Principles of Reliable Distributed Systems

Lecture 1: Two Generals & Specifying Systems

Idit Keidar
Staff

• Lecturer: Idit Keidar
  – idish@ee
  – Appointment by email

• TA: Ittay Eyal
  – ittay@tx
  – Appointment by email
Material

• Textbooks:
  – Distributed Systems  2nd edition
    Sape Mullender (Editor), Addison Wesley
  – Distributed Computing; Fundamentals, Simulations and
    Advanced Topics
    Hagit Attiya and Jennifer Welch, McGraw Hill

• Research papers
  – See links on course web page

• Lecture slides
  – Do NOT cover all the material!
Grading and Requirements

• Two quizzes – 75%-100%
  – **Allowed material:** annotated lecture slides, your own notes and solved HW assignments
  – **No** books, articles, printed material

• Dry (or semi-dry) HW, top 5 of 6 – 25% MAGEN
  – **Average \( \geq 55 \) required** in order to take the final quiz
  – Will count in final grade only **if the difference from the average quiz score is < 30**
  – Good practice for quizzes!
    • HW will be harder than quizzes
  – Submit individually (no pairs)
    • You may discuss with others
    • Write the solutions by yourself, in your own words
Quiz Dates

1. Tuesday, 1/5/2012, 10:30
2. Tuesday, 26/6/2012, 10:30
Each will be 1.5 hour long

MOED BET will have two parts, 1.5 hour long each
Prerequisites

• You need background in algorithms and operating systems
• You need to be comfortable with formal definitions and proofs
• If you do not have the prerequisites (or CS equivalents), you need explicit permission from me to take the course
Interpreting the Course Name 1/3

• Principles
  – This means *theory*
  – Formal models, definitions, proofs

• of … Systems
  – Algorithms that people actually implement
    • In systems research as well as in industry
  – Study practical measures (not asymptotic) and considerations
Course Name 2/3 : Distributed

• Multiple computers
  – Each having CPU, local memory, stable storage (disk), I/O to the environment

• Interconnected (*networked system*)

• Shared state
  – Correct operation of the system described in terms of *global* properties (invariants)
  – Maintaining these requires coordination
  – We will focus on this coordination
Course Name 3/3 : Reliable

- Fault-tolerance
- Techniques for tolerating different types of failures:
  - Process crash, machine crash
  - Network communication delay, message loss
  - Software bugs
  - Malicious intrusions
Birdseye View of Course Syllabus

• Specifying distributed systems
• Broadcast services and replication
• Peer-to-peer computing
• The consensus problem in different models
  – Using consensus for state-machine replication
• Shared memory
• Storage-based systems
Today’s Agenda

• The two generals problem [Gray 78]
  – Motivation: Atomic Commit of distributed transactions
  – Reading material: Chapter 2 of Mullender
  – Variants:
    • Coordinated attack
    • Weak coordinated attack

• Specifying systems
  – Safety and liveness properties
The Two Generals Problem

Jim Gray, 1978
Coordinated Attack: The Setting

Let’s attack
The Coordinated Attack Problem

• Each general votes “ready” (yes) or “not ready” (no) to attack
  – This is the input

• And outputs a decision
  – To attack or not to attack

• Requirements:
  – Both generals must decide the same
  – If both are not ready to attack they must not attack
  – If both are ready to attack then they must attack
The Coordinated Attack Problem

• Model: reliable generals, lossy communication
  – Any message sent from one general to the other may be lost

• Motivation: atomic transaction commit in distributed databases [Gray 78]

• Can you solve it?
A Simple Solution

• Each general sends his vote ("yes" or "no") to the other general
• If both say yes, they attack
• Otherwise they do not attack
• This is a 2-phase commit protocol
  – Symmetric version

Problems?
Oops

• This solution does not work 😞
• Nor does any other solution 😞
• A coordinated attack is impossible in the presence of message loss!
• To prove it, we will need to be a lot more formal about
  – What is a solution
  – The problem specification
  – The model
A Solution is a Protocol

• Each process (general) runs a protocol
• Starts from some initial state
  – In our case, “ready” or “not ready”
• Then makes state transitions
  – Associated with actions
  – We consider deterministic protocols
  – state transition function :
    states X messages \rightarrow states X messages
• In every step, a process can
  – Receive pending messages
  – Change its local state
  – Send a message to the other general

defined by state transition function
Example: Counter Protocol

- One state variable: count
- Initial state: count = 0
- Action: Inc_p - process p performing increment
- Transitions:
  - inc_p - count++
Runs / Executions

- A **global state (configuration)** is a vector of the states of all system components
- A **run (execution)** of a protocol is an alternating sequence of global states and events (occurrences of actions)
  - Where all steps follow the protocol’s transition function
- Example run of the counter:
  
  \[0, \text{inc}_A, 1, \text{inc}_B, 2, \text{inc}_B, 3, \text{inc}_B, 4\]
Simple Solution Protocol

• Each general’s state consists of variables:
  – input - 1 for “ready” or 0 for “not ready”
  – sent - 1 if sent message to other general, 0 otherwise
  – decision - 1 “attack” or 0 “no attack” or ⊥ if not decided yet

• Initial states: (1,0,⊥) or (0,0,⊥)

• Protocol state transitions (steps):
  – if !sent then
    send input; sent ← 1
  – upon receive message,
    if message = 1 and input = 1 then
      decide “attack”; decision ← 1
    else
      decide “not attack”; decision ← 0

© Idit Keidar, Principles of Reliable Distributed Systems, Technion EE
Example Run of Simple Solution

Step 1: general A sends “yes”

(1,0,⊥),(0,0,⊥), send_A(1), (1,1,⊥),(0,0,⊥),
send_B(0), (1,1,⊥),(0,1,⊥),
receive_B(1), decide_B(“no attack”), (1,1,⊥),(0,1,0),
receive_A(0), decide_A(“no attack”), (1,1,0),(0,1,0),

B decides not to attack

Step 2: general B sends “no”

A decides not to attack

Is it correct?
Example Run 2 of Simple Solution

$(1,0,\perp),(0,0,\perp)$, $\text{send}_A(1), (1,1,\perp),(0,0,\perp)$,
$\text{send}_B(0), (1,1,\perp),(0,1,\perp)$,
$\text{receive}_B(1), \text{decide}_B(“\text{no attack}”), (1,1,\perp),(0,1,0)$,
message sent by $B$ is lost

A never decides

Is it correct?
Runs are Sequential

• A run is a sequence (by definition)
• Events that occur simultaneously at different processes are ordered in some arbitrary way
  – send_A(“yes”), send_B(“no”)
  or
  – send_B(“no”), send_A(“yes”)
Specifying Requirements

• Requirements are specified as properties of runs
• All runs of the protocol must satisfy them
  – All runs allowed by the model, that is

• In the coordinated attack, both generals must decide in all runs
• The model allows for runs where any number of messages are lost
Coordinated Attack Specification

• **Agreement:** If both generals decide, they decide the same

• **Termination:** Every general eventually decides

• **Validity:**
  – If both inputs are “not ready” the only valid decision is “no attack”
  – If both inputs are “ready” then the only valid decision is “attack”

• We will see later on that this is a special case (with 2 processes only) of Uniform Consensus
Atomic Commit Problem

• **Agreement:** If both generals decide, they decide the same

• **Termination:** Every general eventually decides

• **Validity:**
  – If some input is “not ready” the only valid decision is “no attack”
  – If both inputs are “ready” then the only valid decision is “attack”

• Defined more generally for any number of generals (databases)
Coordinated Attack is Impossible

• Recall: any number of messengers can be captured (message loss)

• To show that no solution exists
  – It is enough to show that no solution works in all runs where all messages are lost
  – Why?
Impossibility Intuition

• If all messages are lost, general A “cannot tell the difference” between the case that B votes “yes” and the case that B votes “no”

• This intuition is captured by indistinguishability
Indistinguishability

• Two runs are indistinguishable to process A if A takes the same steps in both runs

• If a protocol is deterministic, runs in which A has the same input and receives the same messages are indistinguishable to A
Impossibility Proof

• **Lemma 1**: No protocol solves the coordinated attack problem in all runs where no messages are delivered

• Formal proof on board

• Perhaps the problem is with the specification?
  – How can we weaken it?
Weak Coordinated Attack

• Both generals must decide the same
  – Either to attack or not to attack
• If both are not ready to attack they must not attack
• If both are ready to attack and no messages are lost then they must attack

Now, is it possible?
Notes on Weak Definition

- We did not change the model
- It is still possible for any number of messages to be lost
  - But then the protocol is not required to decide to attack
- We did not relax agreement, only validity
  - The generals still have to agree no matter how many messages are lost
Weak Coordinated Attack

• **Agreement:** If both generals decide, they decide the same

• **Termination:** Every general eventually decides

• **Weak (Conditional) Validity:**
  – If both inputs are “not ready” the decision is “no attack”
  – If both inputs are “ready” and no messages are lost then the decision is “attack”
Impossibility Revisited

• Lemma 1 no longer holds
  – Now “not to attack” is a valid decision in all runs where all messages are lost

Instead, we’ll use:

• **Lemma 1’**: There is no protocol that solves the weak coordinated attack problem and sends no messages before deciding in runs where both generals are “ready”

• **Homework**: formally prove Lemma 1’
  – **Hint**: similar to Lemma 1
Weak Coordinated Attack is Impossible

• Proof by induction on the number of messages sent before decision in runs where both generals are “ready”
  – Base case?
  – Proof of inductive step on the board
Safety and Liveness Properties
Some Definitions

- **Sequence**: $a_1a_2a_3a_4a_5,...$
- **Prefixes** of this sequence: $a_1$, $a_1a_2$, $a_1a_2a_3$, etc.
- A **predicate** is a formula evaluated to a boolean value (true or false)
  - E.g., $x > 10$
- Recall: a **run** is an alternating sequence of global states and events
- A **trace** is the sequence of events occurring in a run
Example: Chat System

Client A

Chat Application

send

receive

chat

view

Client B

Chat Application

send

receive

chat

view

Network
Predicate Examples

• Evaluate the predicate “if m is delivered it was previously sent” over the traces:

1. $\text{chat}_A(\text{"hello"}) \ \text{send}_A(\text{"hello"}) \ \text{receive}_B(\text{"hello"})$
   $\ \text{view}_B(\text{"hello"})$
   True or false?

2. $\text{chat}_A(\text{"hello"}) \ \text{chat}_B(\text{"good-bye"}) \ \text{send}_A(\text{"hello"})$
   $\ \text{receive}_B(\text{"hello"}) \ \text{receive}_A(\text{"good-bye"})$
   True or false?
Specifying Properties

• Concurrent (distributed / parallel) systems can be specified using properties
• A property is a predicate evaluated over a run or trace of the concurrent system
• Example property: every message that is received was previously sent
• Not a property: the average number of messages sent in a run is 34
• Can you think of properties of the counter?
Safety Properties

• A safety property is of the form *nothing bad happens* (that is, all states are safe)

• Examples:
  – The number of processes in a critical section is always less than 2
  – Every delivered message was previously sent
  – If both generals are not ready, the only valid decision is not to attack
Liveness Properties

• A liveness property is of the form something good happens
  – An interesting state is eventually achieved

• Examples:
  – A process that wishes to enter the critical section eventually does so
  – The value of $p$ grows without bound
  – Every general eventually decides
More Formally

• A safety property is prefix-closed:
  – If it holds in a run, it holds in every prefix
  – You can’t “fix” it after it’s “broken”

• Every run can be extended to satisfy a given liveness property:
  – No matter how “broken”, you can always “fix” it
Safety & Liveness are Complete

• Every property is one of the following:
  – A safety property
  – A liveness property
  – Equivalent to the conjunction of a safety property and a liveness property

• E.g., Critical Section
  – A conjunction of Mutual Exclusion and Progress
Timing Properties

- If a message is sent, then it arrives within five minutes
- Safety or liveness?
- Can be expressed only in a timed model
Summary

• To reason about a system, we need to formally specify
  – The model
  – The requirements

• Some requirements will be impossible to satisfy in some models
  – E.g., weak coordinated attack with message loss

• To realize a solution, we would need to either
  – Weaken the requirements OR
  – Strengthen the model
  – Homework: Suggestions for coordinated attack?