lecture 14:
Corybantic and Athens:
conflict resolution by voting

5590: software defined networking

anduo wang, Temple University

TTLMAN 401B, R 17:30-20:00

Corybantic

Jeffrey C. Mogul., et al. "Corybantic: Towards the Modular Composition of SDN Control Programs "

SDN promises vigorous innovation

the problem of independent controller modules

- manage different aspects of the network
- competing for resources

Corybantic approach

- modular composition
- coordinate between the modules to maximize the overall value
  - each module optimizes its local objective functions

Corybantic approach

SDN relies on reliable, scalable, and efficient controller software

- arbitrarily complex, central controller

modularity

- to build, maintain, extend

challenge — inter-module interface

- as narrow as possible
- expose sufficient information about local objectives and policies

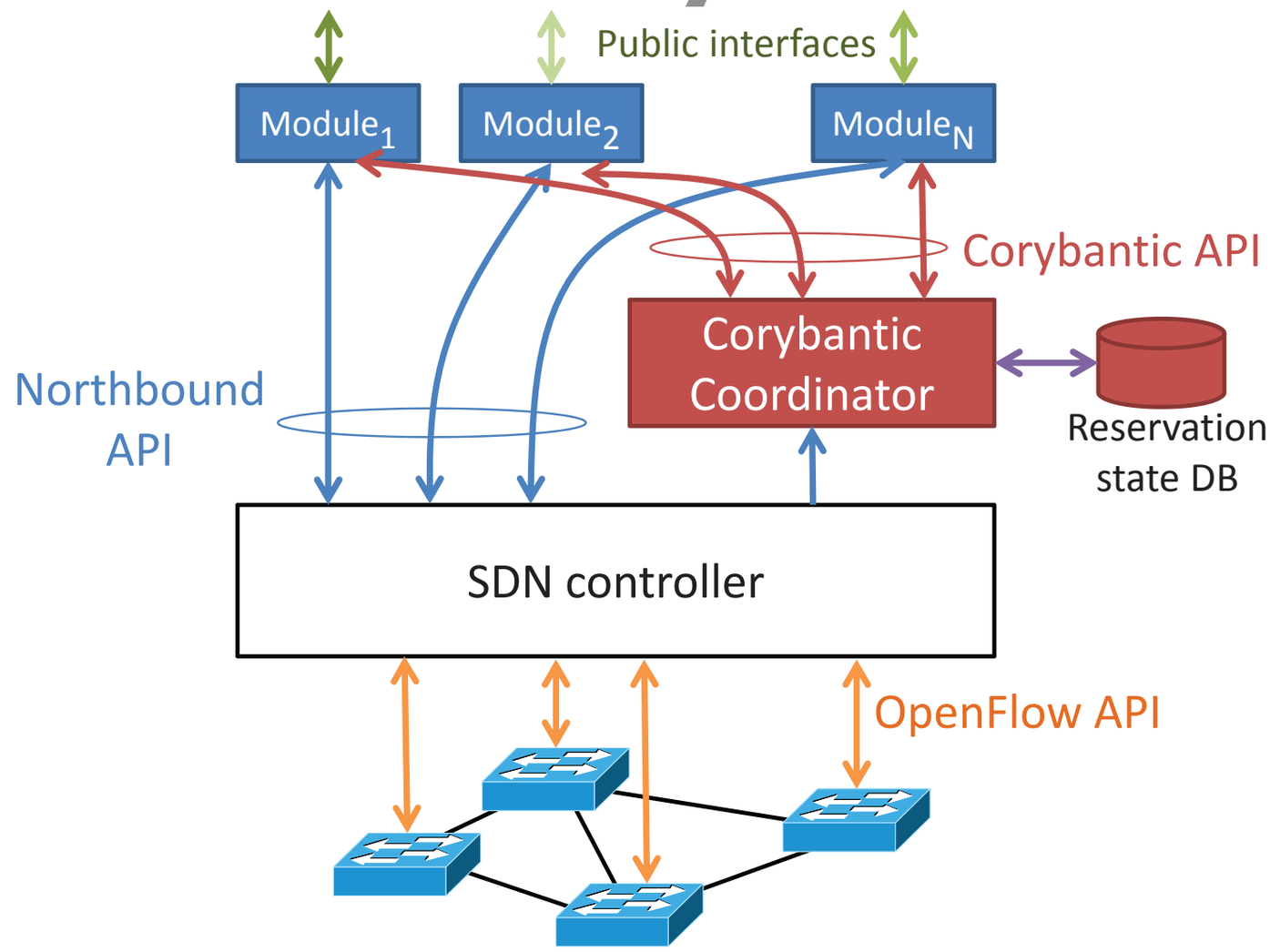
goal: controllers collaborate through Corybantic

- maximize system-wide objectives

Corybantic vs Pyretic

modular composition

conflict resolution



Pyretic

- better ways of getting the network to do …

Corybantic

- deciding what to do

Corybantic approach

modules express local objectives

- using a single currency

sidestep hard problems

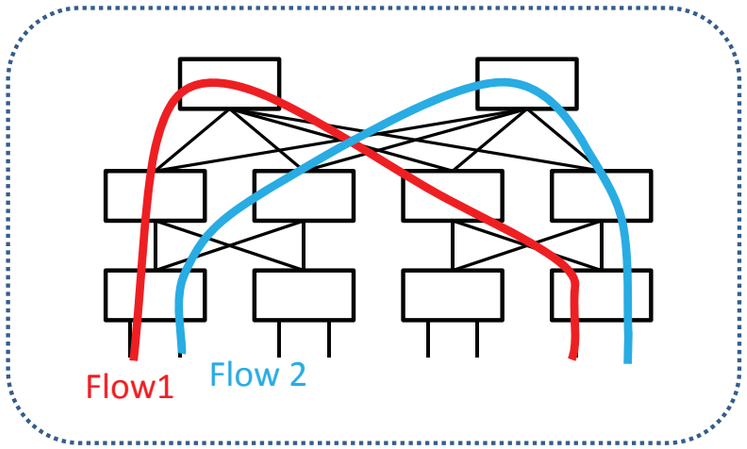
- converts a multi-objective problem into a single-objective one
- use heuristic, iterations to improve allocation decision

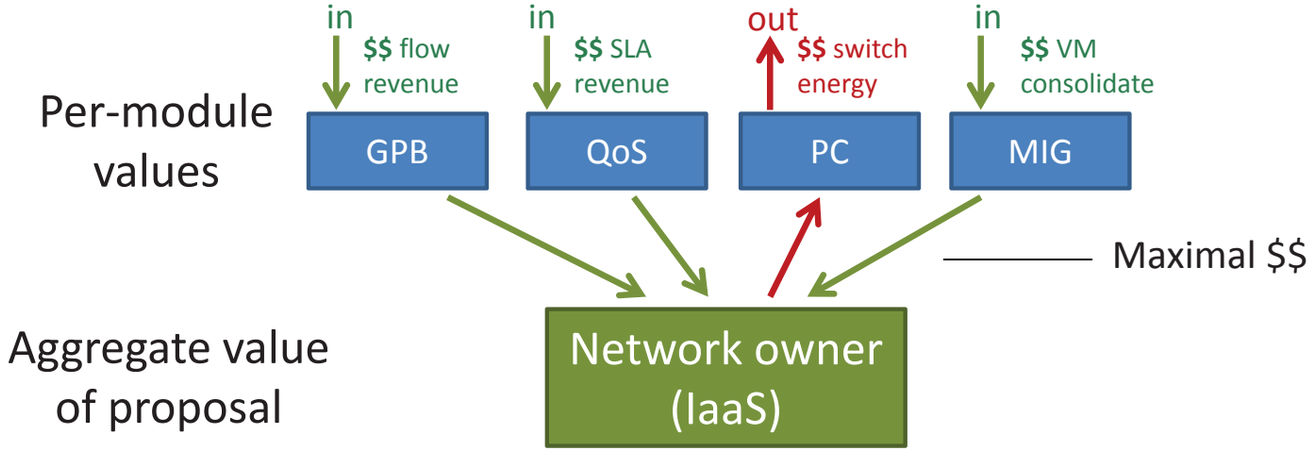
aims at adapting to new demands

- NOT to converge to an optimal solution

Corybantic overview

Accepted proposal

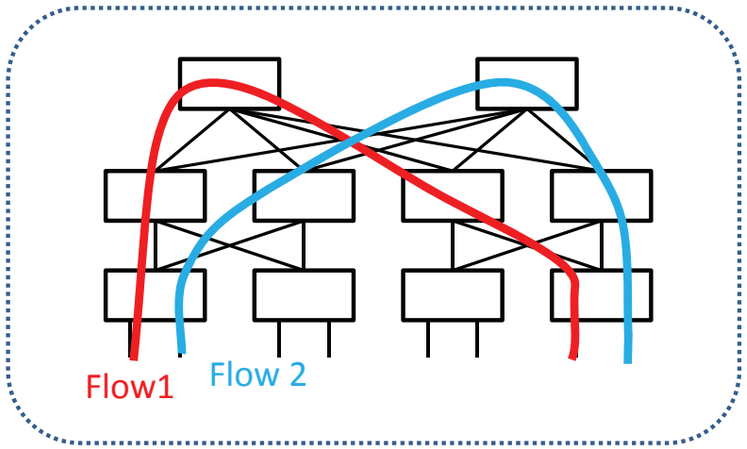


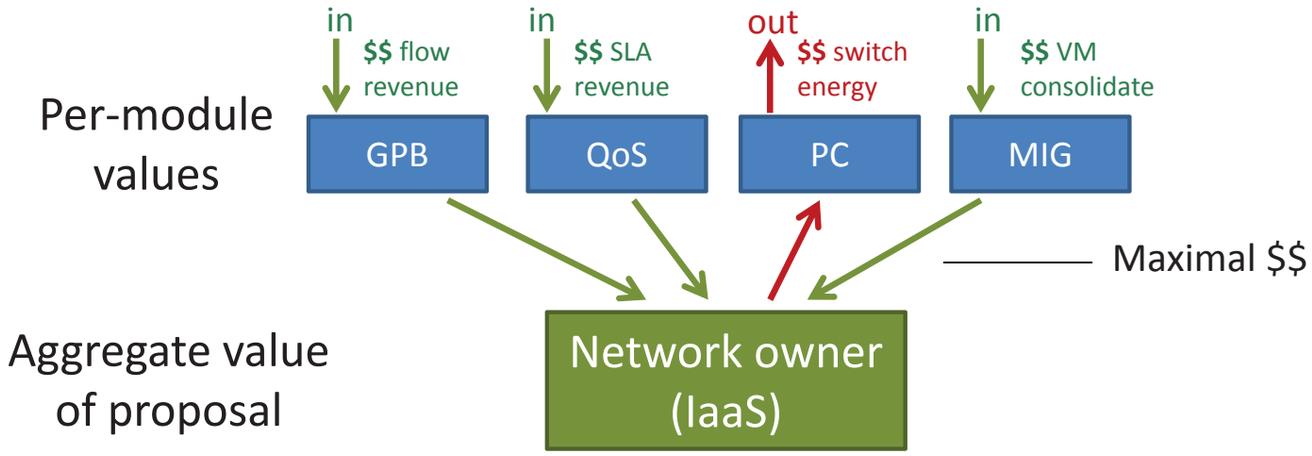


1. modules propose change in a common currency
   - express module objective as virtual subset topology
   - a graph of resources including links and switches

Corybantic overview

Accepted proposal



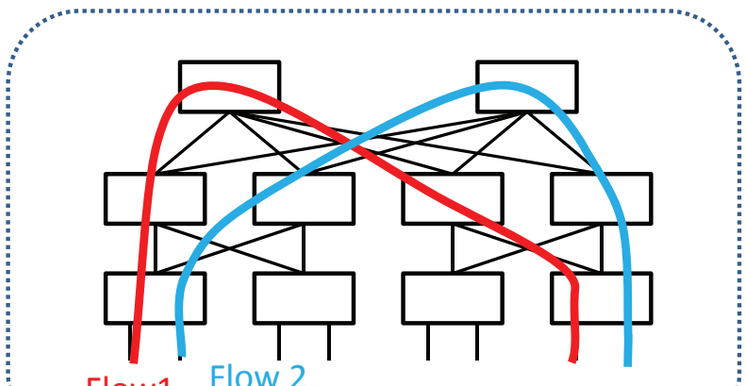


2. each module evaluates every current proposal

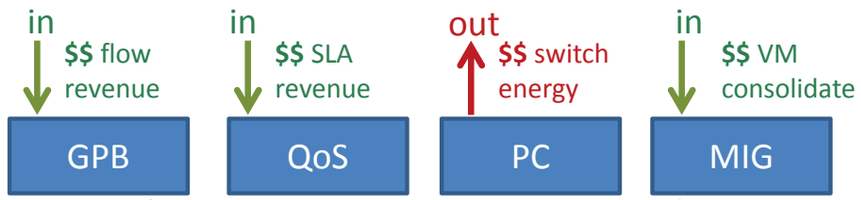
- distributing computation
- proposal generator does not need to understand values of other modules

Corybantic overview

Accepted proposal



Per-module values



Maximal $$

Aggregate value of proposal

Network owner (IaaS)

3. coordinator picks the best proposal
4. the modules instantiate the chosen proposal

open questions

make good proposals

- small?
- # of proposals, variance
- # of interactions

select best proposal — optimality vs. oscillation

- occasional jump (genetic algorithm)
- convex objective function

Athens

resource conflicts

fault-tolerant module (FTM)

- objective
  - maximize the average service availability of tenant's VM instances
- proposal
  - place VMs in isolated fault domains

guaranteed-bandwidth module (GBM)

- objective
  - reserve inter-VM network bandwidth for each tenant's set of VMs
- proposal
  - place as many tenant requests as possible for VM clusters
  - e.g., place each requirement on the smallest network subtree

more on monolithic solutions

simple static policies — prioritizing one module over another *insufficient*

- potential dependency grows exponentially with the number of modules
- untenable for one person by hand

more on alternative composition

Pyretic

- resolving rule-level conflicts in the context of OpenFlow

Merlin/Pane

- manual resolution (by operator) for module-level conflicts

statesman

- loosely coupled
- *BUT*, resolving conflicts without regard to any objective functions / system performance

Athens (revision of Corybantic)

voting mechanism as the abstraction to determine the result of conflict resolution

- voting depends on two module characteristics: precision, parity

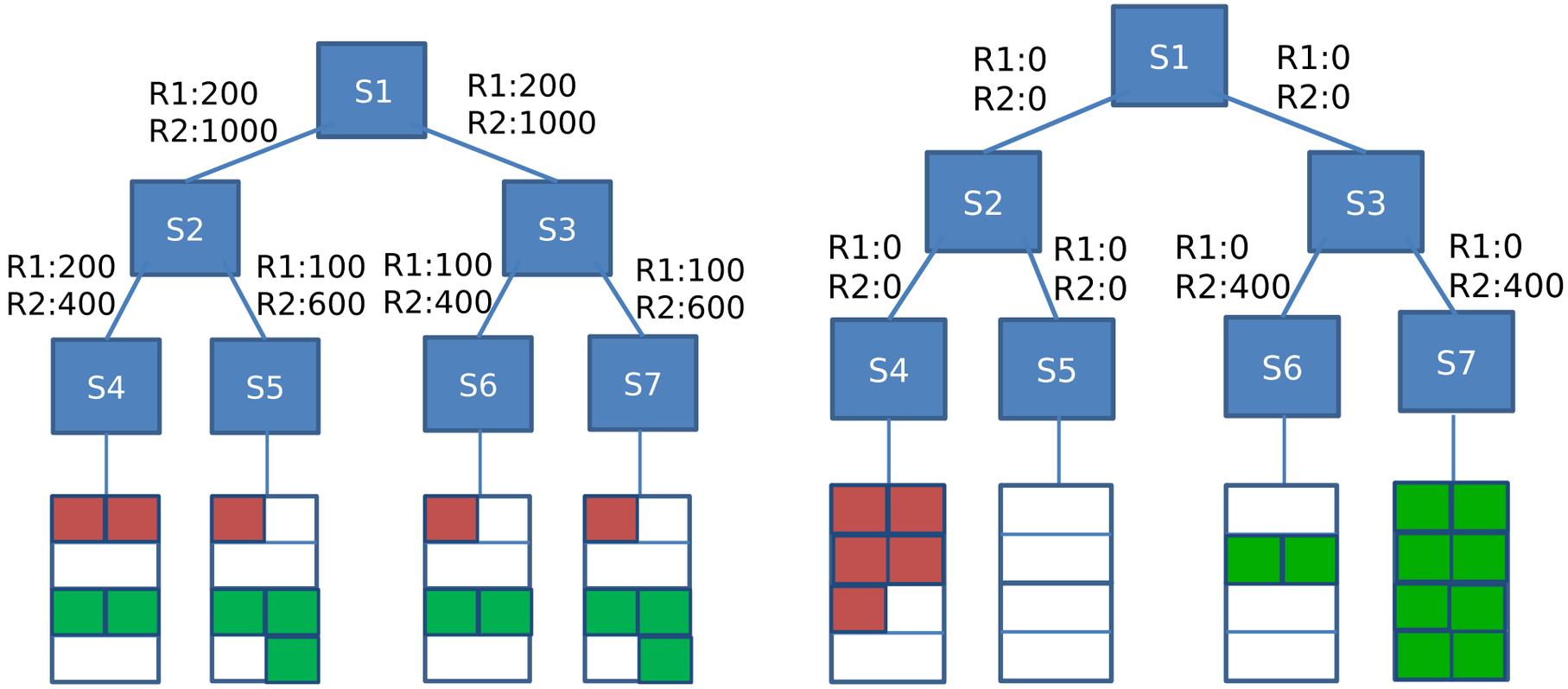
precision

- how accurately a module is able to compare alternative proposals

parity

- how easy it is to normalize the objective functions across modules

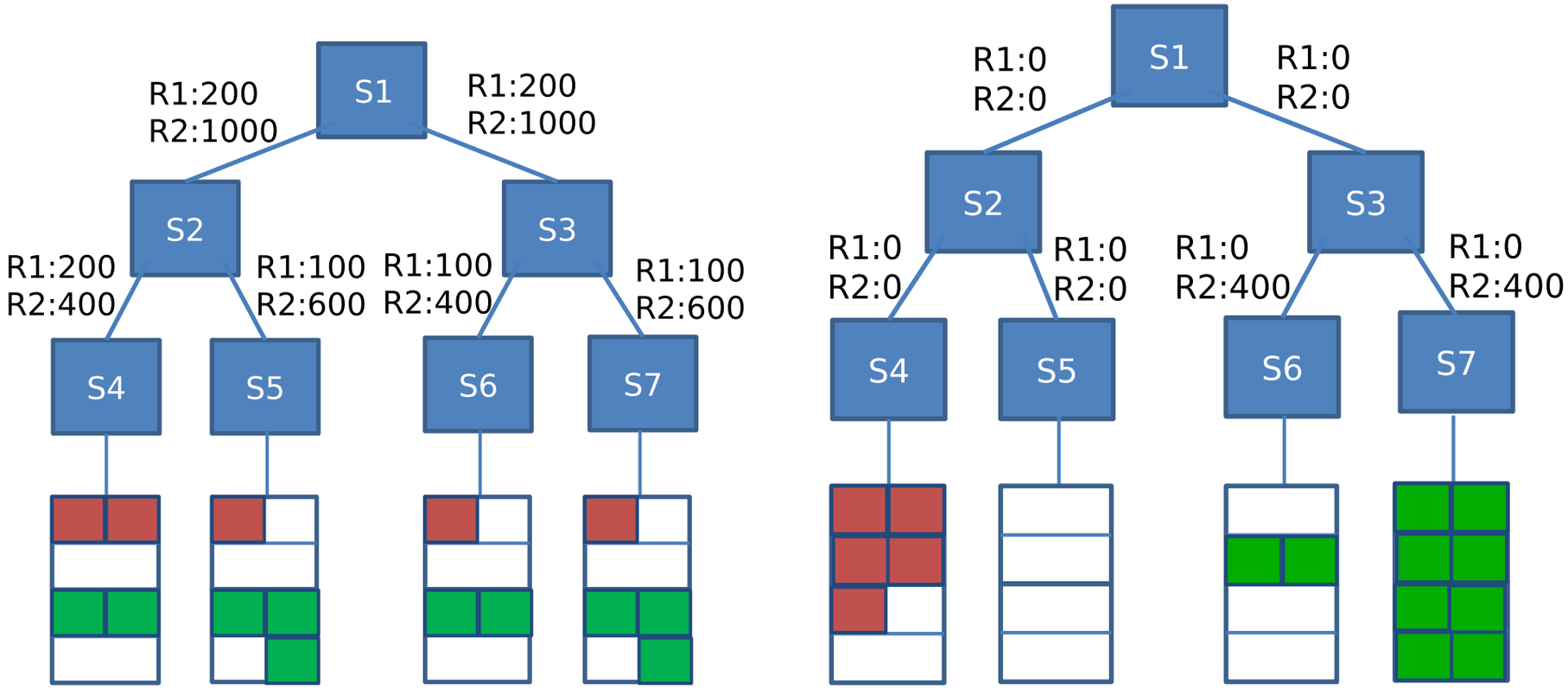
example



(a) Example FTM proposal (b) Example GBM proposal

Figure 2: Proposed network states by FTM and GBM for tenant requests R1: <5, 100 Mbps> and R2: <10, 200 Mbps>, respectively. Red slots are occupied by R1 and green slots by R2. Numbers beside a link show reserved bandwidth on the corresponding link for each request

example



(a) Example FTM proposal (b) Example GBM proposal

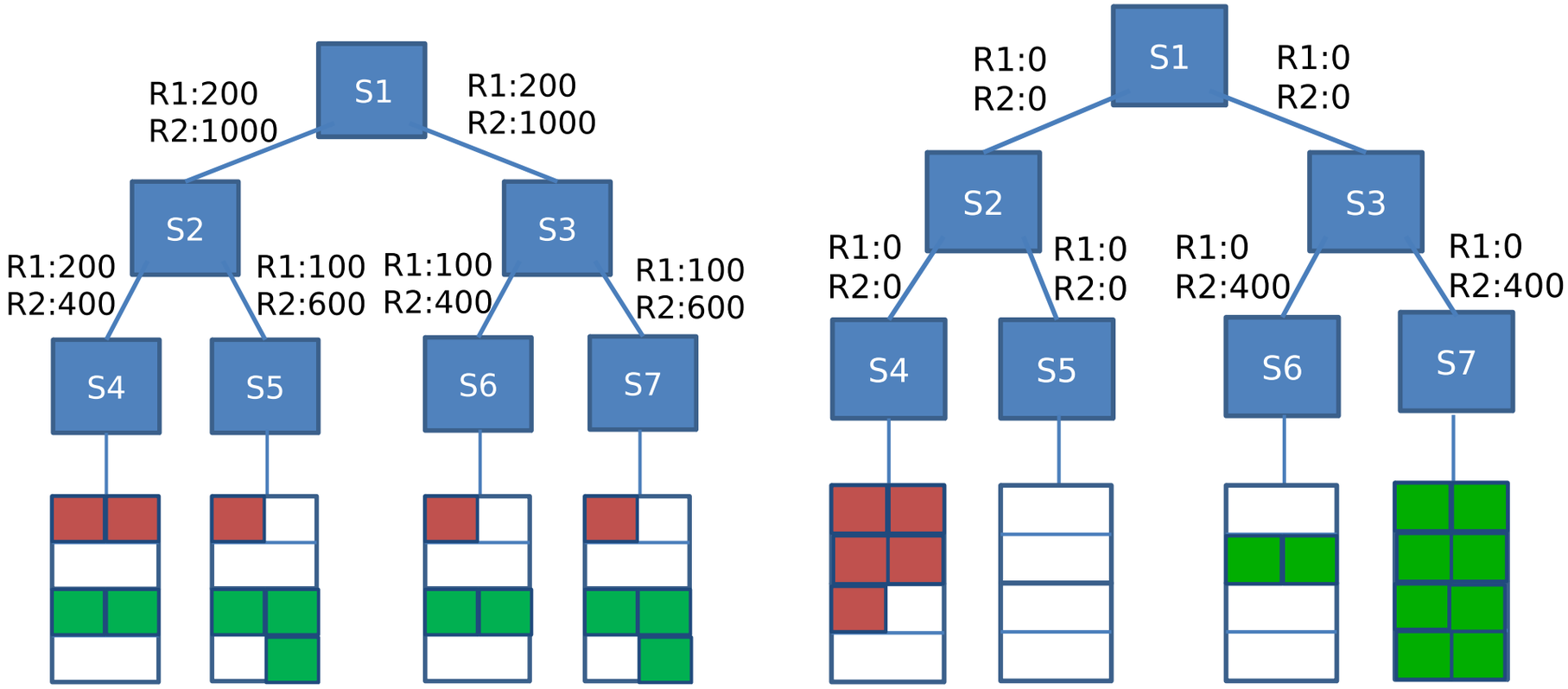
FTM

- spreads each tenant's request across full domains

GBM

- places VMs in the smallest (lowest) subtree

example

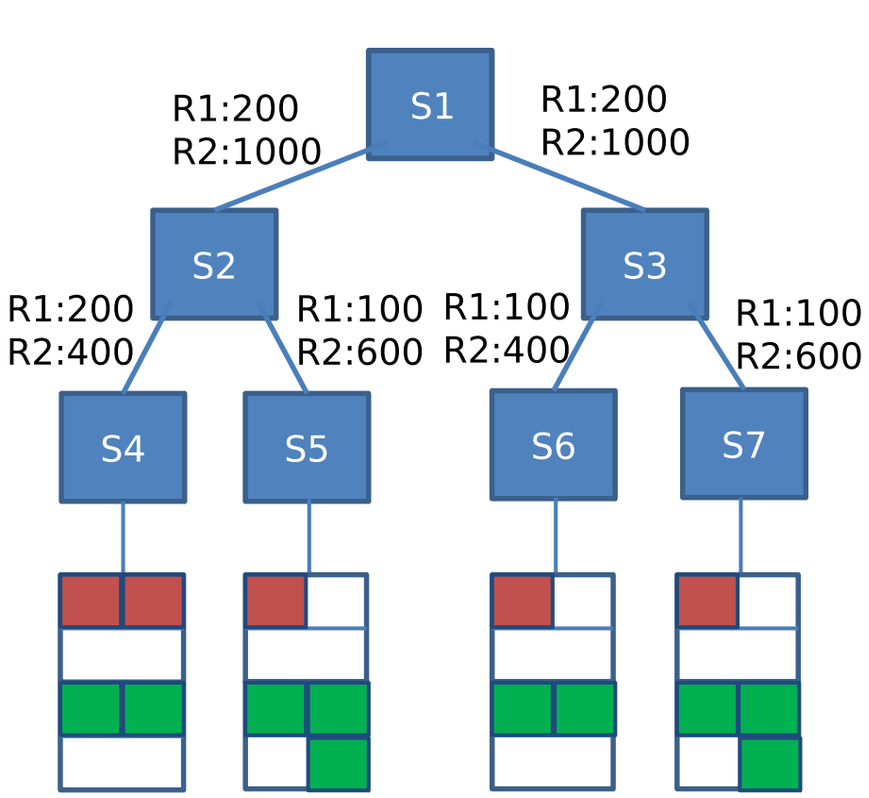


(a) Example FTM proposal (b) Example GBM proposal

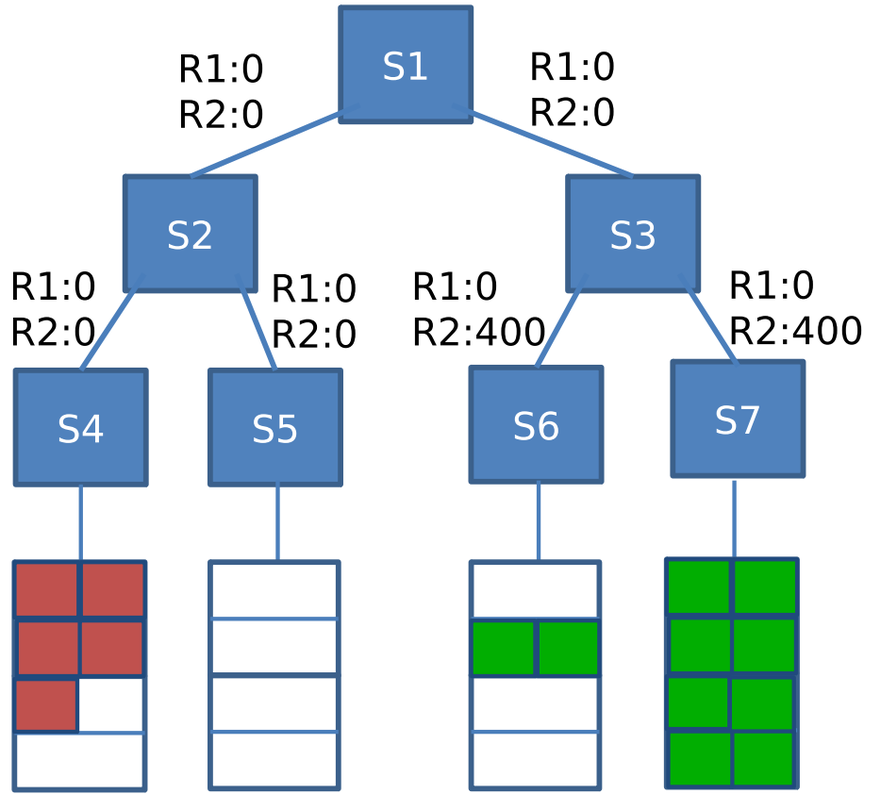
parity across modules

- (implies) their preferences are inherently on equal footing
- i.e., relative ranking are known or can be easily normalized

example



(a) Example FTM proposal



(b) Example GBM proposal

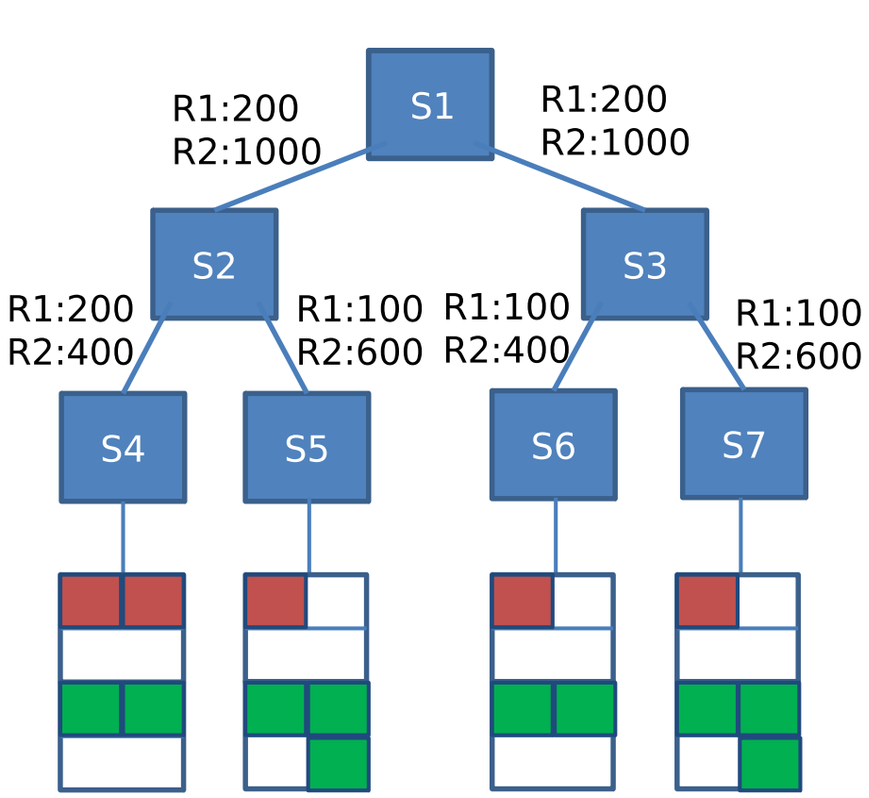
parity in Corybantic

- modules express objectives in a single currency (e.g., dollar)

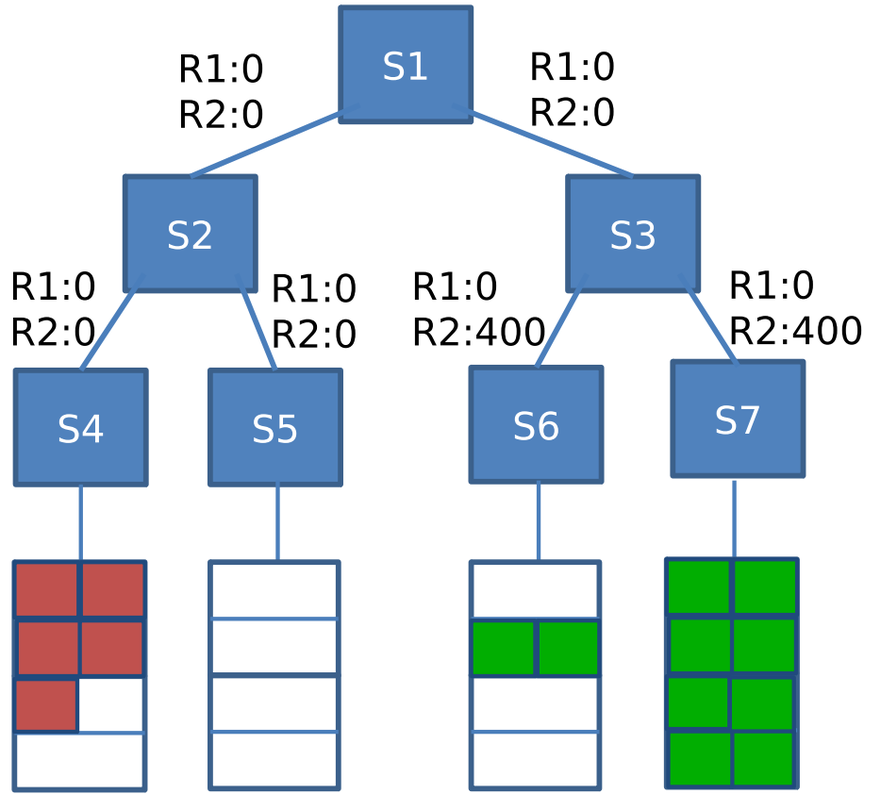
practice

- very hard to relate a module's preference to a dollar amount

example



(a) Example FTM proposal



(b) Example GBM proposal

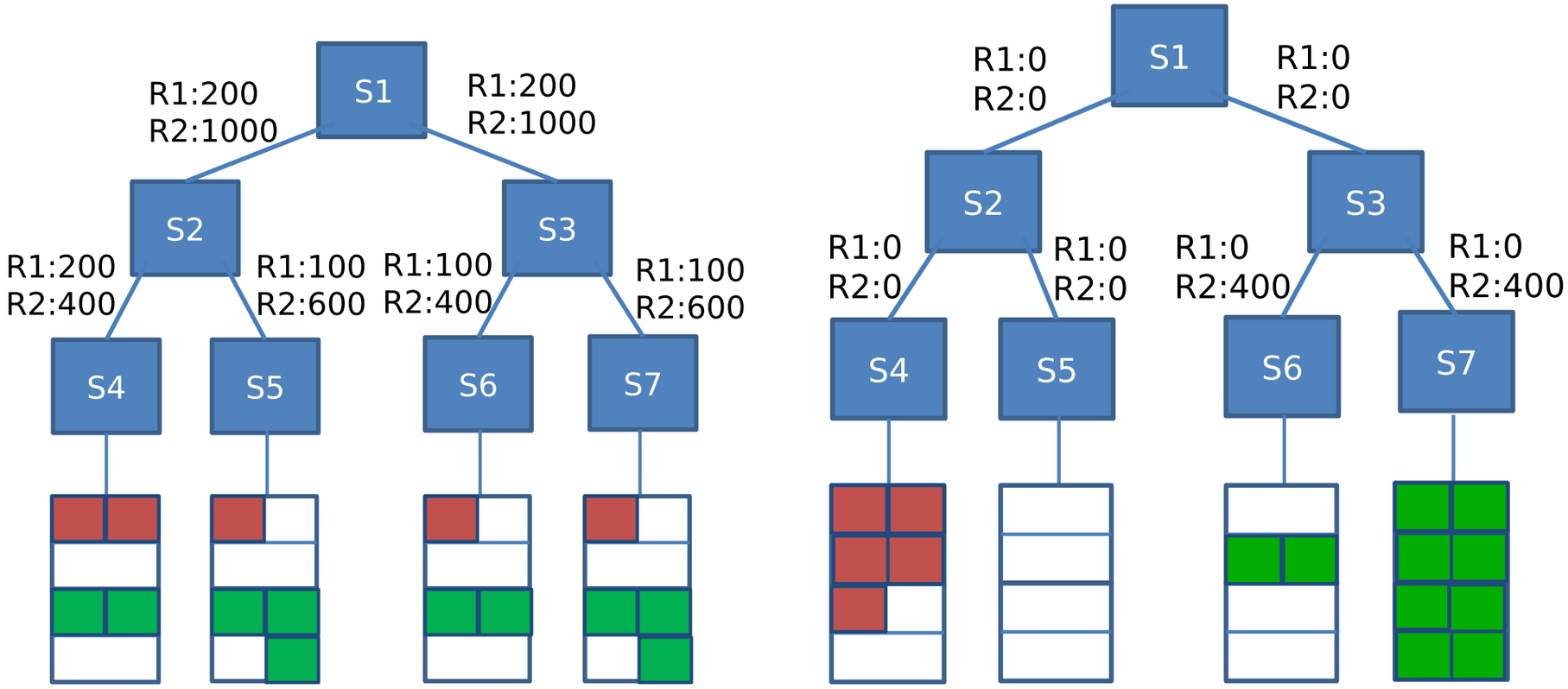
parity in Corybantic

- modules express objectives in a single currency (e.g., dollar)

practice

- very hard to relate a module's preference to a dollar amount

example



(a) Example FTM proposal (b) Example GBM proposal

precision (evaluation)

- FTM
  - evaluate (P1) == 2*evaluate(P2)
  - P1 (allocation) offers twice as much survivability as P2