

Lab 1- Presentation

Creating your first distributed blackboard



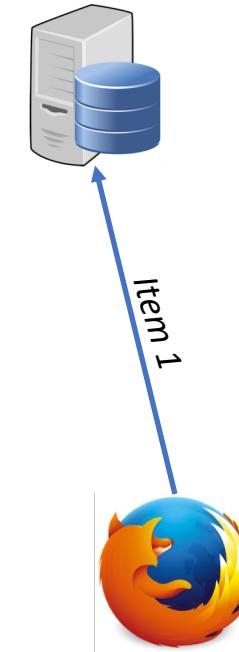
Distributed Blackboard

- A web application running on multiple (distributed) servers



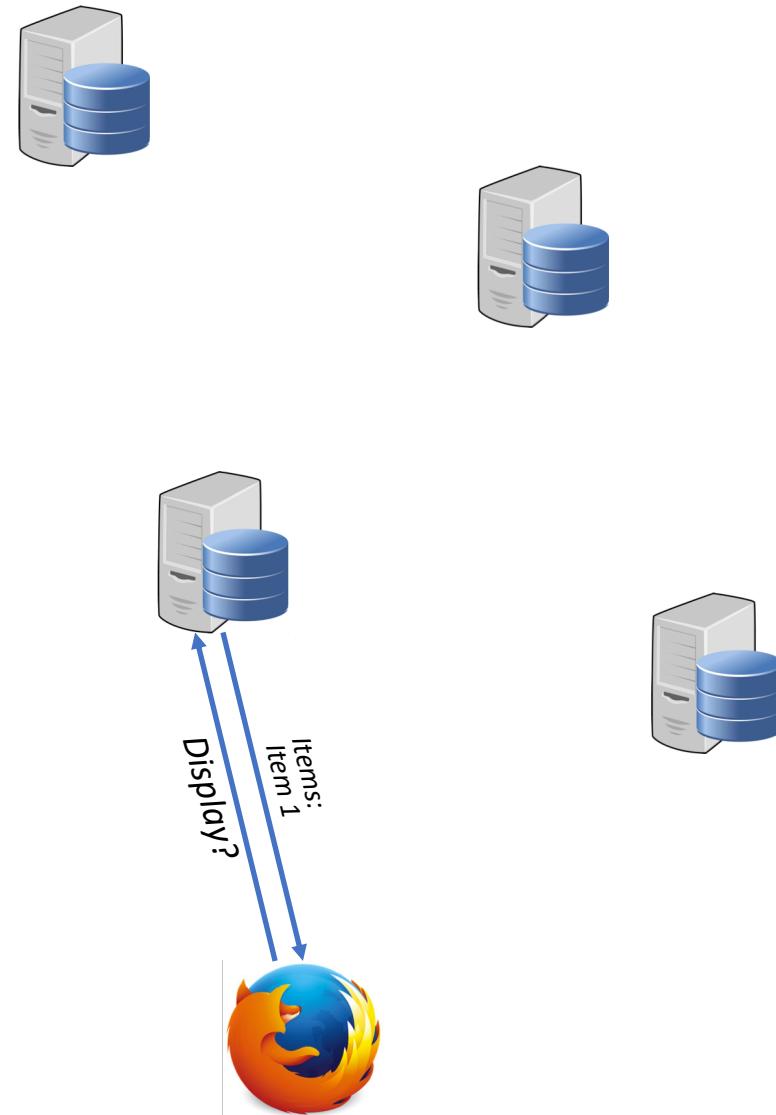
Distributed Blackboard

- A web application running on multiple (distributed) servers
- Clients can connect to any server and post information via a web browser



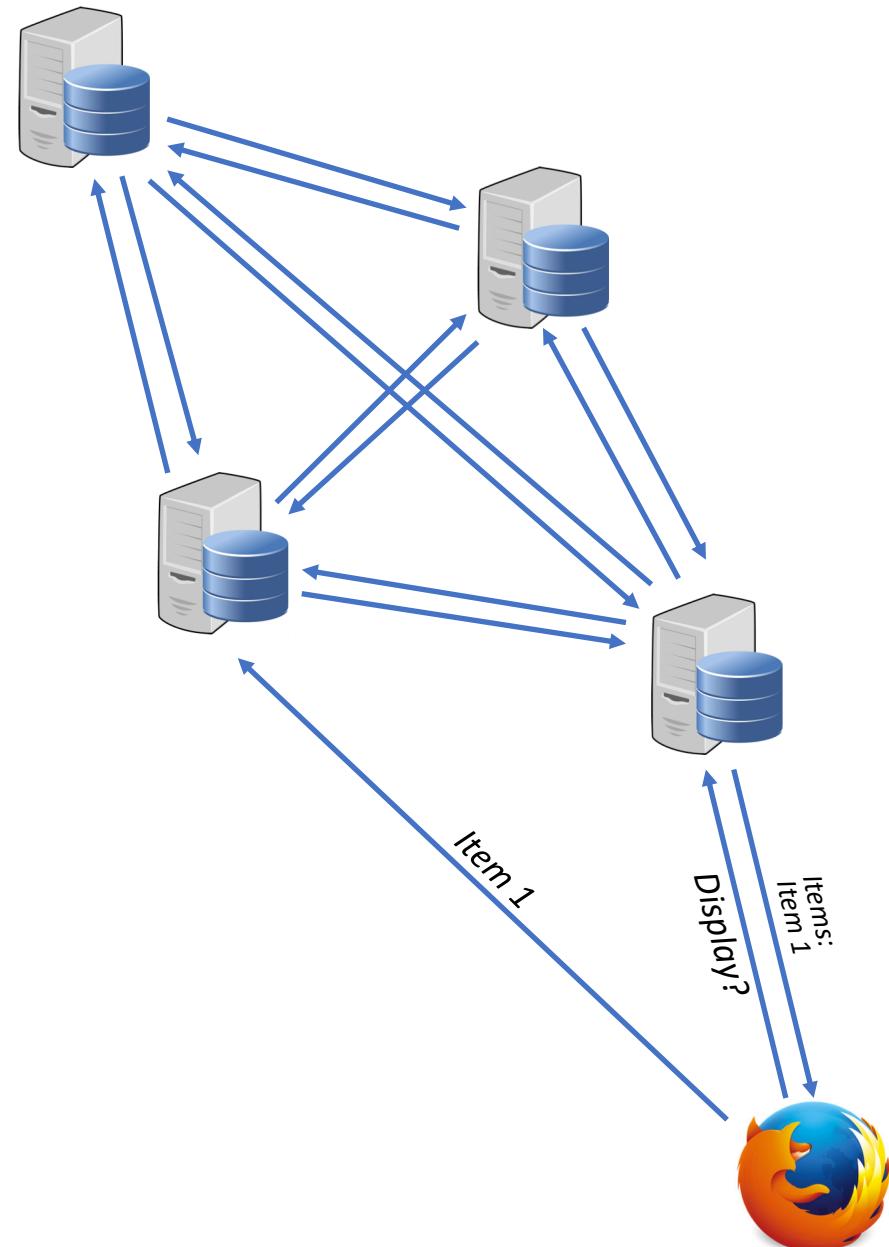
Distributed Blackboard

- A web application running on multiple (distributed) servers
- Clients can connect to any server and post information via a web browser
- Server should store all data received, and display it



Distributed Blackboard

- A web application running on multiple (distributed) servers
 - Clients can connect to any server and post information via a web browser
 - Server should store all data received, and display it
 - Client data must be seen from any server
 - Propagate information to all boards
 - In a peer-to-peer-manner



The blackboard application

- It might look like this (or slightly better)

A screenshot of a web browser window titled "Blackboard GroupName - Chromium". The address bar shows "IP1:myPort". The page content includes:

- A green callout bubble pointing to a text input field and a "Submit to board" button with the text "A form to submit text".
- The text "Submit to board" followed by a button.
- A table titled "Board contents" with columns "Sequence Number" and "Entry". It contains one row with "0" and "test".
- A green callout bubble pointing to the "test" entry in the table with the text "Some blackboard entries".
- A green callout bubble pointing to the "Modify" and "Delete" buttons in the table with the text "Options for each entry".
- The text "Group members: member1@student.chalmers.se, member2@student.chalmers.se".

Example

- Initially, blackboard empty on all servers

The image contains two side-by-side screenshots of a Chromium browser window. Both windows have the title "Blackboard GroupName - Chromium" and show a search bar with "IP1:myPort" and "IP2:myPort" respectively. The browser interface includes standard navigation buttons (back, forward, search, etc.) and a status bar at the bottom.

Left Screenshot (IP1:myPort):

- Page status: "reloading page in: 0 seconds."
- Error message: "3: error- undefined"
- Section: "Submit to board" with a "Submit to board" button.
- Section: "Board contents" with a table:

Sequence Number	Entry
- Group members: member1@student.chalmers.se, member2@student.chalmers.se.

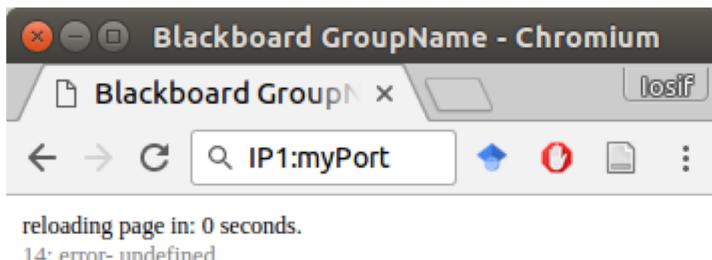
Right Screenshot (IP2:myPort):

- Page status: "reloading page in: 2 seconds."
- Error message: "9: error- undefined"
- Section: "Submit to board" with a "Submit to board" button.
- Section: "Board contents" with a table:

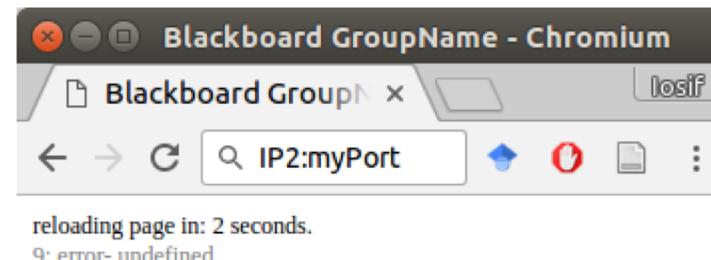
Sequence Number	Entry
- Group members: member1@student.chalmers.se, member2@student.chalmers.se.

Example

On server 1:
Write text and submit

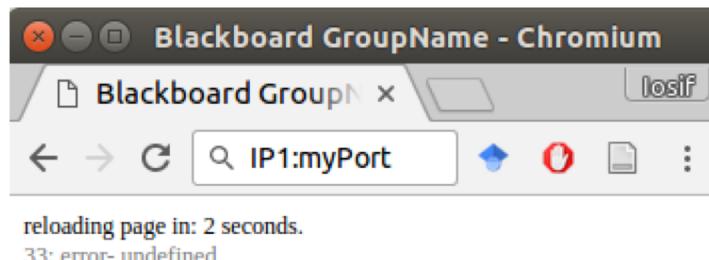


On server 2:
No action on the browser

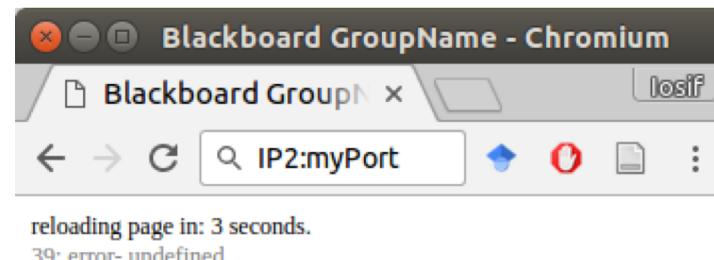


Example

On server 1:
No action on the browser



On server 2:
Hit refresh and see post!



Example

A user should also be able to **modify** and a **delete** each post.

The image displays two screenshots of a web application window titled "Blackboard GroupName - Chromium".

Left Screenshot: The address bar shows "IP1:myPort". The page content includes a message "reloading page in: 2 seconds." and "33: error- undefined". Below this is a "Submit to board" section with a "Submit to board" button. The "Board contents" section shows a table with one row:

Sequence Number	Entry
0	test

Buttons for "Modify" and "Delete" are visible next to the entry. A green arrow points upwards from the "Modify" button, and an orange arrow points downwards to the "Delete" button.

Right Screenshot: The address bar shows "IP2:myPort". The page content includes a message "reloading page in: 3 seconds." and "39: error- undefined". Below this is a "Submit to board" section with a "Submit to board" button. The "Board contents" section shows a table with one row:

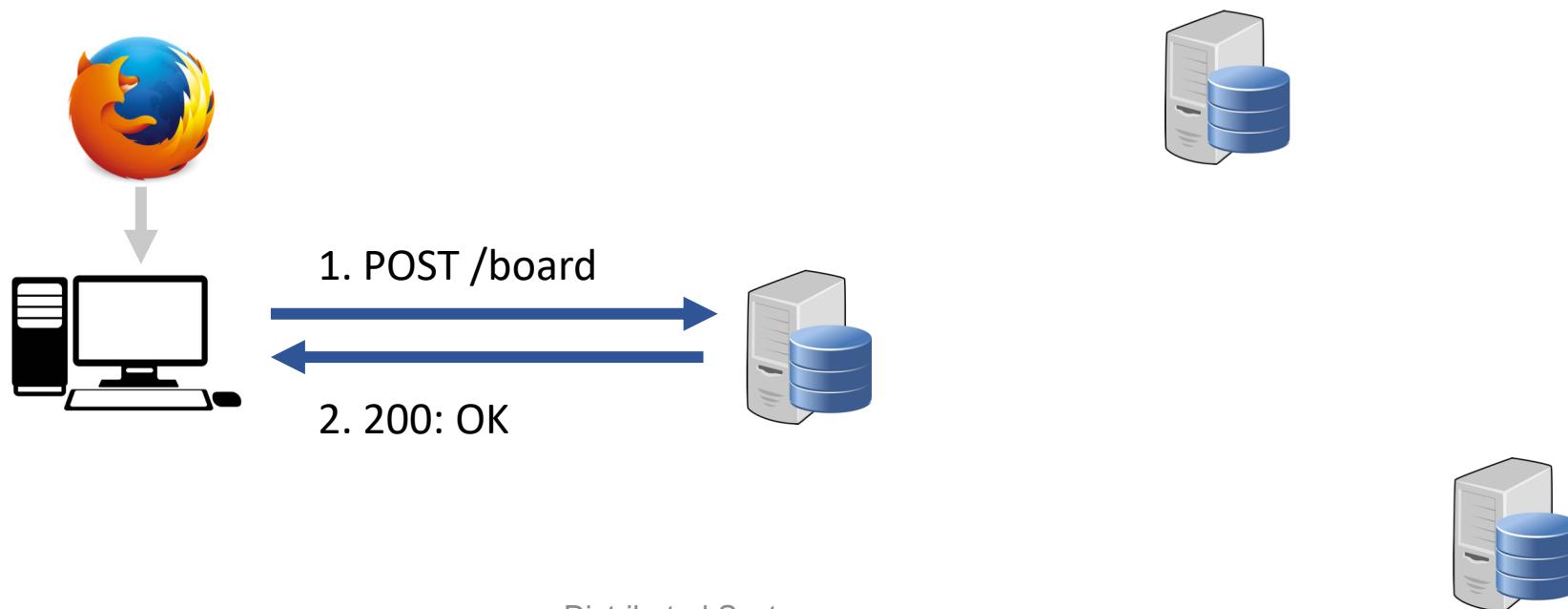
Sequence Number	Entry
0	test

Buttons for "Modify" and "Delete" are visible next to the entry.

Group members: member1@student.chalmers.se, member2@student.chalmers.se

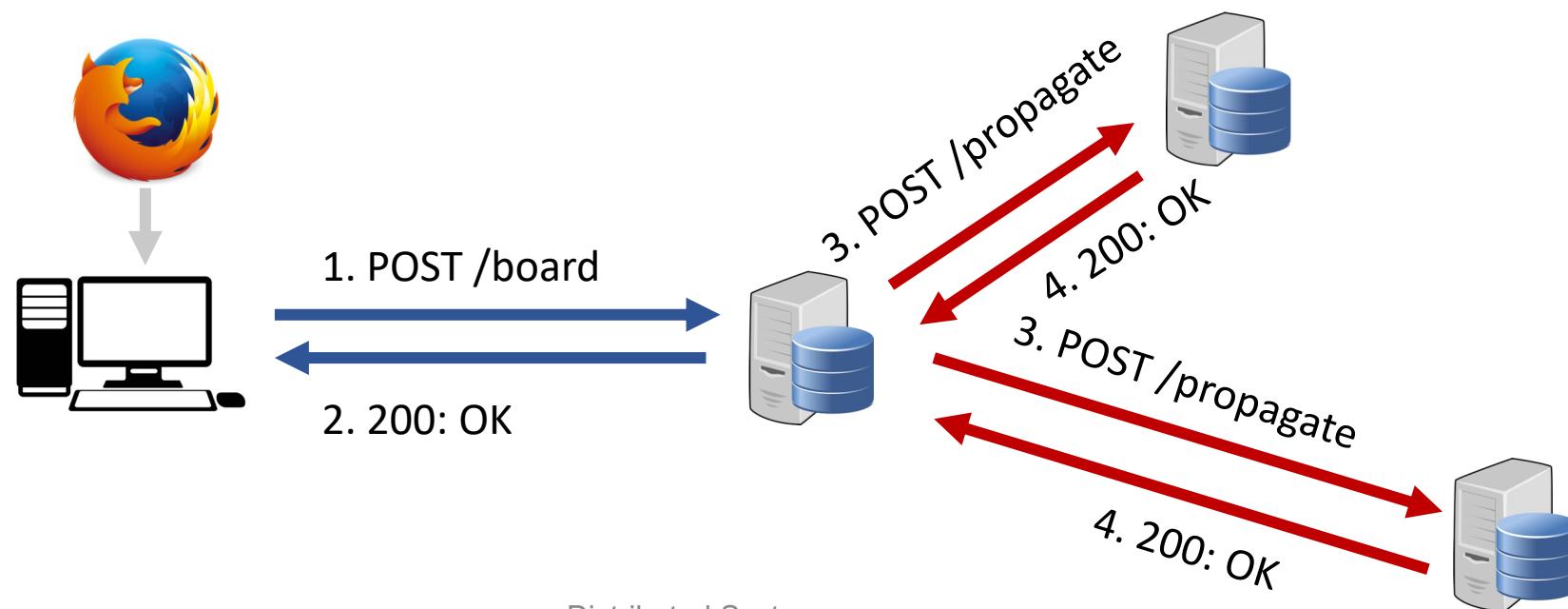
Client-Server Communication

Action	URI	Explanations	Parameters
GET	/	Display index page	
GET	/board	Display blackboard content only	
POST	/board	Add new value to blackboard	<i>entry</i>
POST	/board/<number>	Modify or delete an element from blackboard	<i>entry, delete</i>



Server-Server Communication

Action	URI	Explanations	Parameters
POST	/propagate	Propagate information	You decide!
You decide!	You decide!	You decide!	You decide!



Hints

- Keep a list of all servers locally
 - Not very scalable, but adding/removing servers is hard, trust us!
- Upon a post
 - Locally save the value
 - Propagate the data to all other servers
 - But don't wait until you hear from all of them! This might take a lot of time, and the client is waiting for your answer!
 - Propagation should be done in parallel to responding to client

More hints

- You do not need to look at the mininet script
 - But feel free to
- Upon starting mininet, each server starts *server/server.py*
 - This is your Bottle application
- Bottle will use a template in */templates*
 - *blackboard.tpl* handles the blackboard display
 - *header.tpl* handles everything up to the board
 - *footer.tpl* handles everything after the board

What we give you

- A Mininet script, *start_topology.py*
 - Starts the network, the servers and the server code!
 - Run it with *sudo python start_topology.py*, from the folder!
- A simple Bottle server with template
 - You should improve *server/server.py*
 - You might have to change *server/templates/blackboard.tpl*

Lab 1 -Tasks

What you NEED to implement



Task 1 – make it work

- Implement *Add* in your distributed blackboard
- Scenario:
 - Client connects to server 1, and posts 5 messages
 - Client connects to server 2 and posts 5 messages
 - Client connects to server 3 and should see 10 messages
- Requirements:
 - ≥ 8 servers

Task 2 – modify/delete

- Implement *Modify/Delete*
- Scenario:
 - Client connects to server 1, and posts 5 messages
 - Client connects to server 2 and modifies 1 message
 - Client connects to server 3 and deletes 1 message
 - Client connects to server 4 and should see 4 messages, with one modified
- Requirements:
 - ≥ 8 servers

Task 3 - consistency

- Investigate consistency
- Scenario 1:
 - Client connects to all servers and posts 5 messages to each, concurrently
 - Do server 3 and 6 show the same blackboard, in the same order?
- Scenario 2:
 - Client connects to all servers and posts 5 messages to each, concurrently
 - Client connects to all servers and modifies the same element to a different value (for each server), concurrently
 - Do server 4 and 7 show the same blackboard, in the same order?
- Requirements:
 - ≥ 8 servers
 - Create a script to concurrently send requests

Lab 1 - deliverables

What you **NEED** to give/show us



Deliverables

- Your implementation
 - Including any script you used to test consistency!
 - Upload it on iLearn
- A video of your work
 - ~ 5 min (max 10)
 - Upload on the CAU-cloud
 - Everybody will be able to watch your video!
 - Should include (at least) Design – Demo – Evaluation
 - see Lab presentations for details

Deadline:
Thursday 19th, November

Good luck, and start coding!

