Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Systems@ETH Zürich

Prof. T. Roscoe, Prof. R. Wattenhofer

Computer Systems

Assignment 12

1 Game Theory

Quiz

1.1 Selling a Franc

Form groups of two to three people. Every member of the group is a bidder in an auction for one (imaginary) franc. The franc is allocated to the highest bidder (for his/her last bid). Bids must be a multiple of CHF 0.05. This auction has a crux. Every bidder has to pay the amount of money he/she bid (last bid) – it does not matter if he/she gets the franc. Play the game!

a) Where did it all go wrong?

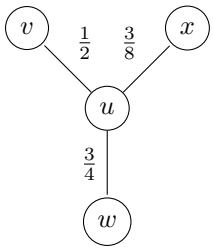
b) What could the bidders have done differently?

Basic

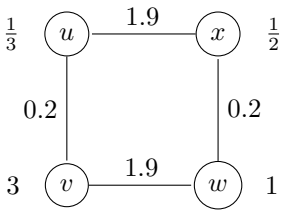
1.2 Selfish Caching

a) For each of the following caching networks, compute the social optimum, the pure Nash equilibria, the price of anarchy (PoA) as well as the optimistic price of anarchy($OPoA$):

i. $d_u = d_v = d_w = d_x = 1$



ii. The demand is written next to a node.



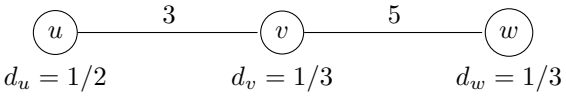
1.3 Selfish Caching with variable caching cost

The selfish caching model introduced in the lecture assumed that every peer incurs the same caching cost. However, this is a simplification of the reality. A peer with little storage space could experience a much higher caching cost than a peer who has terabytes of free disc space available. In this exercise, we omit the simplifying assumption and allow variable caching costs α_i for node i.

What are the Nash Equilibria in the following caching networks given that

i. $\alpha_u = 1$, $\alpha_v = 2$, $\alpha_w = 2$,

ii. $\alpha_u = 3$, $\alpha_v = 3/2$, $\alpha_w = 3$?



Does any of the above instances have a dominant strategy profile? What is the PoA of each instance?

Advanced

1.4 Matching Pennies

Tobias and Stephan like to gamble, and came up with the following game: Each of them secretly turns a penny to heads or tails. Then they reveal their choices simultaneously. If the pennies match Tobias gets both pennies, otherwise Stephan gets them.

Write down this 2-player game as a bi-matrix, and compute its (mixed) Nash equilibria!

1.5 PoA Classes

The PoA of a class $\mathcal{C}$ is defined as the maximum PoA over all instances in $\mathcal{C}$. Let

- $\mathcal{A}^n_{[a,b]}$ be the class of caching networks with n peers, $a \leq \alpha_i \leq b$, $d_i = 1$, and each edge has weight 1,
- $\mathcal{W}^n_{[a,b]}$ be the class of networks with n peers, $a \leq d_i \leq b$, $\alpha_i = 1$, and each edge has weight 1.

Show that $PoA(\mathcal{A}^n_{[a,b]}) \leq \frac{b}{a} \cdot PoA(\mathcal{W}^n_{\left[\frac{1}{b},\frac{1}{a}\right]})$ for all $n > 0$.

2 Authenticated Agreement

Quiz

2.1 PBFT: Basics

a) At which point of the agreement protocol can a node be sure that all correct nodes can only agree on the same request for a given sequence number within the current view?

b) During a view change, how can the backups be sure the new primary did not just make up requests that he wants them to execute?

c) During a view change, will only requests that were already executed by some correct node be included in the set $\mathcal{O}$?

d) It is possible that a node collected a prepared-certificate that will not be included in a new-view-certificate. Why is this not a problem?

Basic

2.2 PBFT: Utility of the Phases of the Agreement Protocol

In the PBFT agreement protocol, some phases seem superfluous. The purpose of this exercise is to get an insight to why those phases are necessary.

a) If correct nodes would not forward requests to the primary, how could a byzantine client slow down the system?

b) Assume that correct nodes do not wait for $2f + 1$ `commit`-messages in the execute phase of the agreement protocol (Algorithm 25.17). Instead, they execute a request as soon as they have a prepared-certificate for it. How could it happen that two different correct nodes execute two different requests with the same sequence number?

Advanced

2.3 Authenticated Agreement

Algorithm 25.2 in the lecture uses authentication to reach agreement in an environment with byzantine processes.

a) Modify Algorithm 25.2 in such a way that it handles arbitrary input. Write your algorithm as pseudo-code. The processes may also agree on a special "sender faulty"-value. Hint: consider a set of values that correct nodes reached agreement for, then work with the size of the set.

b) Prove the correctness of your algorithm.

2.4 Multiple Prepared-Certificates for the Same Sequence Number!? — How can this happen?

One final corner case we did not consider in the script is implied by the answers to the quiz questions c) and d). In this exercise, we consider the possibility of multiple prepared-certificates for the same sequence number but different requests ending up in the $\mathcal{V}$-component of a `new-view`-message. How can it happen that the $\mathcal{V}$-component of a `new-view`-message contains two prepared-certificates, one for (v, s, r) and one for (v', s, r') with $r \neq r'$?

Remark: Exercise **??** *asks you to show how this can be dealt with.*

2.5 Multiple Prepared-Certificates for the Same Sequence Number!? — How can we fix it?

As we saw in Exercise **??**, it is possible that multiple contradictory prepared-certificates for the same sequence number end up in the $\mathcal{V}$-component of a `new-view`-message. How does the primary have to choose between those prepared-certificates when generating $\mathcal{O}$ such that the system remains correct? Prove that your proposal guarantees correctness!