POCS Recitations

Rishabh Iyer

23.09.2021

What is a recitation?

- In-depth discussion/brainstorm about assigned readings
- TAs introduce the papers and lead the discussion
  - Like any discussion, learning is maximized when everyone participates
  - Would be great if everyone has at least one comment/question per recitation
- Ask as many questions as you need
  - There are no stupid questions

Recitation Format

- 2x Per-paper discussion (~40 mins each)
  - ❖ Quick Recap - revise facts in the paper
  - ❖ In-depth discussion - strengths, weaknesses, alternate solutions, tradeoffs
- 1x Design exercise (~30 mins)
  - ❖ Design a similar system with different assumptions in groups

How to prepare for a recitation

- Read the papers
- Read the papers
- Read the papers

How to read the papers?

- Understand the basics of the paper
  - Problem it solves, main insights/ideas
- Read in multiple passes
  - First a quick read to get the gist,
  - Subsequent reads to understand content in-depth
- Move on if stuck at specific implementation details
  - It's ok as long as you can get the point of the paper

Preparing for the recitation

- Prepare your questions on the paper
  - ❖ Make sure they are answered before the in-depth discussion
- Prepare your (critical) thoughts on the paper
  - ❖ Strengths/Weaknesses of the paper
  - ❖ Alternative solutions to the problem
  - ❖ How does the paper embody the POCS principle of the week?
  - ❖ Other interesting stuff you want to share, e.g., related papers/blogs

How to read the BL paper?

- Like the other papers: in multiple passes
- Challenge yourself
  - Do you understand every single sentence in depth?
  - Do you understand every example?
  - Can you think of your own example for each design principle?

Breaking down BL quotes

Slides borrowed from Katerina Argyraki

"Don't hide power"

- "The point of an abstraction is to hide how the code is doing its work, but it should not prevent a client from using all the power of its host."
- "The Internet's UDP protocol is an example; unlike reliable TCP, it gives clients direct access to the basic unreliable best-efforts packet delivery, which is critical for real-time applications like voice."

- Who are the clients of TCP and UDP?
  - ❖ The internet applications

"Don't hide power"

- "The point of an abstraction is to hide how the code is doing its work, but it should not prevent a client from using all the power of its host."
- "The Internet's UDP protocol is an example; unlike reliable TCP, it gives clients direct access to the basic unreliable best-efforts packet delivery, which is critical for real-time applications like voice."

- Why are TCP and UDP abstractions?
  - ❖ They hide the machinery of the network, link and physical layers from internet applications

"Don't hide power"

- "The point of an abstraction is to hide how the code is doing its work, but it should not prevent a client from using all the power of its host."
- "The Internet's UDP protocol is an example; unlike reliable TCP, it gives clients direct access to the basic unreliable best-efforts packet delivery, which is critical for real-time applications like voice."

- What power does TCP hide from its clients?
  - ❖ Transferring data from a source to a destination with the latency of a single one-way trip and throughput equal to the full bandwidth of the path between them

“Don’t hide power”

- “The point of an abstraction is to hide how the code is doing its work, but it should not prevent a client from using all the power of its host.”
- “The Internet’s UDP protocol is an example; unlike reliable TCP, it gives clients direct access to the basic unreliable best-efforts packet delivery, which is critical for real-time applications like voice.”

- Why is this power critical for real-time applications like voice?
  - ❖A real-time application typically prefers timely delivery of some of the data to delayed delivery of all the data

"Keep it simple"

- "A brute force solution may be the best option. "Computers are fast… take advantage of this."
- "Broadcast is [an] example of brute force.
  It is to routing as exhaustive search is to indexing, and likewise scales badly.
  In networking you often need a broadcast to get started, for example when an ethernet node needs to find an Internet router, but this doesn't need to scale."

- In what sense is broadcast brute force?
  - ❖ It involves sending a packet to every single end-system in the network.
    This is like trying all possible passwords

“Keep it simple”

- “A brute force solution may be the best option. “Computers are fast… take advantage of this.”
- “Broadcast is [an] example of brute force.
  It is to routing as exhaustive search is to indexing, and likewise scales badly.
  In networking you often need a broadcast to get started, for example when an ethernet node needs to find an Internet router, but this doesn’t need to scale.”

- Why does broadcast “scale badly”
  - ❖ Per-packet transmissions is linear in the number of end-systems in the network. When a new-end system joins, everyone has to pay

“Keep it simple”

- “A brute force solution may be the best option. “Computers are fast… take advantage of this.”
- “Broadcast is [an] example of brute force.
  It is to routing as exhaustive search is to indexing, and likewise scales badly.
  In networking you often need a broadcast to get started, for example when an ethernet node needs to find an Internet router, but this doesn’t need to scale.”

- Which protocol (that uses broadcast) is Lampson referring to?
  - ❖ Address Resolution Protocol (ARP)

“Keep it simple”

- “A brute force solution may be the best option. “Computers are fast… take advantage of this.”
- “Broadcast is [an] example of brute force.
  It is to routing as exhaustive search is to indexing, and likewise scales badly.
  In networking you often need a broadcast to get started, for example when an ethernet node needs to find an Internet router, but this doesn’t need to scale.”

- Why is it OK for ARP to use broadcast?
  - ❖ ARP requires broadcasting only within a local network which has a few end-systems. So its OK to transmit a packet to all of them.

BL paper (Chapters 1,2)

Specifications

- Much smaller and simpler than the code
- Decouples client from code details
- Two parts to a spec:
  - Abstract state
  - Spec actions

Bad and leaky specifications

- Can you think of examples of widely used systems with incomplete and/or leaky specifications?

Group Design Exercise

- Write a interface/specification for a TCP-like transport protocol that promises reliable, in-order delivery of messages. Like TCP this transport protocol is also responsible for congestion control to ensure the network is not overloaded. You can assume that the receiver can be identified using a receiver id.

- Questions to consider:

  - What are the methods in the interface?

  - What is the abstract state required to express all desirable properties?