MPRI – Course on Concurrency

Lecture 16

The need for randomization: examples in distributed computing and in security

Kostas Chatzikokolakis
LIX, Ecole Polytechnique
kostas@lix.polytechnique.fr
www.lix.polytechnique.fr/~kostas

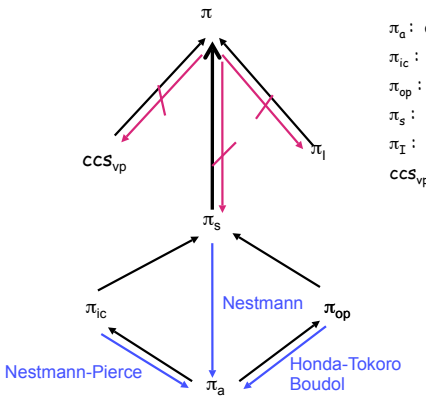
Page of the course:
http://mpri.master.univ-paris7.fr/C-2-3.html

Plan of the lecture

- The power of randomization
  - Some problems in distributed systems that can only be solved with the use of randomization
    - Dining Philosophers
- Randomized protocols for security (in particular anonymity)
  - The dining cryptographers
    - Correctness of the protocol
    - Anonymity analysis
  - Crowds (a protocol for anonymous web surfing)

The π-calculus hierarchy

(partly discussed in previous lecture)



π_a : asynchronous π
π_{ic} : asynchronous π + input-guarded choice
π_{op} : asynchronous π + output prefix
π_s : asynchronous π + separate choice
π_I : π with internal mobility (Sangiorgi)
ccs_{vp} : value-passing ccs

→ : Language inclusion
→ : Encoding
↛ : Non-encoding

The separation between π and π_s

(seen in previous lecture)

This separation result is based on the fact that it is not possible to solve the symmetric leader election problem in π_s, while it is possible in π

Leader Election Problem (LEP): All the nodes of a distributed system must agree on who is the leader. This means that in every possible computation, all the nodes must eventually output the name of the leader on a special channel out

No deadlock

No livelock

No conflict (only one leader must be elected, every process outputs its name and only its name)

The power of Randomization

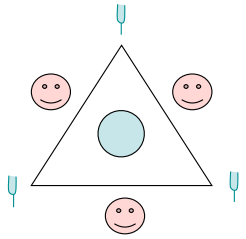
Some problems in distributed systems can only be solved with the use of randomization

- This is the case of the symmetric leader election problem in an asynchronous network

- We will see an analogous problem: the dining philosophers

(1) The dining philosophers

- Each philosopher needs exactly two forks
- Each fork is shared by exactly two philosophers
- A philosopher can access only one fork at the time



Intended properties of solution

- **Deadlock freedom** (aka **progress**): if there is a hungry philosopher, a philosopher will eventually eat
- **Starvation freedom**: every hungry philosopher will eventually eat (but we won't consider this property here)
- **Robustness wrt a large class of adversaries**: Adversaries decide who does the next move (schedulers)
- **Fully distributed**: no centralized control or memory
- **Symmetric**:
  - All philosophers run the same code and are in the same initial state
  - The same holds for the forks

Non-existence of a "deterministic" solution

- Lehman and Rabin have shown that there does not exist a "deterministic" (i.e. non-probabilistic) solution to the dining philosophers, satisfying all properties listed in previous slide.

- The proof proceeds by proving that for every possible program we can define an adversary (scheduler) which preserves the initial symmetry

- **Note**: Francez and Rodeh did propose a "deterministic" solution using CSP. The solution to this apparent contradiction is that CSP cannot be implemented in a fully distributed way

The algorithm of Lehmann and Rabin

1. Think
2. randomly choose fork in {left,right} %commit
3. if taken(fork) then goto 3
4. else take(fork)
5. if taken(other(fork)) then {release(fork); goto 2}
6. else take(other(fork))
7. eat
8. release(other(fork))
9. release(fork)
10. goto 1

Correctness of the algorithm of Lehmann and Rabin

- **Theorem:** for every **fair** adversary, if a philosopher becomes hungry, then a philosopher (not necessarily the same) will eventually eat with probability 1.
- **Question:** why the fairness requirement? Can we write a variant of the algorithm which does not require fairness?

Anonymity

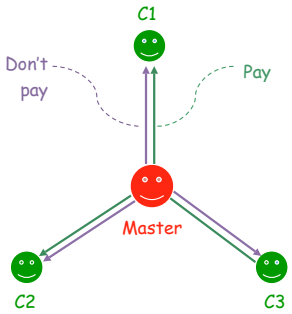
- Hide the identity of a user performing a given action
- The action itself might be revealed
- Many applications
  - Anonymous web-surfing (Crowds)
  - Elections
  - Donations

The dining cryptographers

- A simple anonymity problem
- Introduced by Chaum in 1988
- Chaum proposed a solution satisfying the so-called "strong anonymity"
- Extensions of the protocol are used in practice

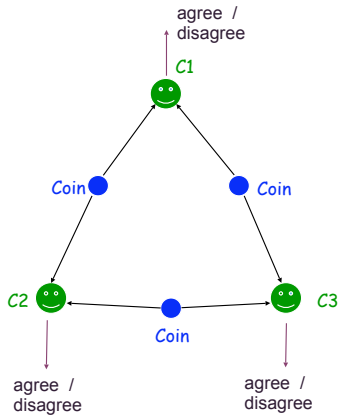
The problem

- Three cryptographers share a meal with a master
- In the end the master decides who pays
- It can be himself, or a cryptographer
- The master informs each cryptographer individually
- The cryptographers want to find out if
  - one of them pays, or
  - it is the master who pays
- **Anonymity requirement**: the identity of the paying cryptographer (if any) should not be revealed

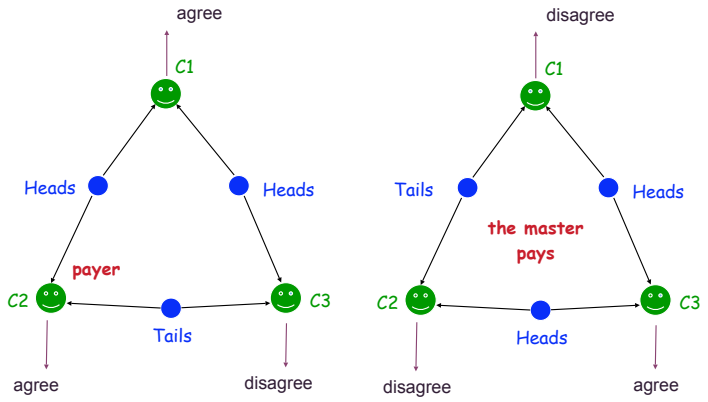


The protocol

- Each pair of adjacent cryptographers flips a coin
- Each cryptographer has access only to its adjacent coins
- Each cryptographer looks at the coins and declares **agree** if the coins have the same value and **disagree** otherwise
- If a cryptographer is the **payer** he will say the **opposite**
- Consider the **number of disagrees**:
  - **odd**: a cryptographer is paying
  - **even**: the master is paying

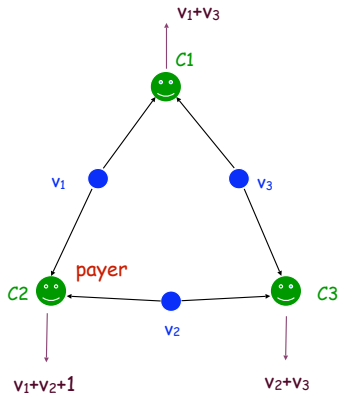


Examples



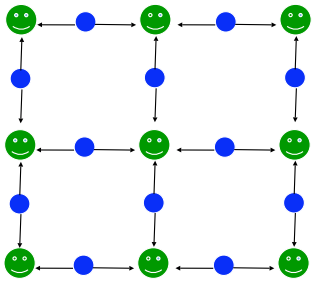
Correctness of the protocol

- Let $v_i \in \{0,1\}$ be the value of coin i
- Each cryptographer announces v_i+v_{i+1} where + is the sum modulo 2:
  - 0 means agree
  - 1 means disagree
- The payer announces $v_i+v_{i+1}+1$
- The total sum is
  - $(v_1+v_2)+v_2+v_3+v_3+v_1 = 0$ if the master pays
  - $v_1+v_2+v_2+v_3+v_3+v_1+1 = 1$ if a cryptographer pays



Correctness of the protocol

- The protocol is correct for any (connected) network graph
- The key idea is that all coins are added twice, so the cancel out
- Only the extra 1 added by the payer (if there is a payer) remains
- Question: can we extend this protocol to transfer actual data?



Anonymity of the protocol

- How can we define the notion of anonymity?
- First we have to answer these questions:
  - What type of anonymity?
    - Strong anonymity: all cryptographers appear equally likely to be the payer
    - Weaker notions
  - With respect to whom?
    - An external observer
    - One of the cryptographers

Anonymity of the protocol

- For an external observer the only visible actions are sequences of agree/disagree (daa, ada, aad, ...)
- For strong anonymity we would like different payers to produce these actions with equal probability

$$p(daa \mid C1 \text{ pays}) = p(daa \mid C2 \text{ pays})$$
$$p(daa \mid C1 \text{ pays}) = p(daa \mid C3 \text{ pays})$$
$$\ldots$$

- This is equivalent to requiring that

$$p(C1 \text{ pays}) = p(C1 \text{ pays} \mid daa)$$

- Exercise: prove it

Anonymity of the protocol

- Assuming fair coins, we compute these probabilities

	daa	ada	aad	ddd
c1	1/4	1/4	1/4	1/4
c2	1/4	1/4	1/4	1/4
c3	1/4	1/4	1/4	1/4

- Strong anonymity is satisfied

Anonymity of the protocol

- If the coins are unfair this is no longer true
- For example, if p(heads) = 0.7

	daa	ada	aad	ddd
c1	0.37	0.21	0.21	0.21
c2	0.21	0.37	0.21	0.21
c3	0.21	0.21	0.37	0.21

- Now if we see daa, we know that c1 is more likely to be the payer

Anonymity of the protocol

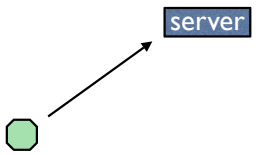
- Even if we don't know the fact that the coins are unfair, we could find out using statistical analysis
- Exercise: suppose we see almost all the time one of the following announcements

  ada aad ddd

  - what can we infer about the coins?
  - then can we find the payer?

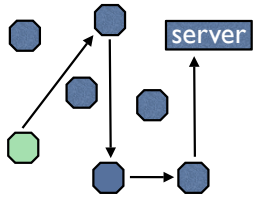
Crowds

- A protocol for anonymous web surfing
- goal: send a request from a user (initiator) to a web serer
- problem: sending the message directly reveals the user's identity
- more efficient that the dining cryptographers: involves only a small fraction of the users in each execution



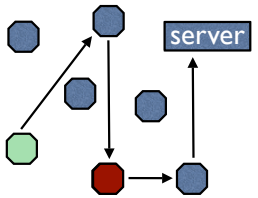
Crowds

- A "crowd" of n users participates in the protocol
- The initiator forwards the message to a randomly selected user (forwarder)
- A forwarder:
  - With probability $1-p_f$ forwards again the message
  - With probability p_f send the message directly to the server



Anonymity of the protocol

- Wrt the server: strong anonymity. The server sees only the last user
- More interesting case: some user is corrupted
- Information gathered by the corrupted user can be used to detect the initiator



Anonymity of the protocol

- In presence of corrupted users:
  - strong anonymity is no longer satisfied
  - A weaker notion called "probable innocence" can be achieved, informally defined as:

"the detected user is less likely to be the initiator than not to be the initiator"