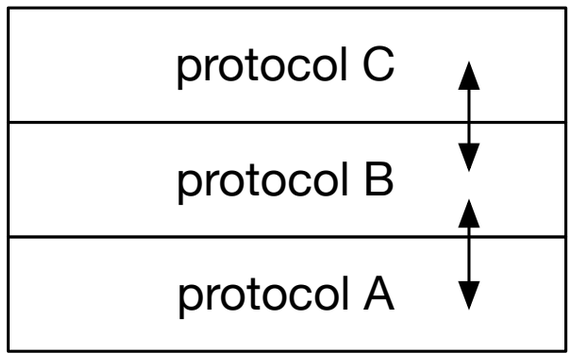
Practice questions

416 2021 W2 (Winter 2022)

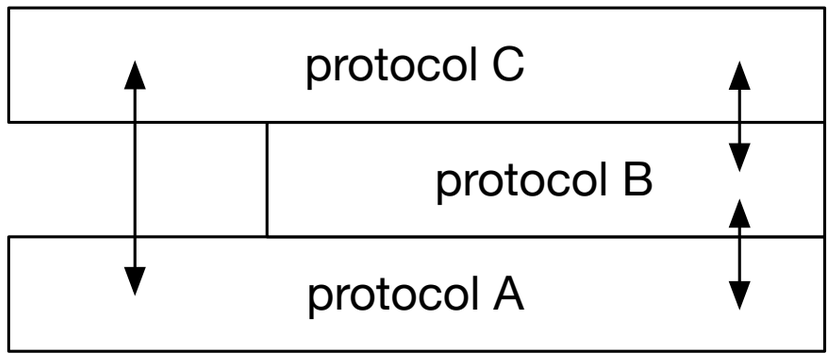
These questions are intended to simulate the final exam. They cover previous lecture/assignment material that is fair game for the final exam.

PQ 1

- You are designing a protocol stack. You have narrowed down your design to two choices. And, you know that the specification for protocol C is likely to change. Which stack design should you use?



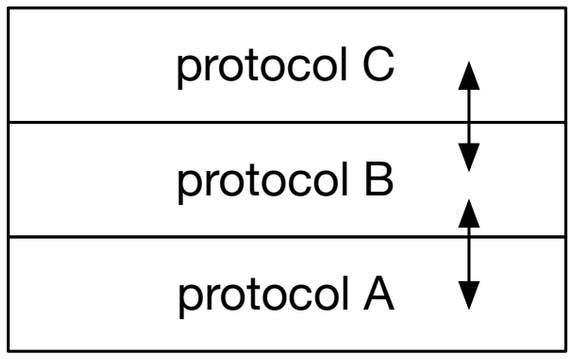
(a)



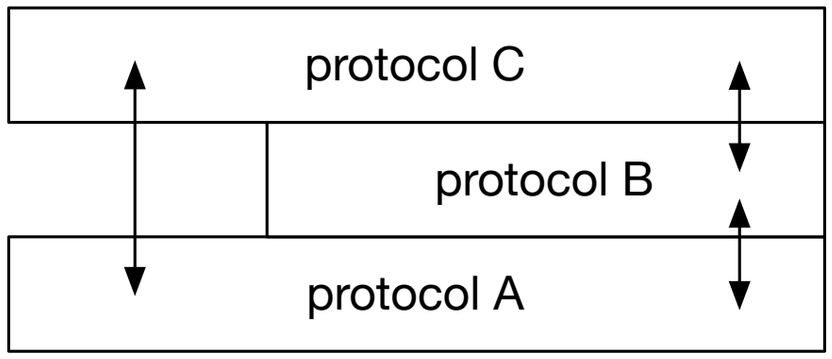
(b)

PQ 1

- You are designing a protocol stack. You have narrowed down your design to two choices. And, you know that the specification for protocol C is likely to change. Which stack design should you use?



(a)



(b)

- If protocol C changes then only protocol B would need to adapt, and not A

PQ 2

- A network element can inspect any of the protocols present in the packet. So, **why not** build e.g., a switch that is aware of HTTP and have it route packets based on HTTP information that it can extract from the packet?

PQ 2

- A network element can inspect any of the protocols present in the packet. So, **why not** build e.g., a switch that is aware of HTTP and have it route packets based on HTTP information that it can extract from the packet?
- Expensive! Line-rate HTTP processing requires more memory/cpu. Also requires interpreting the protocol below HTTP (e.g., TCP/IP)
- Higher-level protocols change, often more frequently (HTTP 2.0)
- More software can access/manipulate HTTP content (not just your OS TCP/IP stack). *Requires more robustness/more security considerations.*
- But, it's not impossible! See "software middleboxes" or "network function virtualization"

PQ 3

- You plan to disrupt the RPC concept by not only sending arguments to the remote procedure, but also sending the *procedure* itself (as a lambda fn). What challenges do you expect with this idea?

- Breakout rooms + 5 minute discussion with your peers

PQ 3

- You plan to disrupt the RPC concept by not only sending arguments to the remote procedure, but also sending the *procedure* itself (as a lambda fn). What challenges do you expect with this idea?
- Breakout rooms + 5 minute discussion with your peers
- Discussion:
    - RPC server env may be rather different than the caller: lambda fn may be in a lang that can't be interpreted by the server
    - Env typical RPC: addresses/memory
    - Do you pass a pointer to fn, or an object? How do you wrap the code in a way that makes it standalone on the server. Need to capture all you need.
    - SECURITY! Fn needs some kind of sandbox (docker? VM?)
    - State — do you let the function store state somewhere?
    - Remote code execution — always bad? … browsers!
    - Javascript… it sort of does exactly what's above!

PQ 3

- You plan to disrupt the RPC concept by not only sending arguments to the remote procedure, but also sending the *procedure* itself (as a lambda fn). What challenges do you expect with this idea?
    - Defining invocation semantics: at most once/at least once. What happens on failures?
    - Defining the capabilities of the procedure: can it read files? Can it open connections/send data?
    - Can it be STATEFUL!? Where do we store state?
    - Related: defining allowable side effects, if any
    - Guaranteeing determinism (procedure should probably run identically regardless of host server). Have to determinize calls to random, env state like files. Have to provide environment that is identical across machines (e.g., runtime language-based VM like a JVM).
    - Encoding of arguments (as in RPC)
    - Encoding of the procedure + env state that it needs besides args

PQ 4

- True/False: According to the A1 specification, if the nim server fails mid-game, the nim client will send the same *StateMoveMessage* indefinitely

- PQ4 zoom poll

PQ 4

- True/False: According to the A1 specification, if the nim server fails mid-game, the nim client will send the same *StateMoveMessage* indefinitely

- True! In a sense, the A1 aim client cannot tell if the server is slow or failed.