Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Systems@ETH Zürich

Autumn Term 2019 Prof. T. Roscoe, Prof. R. Wattenhofer

Computer Systems

Assignment 11

Assigned on: **December 2, 2019**

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?

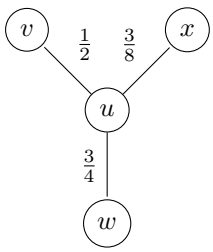
b) What could the bidders have done differently?

Basic

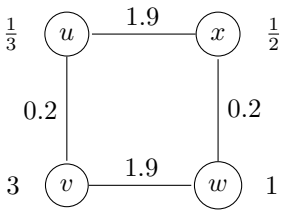
1.2 Selfish Caching

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.



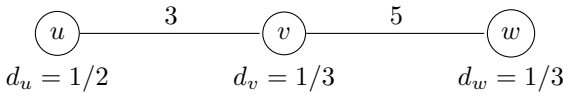
1.3 Selfish Caching with variable caching cost

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 Distributed Storage

Quiz

2.1 Hypercubic Networks

Draw the following hypercubic networks:

a) $M(3,1)$

b) $M(3,2)$

c) $SE(2)$

d) $M(2,4)$

Basic

2.2 Iterative vs. Recursive Lookup

There are two fundamental ways to perform a lookup in an overlay network: recursive and iterative lookup.

Assume node n_0 is attempting to look up an object in a DHT. In the recursive lookup n_0 selects a node n_1 which is closest according to the DHT metric and sends a request to it. Upon receiving the request n_1 selects its closest known neighbor n_2 and forwards the request to it and so on. The request either ends up at the node storing the object, returning the object along the same path, or it ends at a node that does not store the object and does not have a closer neighbor.

In the iterative case n_0 looks up the closest neighbor n_1 and sends it the request. Upon receiving the request n_1 is either the node storing the object and it returns the object, or it knows a closer node n_2 and returns n_2 to the n_0. If n_0 receives a node n_2 it will add it to its neighbor set and sends a new request to n_2 which is now its closest neighbor. The lookup terminates either when n_0 sends a request to the node storing the object, or no closer node can be found.

a) What are the advantages of recursive lookups over the iterative lookups?

b) Most systems that are in use today use the iterative lookup, and not the recursive lookup, why?

2.3 Building a set of Hash functions

Consistent hashing relies on having k hashing functions $\{h_0, \ldots, h_{k-1}\}$ that map a node's unique name and the object ids to hashes. There are several constructions for these hash functions, the most common being iterative hashing and salted hashing. In iterative hashing we use a hash function h and apply it iteratively so that the hashes of an object id o is defined as

$$h_i(o) = \begin{cases} h(o) & \text{if} \quad i = 0 \\ h(h_{i-1}(o)) & \text{otherwise.} \end{cases}$$

With salted hashing the object id is concatenated with the hash function index i resulting in the following definition

$$h_i(o) = h(o|i).$$

Which hashing function derivation is better and why?

Advanced

2.4 Multiple Skiplists

In the lecture we have seen the simple skip list in which a node is in the root level and promoted with probability 1/2. We now redefine the promotion so that a node is promoted to a list s if s is a suffix of the binary representation of the node's id. At each level l we now have multiple lists, each defined by a suffix s of length l. The root level is defined as the empty suffix with l. The first level has two lists $p \in \{0, 1\}$, the second level has four lists $p = \{00, 01, 10, 11\}$ and so on. We call the resulting network a multi-skiplist.

a) Assuming we have an 8 node network, with ids $\{000, \ldots, 111\}$, draw the multi-skiplist graph.

b) What is the minimum degree of a node in the multi-skiplist if we have d levels?

c) What is the maximum number of hops a lookup has to perform?