

# Modularity and Abstraction in Broadcast Specifications

CS262

March 10, 2008

# Administrivia

- Next week
  - Review on Monday (Ian)
  - Exam on Wednesday
- Next programming assignment
  - This week

# What's the point?

- Layering (and interfaces) are good
  - Define an abstraction (with an interface)
  - Build on top of that abstraction
  - Result in another abstraction
- Allows you to
  - Simplify
  - Control change

# Failure models

- Processes
  - Crash, send-omission, receive omission (benign)
  - Arbitrary
- Networks
  - omission failure (benign)

# Timing models

- Synchronous
  - An upper bound on process steps
  - A local clock with bounded drift
  - A (known) upper bound on message delay
- Asynchronous
  - None of the above

# Clock Models

- Monotonicity
  - A local clock value,  $v$ , never decreases or skips values
- Logical clocks
  - if  $e$  occurs at  $p$  and  $f$  occurs at  $q$ , then if  $e \rightarrow f$ ,  $C_p(e) < C_q(f)$
- $\epsilon$  Synchronized
  - Clock values differ by at most  $\epsilon$

# Broadcast Specifications

- Reliable Broadcast = Everyone (non-faulty) gets the message
- FIFO Broadcast = Reliable + order from a particular sender
- Causal Broadcast = FIFO + Causal order
- Atomic Broadcast = Everyone gets the message in the same order

# More Precisely

- **Reliable Broadcast**
  - **Validity:** If a correct process broadcasts  $m$ , then it eventually delivers  $m$
  - **Agreement:** If a correct process delivers  $m$ , then all correct processes deliver  $m$
  - **Integrity:** For any  $m$ , every correct process delivers  $m$  at most once, and only if  $m$  was broadcast

# FIFO

- FIFO Broadcast
  - Reliable Broadcast (Validity, Agreement, Integrity)
  - FIFO Order: if a process broadcasts  $m$  before  $m'$ , then no correct process delivers  $m'$  unless it has previously delivered  $m$

# Causal Broadcast

- Causal Broadcast = Reliable Broadcast +
  - Causal Order: If a broadcast of  $m$  causally precedes the broadcast of  $m'$ , then no correct process delivers  $m'$  before delivering  $m$
- Causal Broadcast = FIFO Broadcast +
  - Local Order: If a process broadcasts  $m$  and a process delivers  $m$  before broadcasting  $m'$ , then no correct process delivers  $m'$  unless it has delivered  $m$

# Why is it obvious

- That Causal Order implies FIFO and Local Order?

# Atomic Broadcast

- Atomic Broadcast = Reliable Broadcast +
  - Total order: If correct processes  $p$  and  $q$  both deliver messages  $m$  and  $m'$ , then  $p$  delivers  $m$  before  $m'$  iff  $q$  delivers  $m$  before  $m'$
- Note that this just says that everyone agrees, not that they are right

# So we get

- FIFO Atomic Broadcast
  - Atomic + FIFO
- Causal Atomic Broadcast
  - Atomic + Causal

# And then there is

- $\Delta$ -Timeliness
  - There is a known  $\Delta$  such that if  $m$  is broadcast at (real) time  $t$ , no correct process delivers  $m$  after  $t + \Delta$ .
  - There is a known  $\Delta$  such that no correct process delivers a message  $m$  after local time  $ts(m) + \Delta$  on  $p$ 's clock

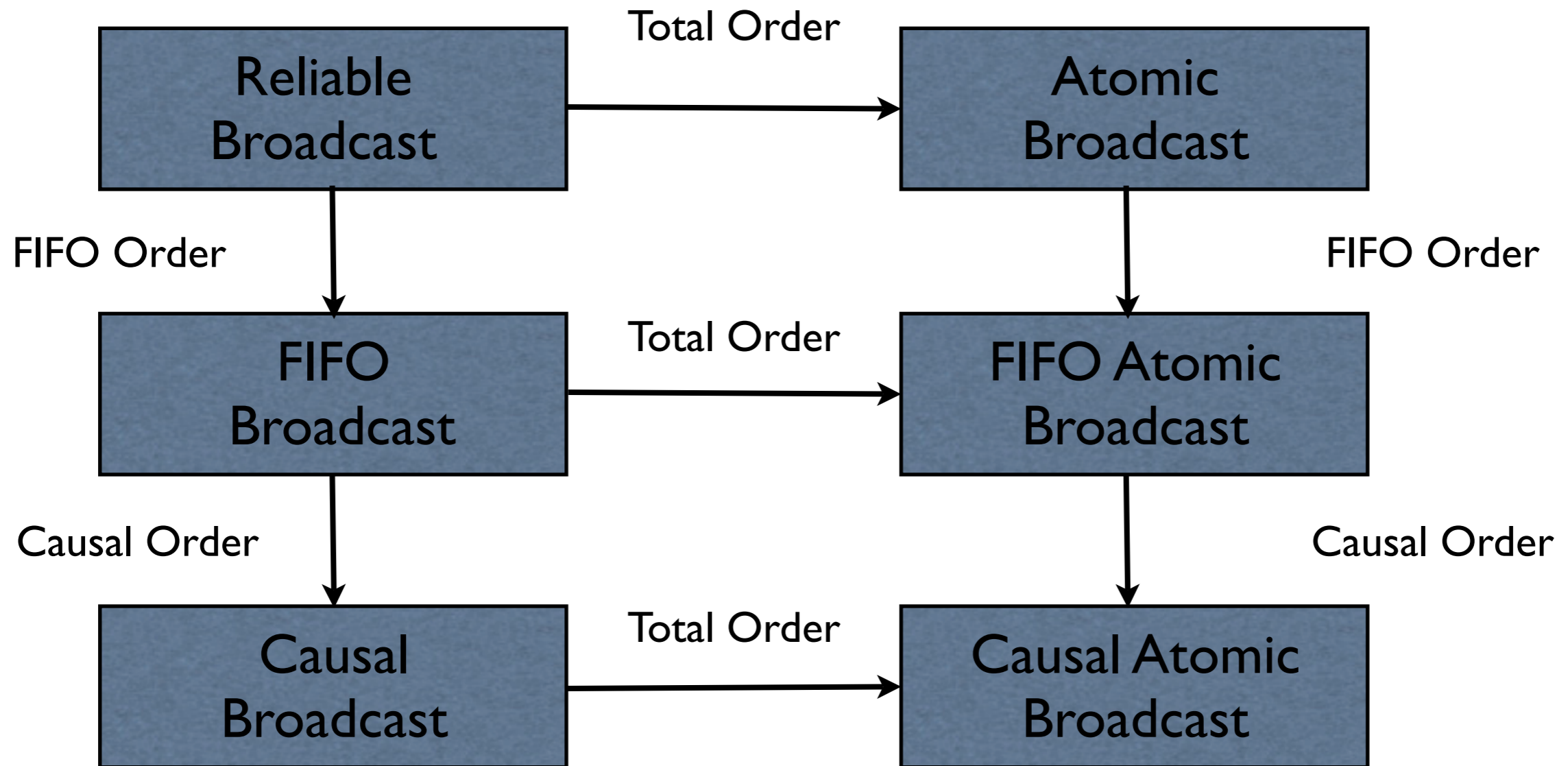
# And Finally

- **Uniformity: extending to faulty processes**
  - **Uniform agreement:** if any process delivers  $m$ , all correct process will eventually deliver  $m$
  - **Uniform Integrity:** for any  $m$ , every process delivers  $m$  at most once, and only if  $m$  was broadcast
  - **Uniform timeliness:** as you would expect

# Uniform Order

- Uniform FIFO: if  $m$  is broadcast by  $p$  before  $m'$ , then no process delivers  $m'$  before  $m$
- Uniform Local Order
- Uniform Causal Order
- Uniform Total Order

# So We Get...



# Which Gives Us

- An implementation strategy
- A proof strategy
- An algorithm strategy

# Real-life Side Note

- The right abstraction makes a technology
  - TCP/IP is the common layer
  - Everything converges on this layer
  - You can innovate below
  - You can innovate above
  - Don't try to change the common layer

# With Arbitrary Failures

- Agreement can be defined
- Validity can be defined
- Integrity can't be defined
  - What does it mean for a message to be sent?
- No reliable broadcast
- None of the others, either