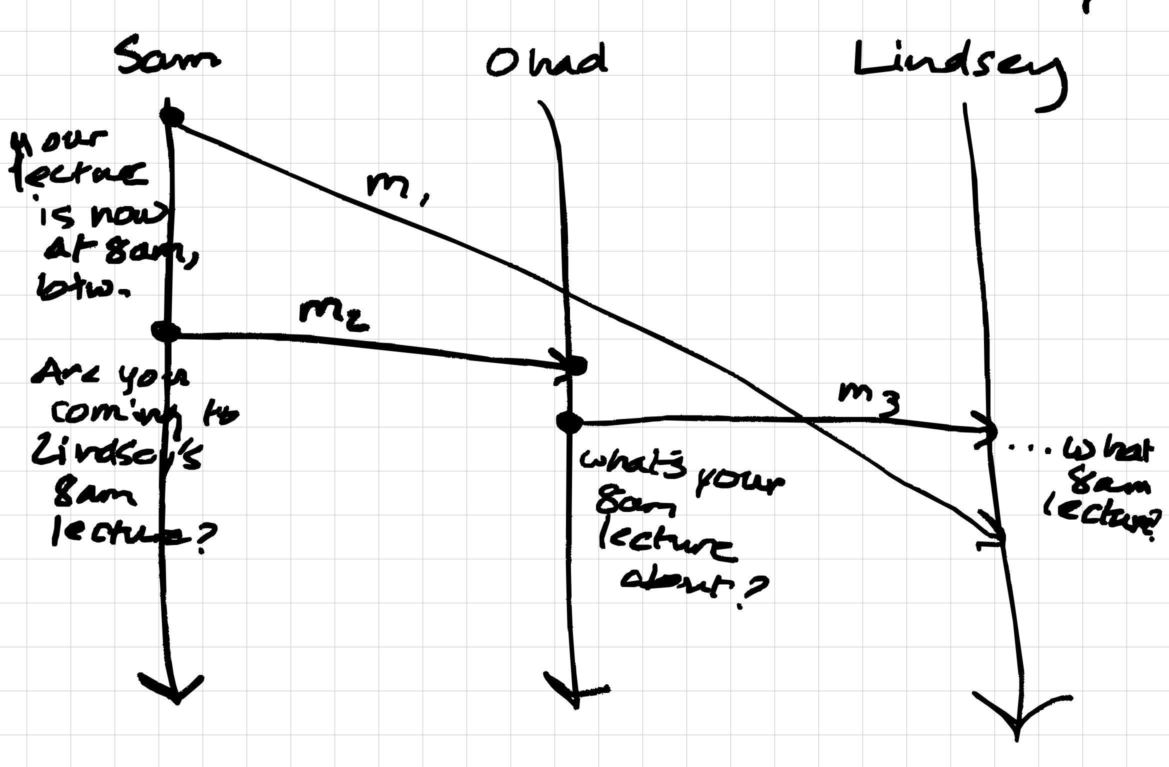
A few ideas from distributed systems for PL folk

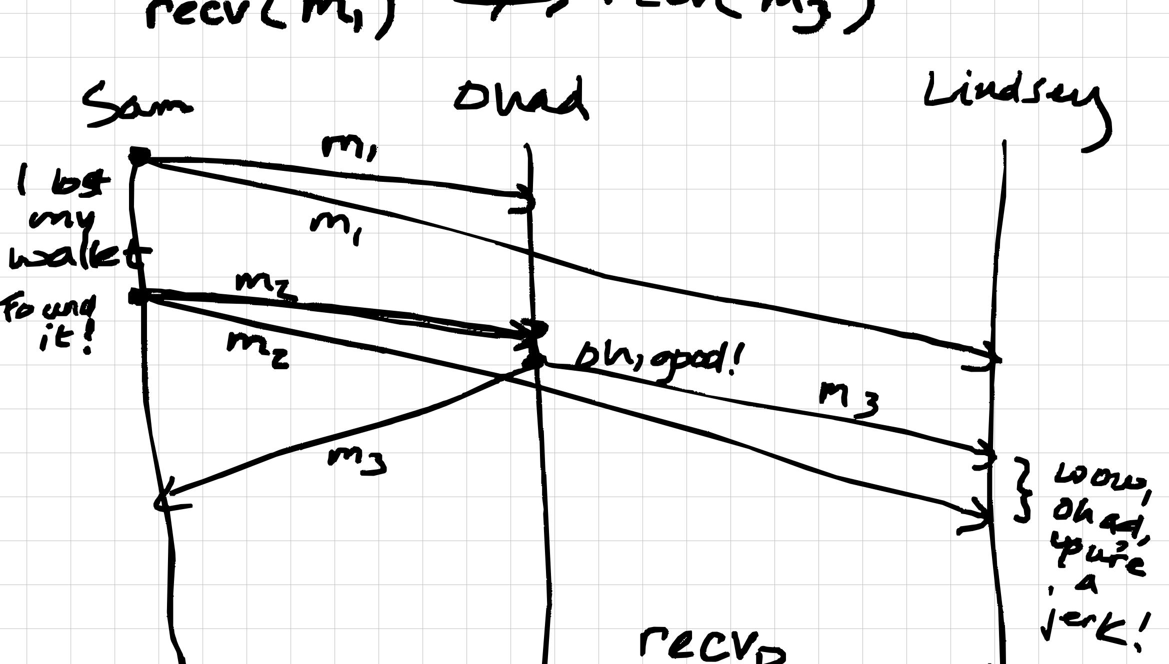
Lindsey Kuper (UC Santa Cruz)

lecture 2 agenda (time permitting):

→ causal message delivery
→ vector clocks
→ a causal broadcast protocol
- safety & liveness } these'll probably have
- fault models } to wait until Friday



$send(m_1) \rightarrow send(m_3)$,
but
$recv(m_1) \not\rightarrow recv(m_3)$

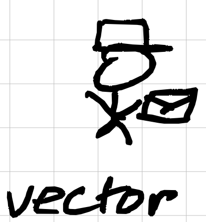


$send(m_2) \rightarrow send(m_3)$,
but
$recv_{Lindsey}(m_2) \not\rightarrow recv_{Lindsey}(m_3)$

causal message delivery:

Given messages m, m' in an execution,
if $Send(m) \rightarrow Send(m')$
then $\forall P$ such that
P delivers both m and m',
$deliver_P(m) \rightarrow deliver_P(m')$.

what's "delivery"?
- sending is something you do
- receiving is something that happens to you.
- delivering is something you can choose to do with a message you receive.



vector clocks!

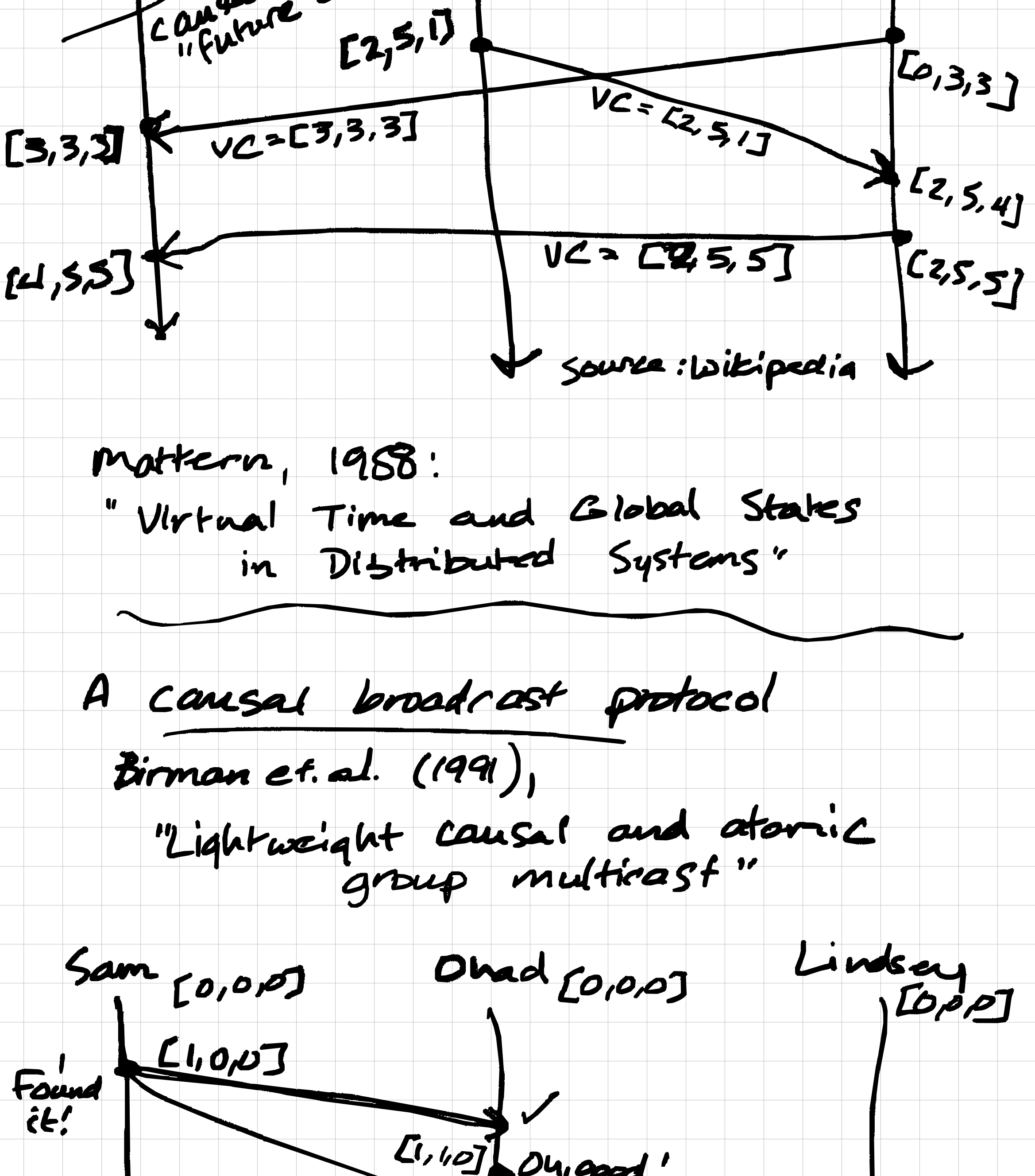
$e_1 \rightarrow e_2 \iff LC(e_1) < LC(e_2)$

The vector clock algorithm:
- Every process maintains a vector of natural numbers, all initially 0, one entry for each process. [0, 0, 0]
- On every event, a process increments its own entry in its vector.
- when sending a message, a process includes its current vector (after incrementing!)
- when receiving a message, a process updates its vector to the pointwise maximum of the received vector and its own vector. (and increment its own entry, too!)

max([2, 0, 3], [1, 3, 4])
= [2, 3, 4]

$e_1 \rightarrow e_2 \iff vc(e_1) < vc(e_2)$
↑

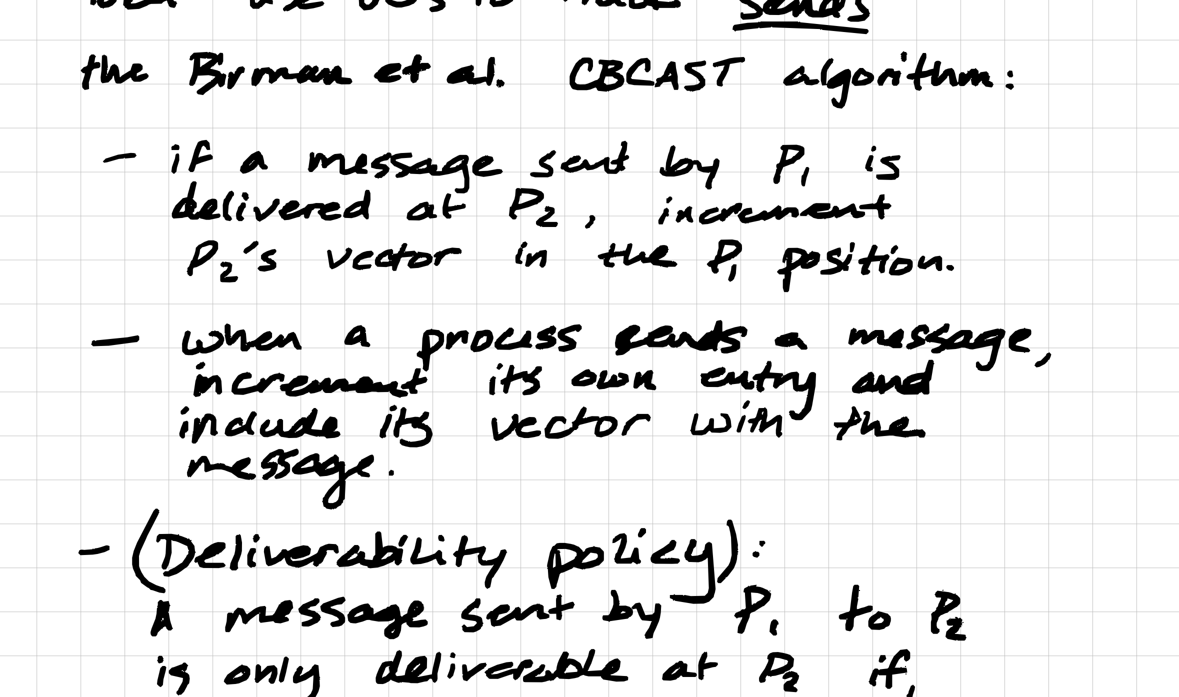
$vc(e_1) < vc(e_2) \stackrel{def}{=}$
for all indices i,
$vc(e_1)_i \le vc(e_2)_i$
and
$vc(e_1) \ne vc(e_2)$.



Mattern, 1988:
"Virtual Time and Global States in Distributed Systems"

A causal broadcast protocol

Birman et. al. (1991),
"Lightweight causal and atomic group multicast"



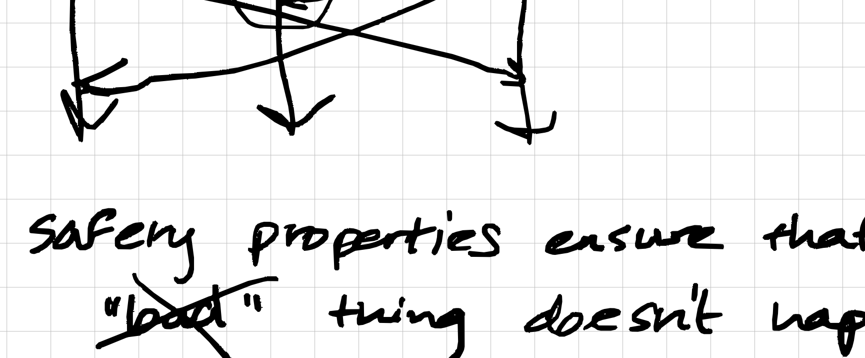
idea: use VCs to track sends

the Birman et al. CBCAST algorithm:

- if a message sent by P_1 is delivered at P_2, increment P_2's vector in the P_1 position.
- when a process sends a message, increment its own entry and include its vector with the message.
- (Deliverability policy):
A message sent by P_1 to P_2 is only deliverable at P_2 if,
let VC_m be the message's VC.

$VC_m[P_1] = VC_{P_2}[P_1] + 1$
and
$VC_m[P_K] \le VC_{P_2}[P_K]$.
for all $K \ne 1$

} deliverability condition!



safety properties ensure that a ~~"bad"~~ thing doesn't happen.

Liveness properties ensure that a ~~"good"~~ thing (eventually) happens.