Codeine - Computing over Decentralized Network, with P2P

AGH University of Science and Technology
Faculty of Electrical Engineering, Automatics, Computer Science and Engineering in Biomedicine

Kacper Tonia Przemysław Nocoń Kuba Komnata Sławomir Kalandyk

1 Glossary

- Agent - single application instance, is able to compute one subproblem (task) at a time
- Computational problem - problem solvable with Codeine. It should be divisible into a finite amount of subproblems which can be solved independent of each other
- Computational network (usually referred to as "network") - a network of agents communicating with each other, who together solve one computational problem
- Subproblem - a single subproblem of the computational problem

2 Requirements

- Project should implement peer-to-peer networking on LAN
- We should assume that about 5 agents at once can work on our computational problem
- Every agent should have exactly the same application
- After application launch, agent should automatically attempt to discover other agents in the network
- The computational problem itself doesn't matter, it should only allow for long enough computing time to let us see the network working as intended (officially 1 hour on 5 agents, non-officially 10 - 20 mins)
- Task assignment should be decentralized
- Agents should be immune to other agents disconnecting from the network, there should be no side effects
- Results should be visualized, accessible (in the best case in real time)

3 Assumptions & Constraints

- Packet type - a seven letter long word consisting only of upper case letters (e.g. IMALIVE, NETTOPO)
- The solution of a single task should be able to fit in a single UDP packet (`<64kB`)
- Every task has it's ID and immutable State, common for all tasks
- Every task can be solved

4 Networking

4.1 Packets

- Topology discovery, registering agents, agent check
  - IMALIVE - send empty packet informing that agent is still in the network <>
  - NETTOPO - send network topology <agent[]>
- Task assignment
  - REGTASK - register new task <task_id>
  - STOPWIP - don't start: this task is work-in-progress <task_id>
  - STOPLOW - don't start: lower priority <task_id>
- Result distribution
  - TASKRES - send task result <task_id, task_result>
- Confirmation
  - ACKNOWL - acknowledge <previous_package_control_sum>

4.2 Rules

- broadcast

- #IMALIVE → NETTOPO (optional)
- #REGTASK → STOPWIP | STOPLOW | TASKRES
- #TASKRES → ACKNOWL

4.3 Network scenarios

4.3.1 Scenario 1

Story:
Agent tries to join the network right after launching Codeine.

Prerequisites:

- None

Scenario:

1. Agent broadcasts IMALIVE packet.
2. Agent waits for ??? seconds for replies from every other agent already in the network.
3. Every other agent already in the network replies with NETTOPO packet.
4. Agent registers the network. End of Scenario 1.

Scenario extensions:

3a. There is no response from the network (proceed to calculations). End of Scenario 1.

4.3.2 Scenario 2

Story:
Agent periodically informs the network that he's still alive.

Prerequisites:

- Agent is already in the network

Scenario:

1. Agent broadcasts IMALIVE packet. End of Scenario 2.

4.3.3 Scenario 3

Story:
Agent wants to register a task

Prerequisites:

- Agent is already in the network

Scenario:

1. Agent broadcasts REGTASK packet.
2. Agent starts calculations.
3. No response from network. End of Scenario 3

Scenario extensions:

3a. An agent replies with STOPWIP.

- 3a.1. Agent stops calculations.
- 3a.2. Agent sets that task's state to WIP.
- 3a.3. Agent chooses another task. Repeat from point 1. End of Scenario 3.

3b. An agent replies with STOPLOW.

- 3b.1. Agent stops calculations.
- 3b.2. Agent chooses another task. Repeat from point 1. End of Scenario 3.

3c. An agent replies with TASKRES.

- 3c.1. Agent stops calculations.
- 3c.2. Agent registers received task result.
- 3c.3. Agent chooses another task. Repeat from point 1. End of Scenario 3.

4.3.4 Scenario 4

Story:
Agent wants to broadcast task results.

Prerequisites:

- Agent is already in the network
- Agent has calculated a task and received a concrete result

Scenario:

1. Agent broadcast TASKRES packet.
2. Agent waits for ??? seconds for replies.
3. Every other agent registered in his net topology base replies with ACKNOWL. End of Scenario 4.

Scenario extensions:

3a. If any agent hasn't replied with ACKNOWL, send packet TASKRES again to him. Repeat ??? times.

 3a.1. An agent hasn't replied after ??? sent TASKRES packets. Remove him from net topology base.

4.3.5 Scenario 5

Story:
Agent has received a TASKRES packet with task ID of a task he already has a result of.

Prerequisites:

- Agent is already in the network
- Agent already has results of at least one task

Scenario:

1. Agent received a TASKRES packet.
2. Agent tries to register the task result. Task with that ID already has a registered result.
3. Task result of received packet is ignored.
4. Reply with ACKNOWL packet. End of Scenario 5.

4.3.6 Scenario 6

Story:
Agent A tries to register a task with ID == X. Agent B replies with STOPWIP packet. Agent A sets task X's state to WIP. Agent B then disconnects.

Prerequisites:

- Agent is already in the network

Scenario:

1. Agent A broadcasts REGTASK packet with task ID == X.
2. Agent B replies with STOPWIP.
3. Agent A sets task X's state to WIP.
4. Agent B disconnects.
5. Agent A doesn't receive IMALIVE packets from agent B for ??? minutes.
6. Agent A sets task X's state to unregistered/free/not-yet-completed. End of Scenario 6.

5 Computational problem

The chosen problem is finding a hash created with SHA1 cipher corresponding to a hardcoded hash of a 6 letter password. The goal is to find hashes for all possible 6 character combinations of lower case letters and digits and compare them to the hardcoded password until the correct one is found. In the worst case scenario, 36^6 hashes have to be calculated.

To solve it with a decentralized computing network, it has been split into 36^2 subproblems, each consisting of 36^4 hashes to calculate. They are divided based on the first two characters, e.g. one subproblem is to find hashes of all 6 character long strings that start with "bg". Each agent can compute only one subproblem at a time.

6 Diagrams

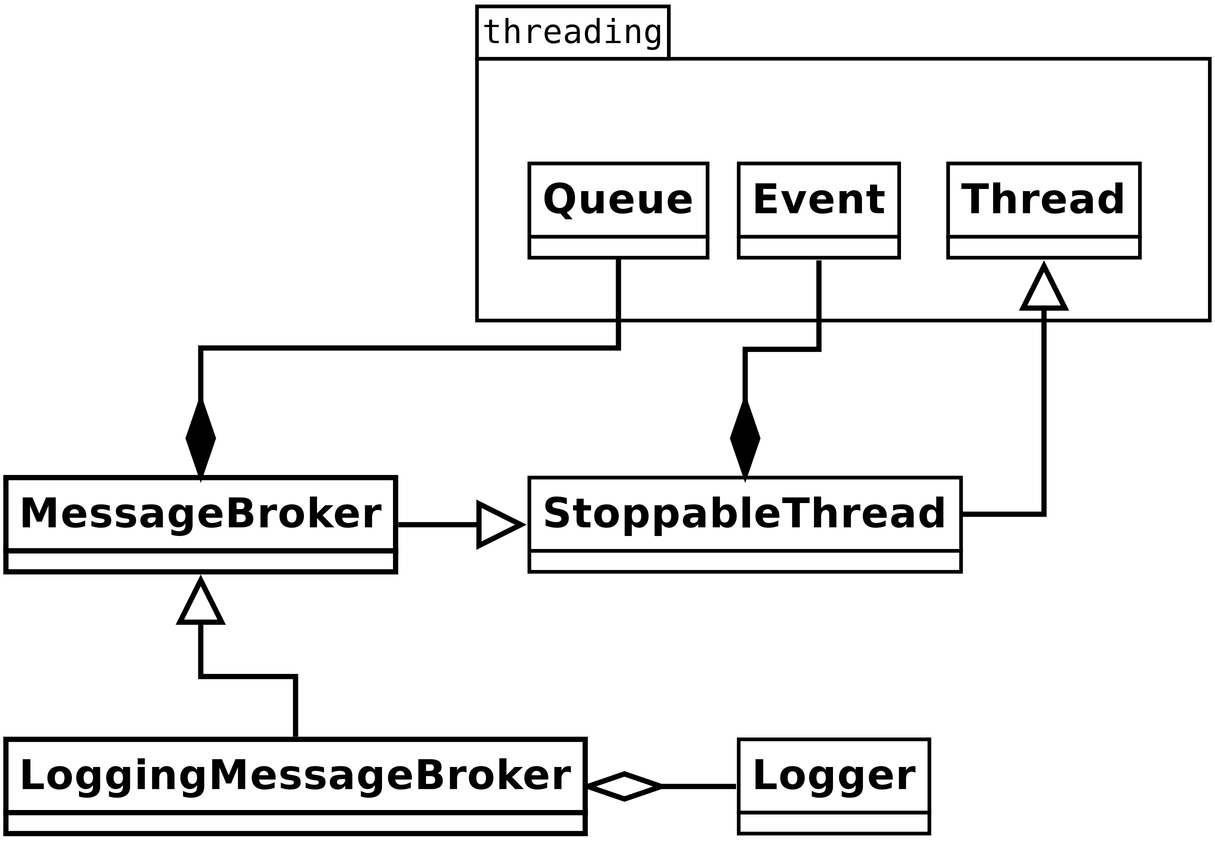

Figure 1: MessageBroker Class Diagram

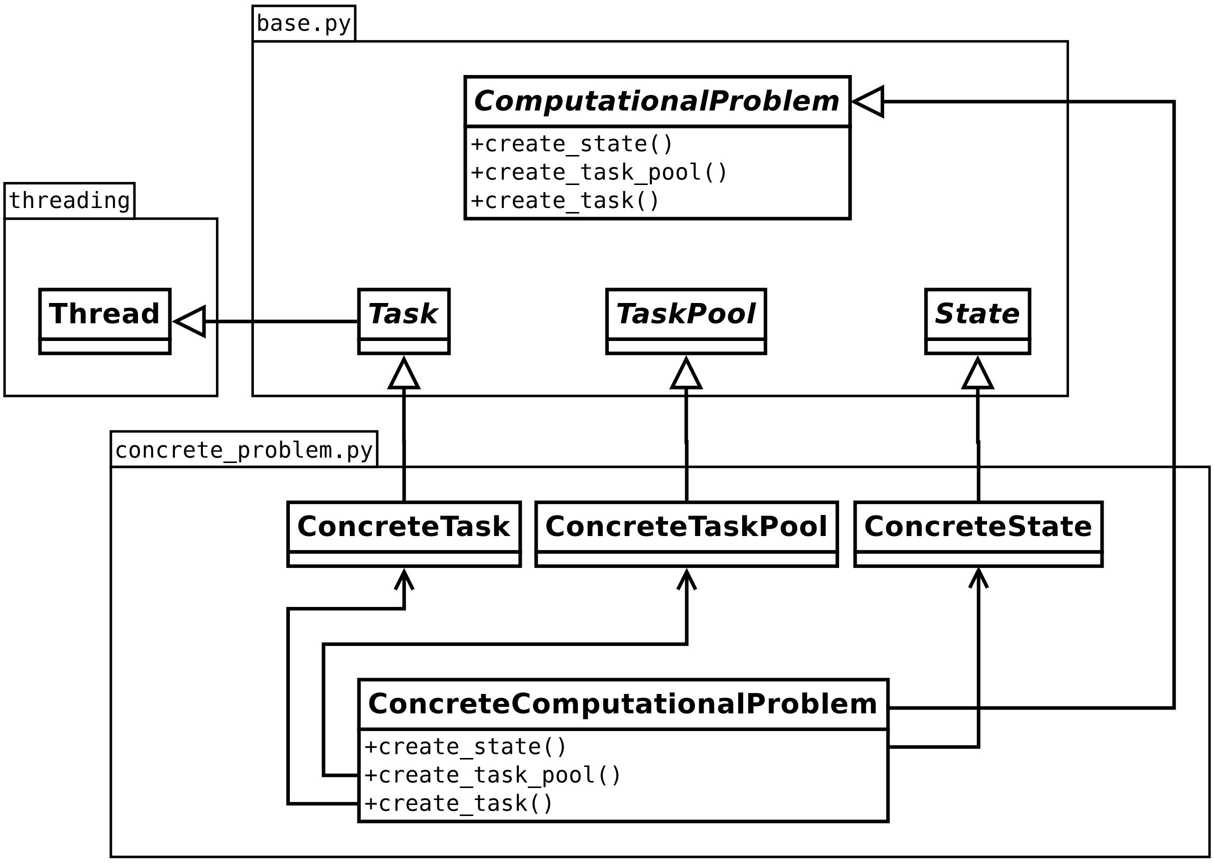

Figure 2: Computational Problem Factory Diagram