

CS244B Midterm Review

Spring 2011

Administration

- Tuesday, May 3rd: Midterm!
- Project 2 Out Next Week
 - Review session next Friday

Midterm Information

- Tuesday, May 3rd, 7:30-8:45pm
 - Hewlett 201
 - Closed book. No notes.



Midterm Topics

- All of chapters 1 - 4
 - Chapter 1: “Overview”
 - Chapter 2: “Distributed Shared Memory”
 - Chapter 3: “Structuring of Dist. Systems”
 - Chapter 4: “Clocks: Real, Virtual, Logical”

Midterm Topics Cont'd

- Assigned Readings
 - Listed at end of each chapter as “Readings”
 - “References” are optional
- 6 in total
 - Hint: Give priority to ones with most focus in reader and lectures

Midterm Hints

- Look at prior midterms:
 - <https://www.stanford.edu/class/cs244b/exams/>
 - Format is very consistent year after year
- Look for the key points
 - Introductions and conclusions in readings have good summaries
 - Understand the bad ideas and the good ideas
 - E.g. Limitations of RPC and why stateful proxies?

Chapter 1

- 3 Views of Distributed Systems
 1. Object View
 2. Protocol View
 3. Shared State View

Chapter 1: Object View

- DS as a set of objects with well-defined interfaces
 - Enforces modularity
 - Uniform interface, regardless of actual implementation
 - Can exploit interface for sufficient consistency
 - E.g. What are the app-specific interface semantics?
- Forces focus on state

Chapter 1: Protocol View

- What are the syntax, semantics, and timing of packets?
- Focus on low-level protocols
- Interoperability
 - Can implement protocol in many ways, with different languages, etc and be compatible

Chapter 1: Shared State View

- Shared state is what makes a bunch of distinct computers and processes a distributed system
- What state is shared?
- How much consistency is sufficient?
 - Strict consistency often too expensive.
- Mazewar was an exercise in thinking about shared state and sufficient consistency

Chapter 2: Distributed Shared Memory

- DSM seems compelling because it maintains a familiar abstraction (shared memory)
- But there are some performance issues
 - Expense of consistency, false sharing, unpredictability (100ns vs milliseconds for memory access), fault tolerance, locking overheads, heterogeneity of nodes
 - Many solutions to these, but they break down the abstraction
 - Becomes less and less transparent

Chapter 3: Structuring Distributed Systems

- Message Abstraction
- File Abstraction
- Socket Abstraction
- DSM Abstraction
- OO RPC Abstraction
- Smart Proxies
- Stateful Proxies

Chapter 4: Clocks

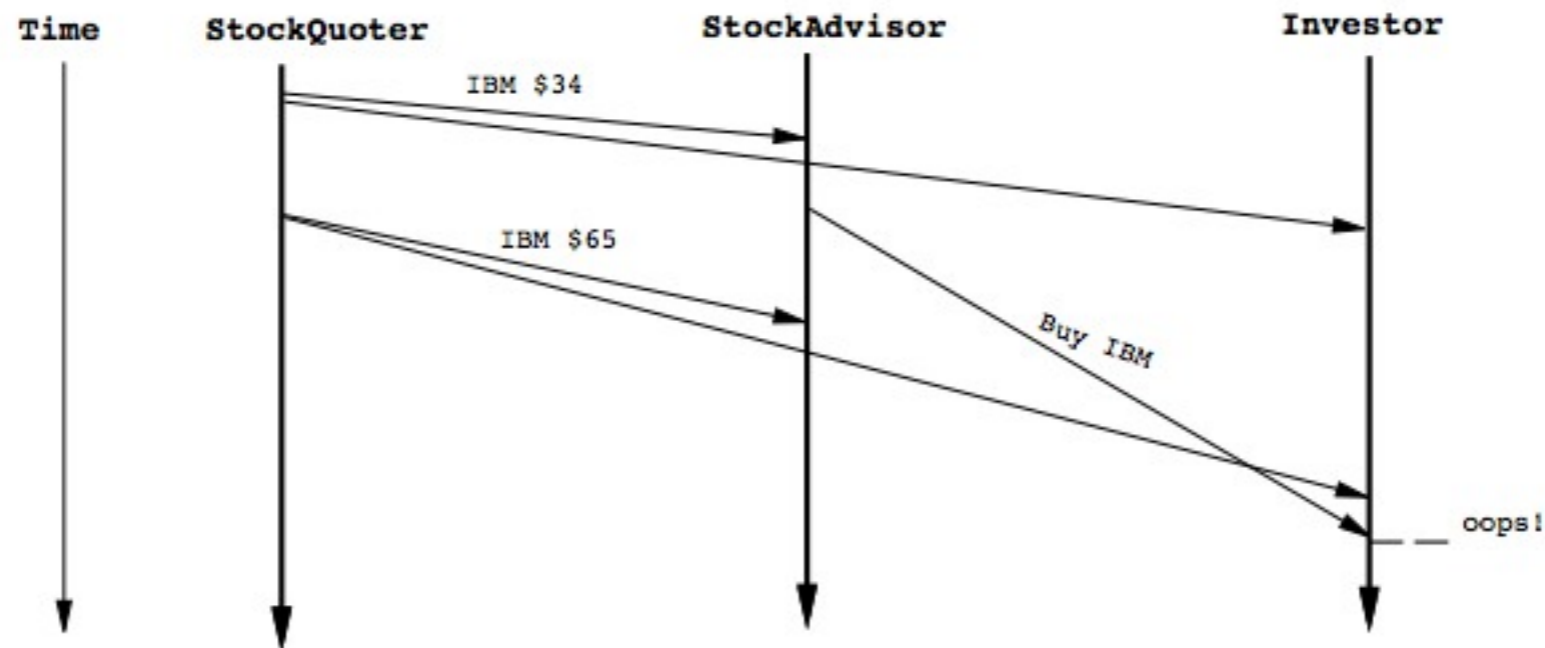
- Real Clocks
 - Need for reasonable clock synchronisation
 - NTP
 - Symmetric, Multicast, RPC modes
 - Hierarchical structure for scaling (strata)
 - Problem: How to manage the hierarchy
 - Cheriton's view: put time servers in the routers (already trust them anyhow)

Chapter 4: Clocks Cont'd

- Virtual Time
 - Monotonically increasing, but not at fixed rate
 - Used for simulation
 - E.g. Timewarp
 - Speculative execution, anti-messages to roll back, recurses
 - Neat idea, but bad cost/benefit trade-off
 - Limited improvement, very complex model

Chapter 4: Logical Clocks

- Used to sequence events
- E.g. version numbers
 - Avoid processing request if it is delayed and out of date
 - Stock trading example



Logical Clocks Cont'd

- CATOCS - Causally and Totally Ordered Comm. Support
 - Put ordering in the communication layer
 - Messages contain *potential* causal dependency information
 - Delay message delivery at receiver to adhere to ordering imposed by potential causality
- Problems:
 - “Can’t say for sure” - hidden communication channels, e.g. DBMSes
 - “Can’t say together” - cannot order entire groups of messages
 - “Can’t say the whole story” -
 - “Can’t say efficiently” - No efficiency gain, scalability concerns.
- Conclusion: Need to use state-level solutions to overcome limitations in CATOCS, which obviates CATOCS entirely

Example Question

- “Define false sharing and describe why it is difficult/expensive to avoid and to have in DSMs.”

Example Question

- “Describe why ‘exactly once’ semantics is difficult to guarantee even though we understand how to do duplicate suppression and retransmission to achieve this at the transport (TCP) level.”

Example Question

- Cheriton Describes a distributed system as “at best a necessary evil.”
 1. Explain what makes it evil, with a specific example.
 2. Explain what can make it necessary, with a specific example.

The End

Good Luck!