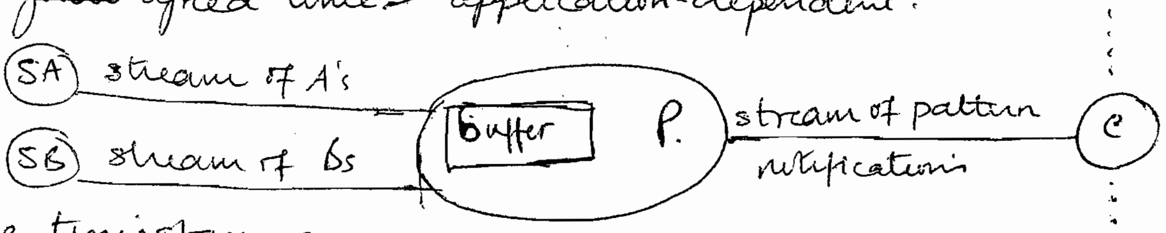
Papers 8 and 12 Distributed Systems.
on Intro, Time, Comm^n. JMB

2003 p8q4

a) Problem: absence of alarm message could mean:
- all's well
- faulty alarm service } alarm might or might not have occurred
- faulty communications }

Solution: set up a heartbeat protocol. Rate depends on "paranoia" le
4 Service needs policy on what to do if neither heartbeat nor event arriv
in the ~~given~~ agreed time - application-dependent.

b)



Message timestamps could be:-
- local clock value
- locally determined interval which contains UCT, taking into account all possible (atmospheric, network, software) delays in receiving a UTC value.

Decision depends on requirement for "real time" semantics. A naive total order can be based on single local clock value with a tie-breaker for equal values.

NOTE we DON'T KNOW the "real" ~~total~~ order of some messages.

Operators

A OR B: could send out stream as messages arrive or could delay to ensure correct (as far as possible) ordering.
could assume no faults and delay for max. network delay.
clients must be aware of precise semantics w.r.t. fault & time.

A AND B: unordered pairs.
need a consumption policy on which A's & B's to match (as in early active DB) & how to collect garbage. - EXPAND.
Again - may have communications & service failures which delay notifications.

A BEFORE B: ordered pairs
need a consumption policy on which A's & B's to match as above
— EXPAND
Failures may delay notifications as above.
In this case we sometimes CAN'T SAY whether A is before B.
Can do this naively with single timestamp and tie breaker but this is not real UTC ordering.
With intervals we can determine when A is before B and when we can't say. Client must be aware of precise semantics & may perhaps be able to request strong or weak ordering.