The BalANS protocol

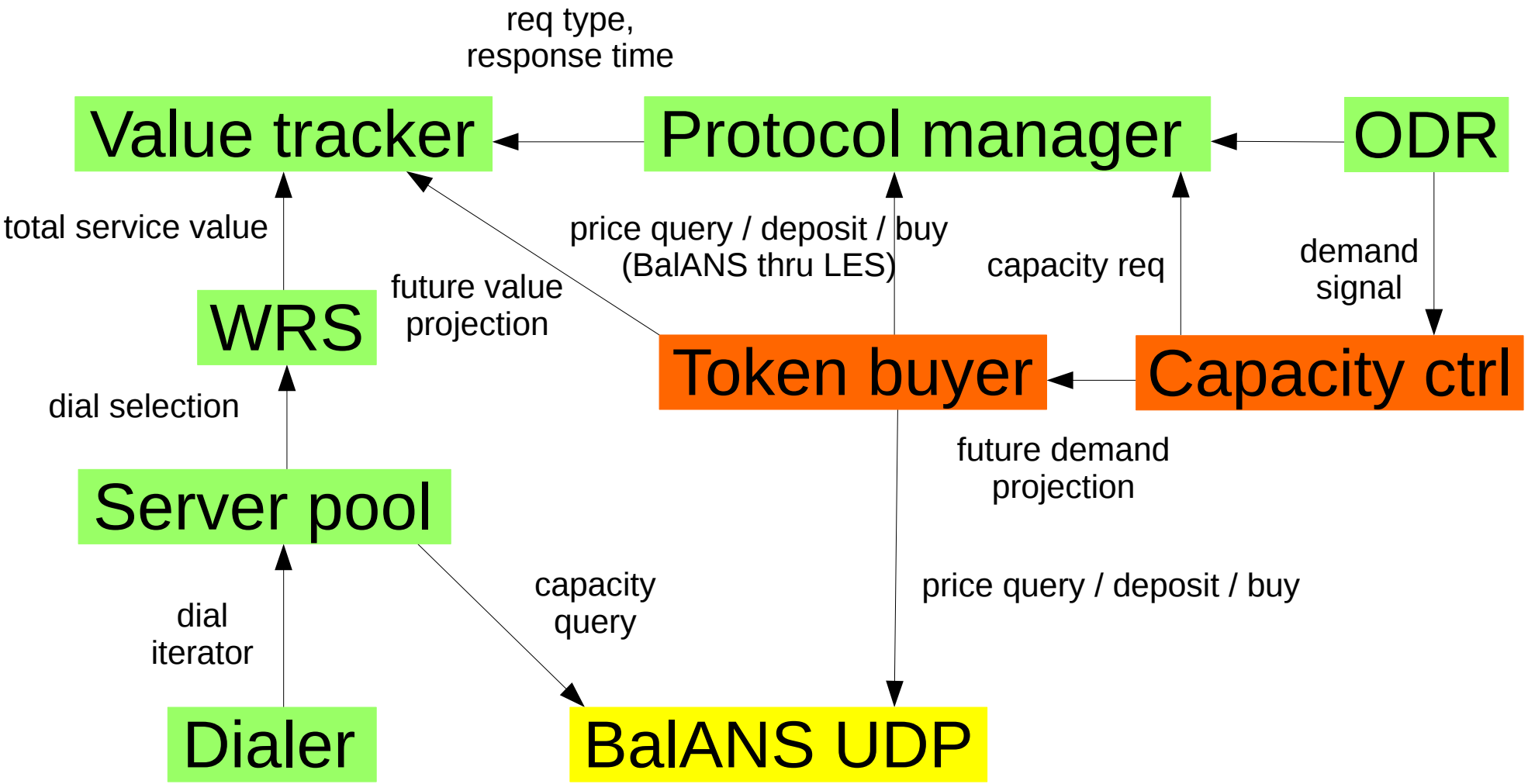
“Balanced Asymmetric Network Scaling”

- implementation and embedding into LES (protocol upgrade, code refactoring)
- decentralized economic/trust model
- smart client strategy
- future plans: more advanced network topologies (sharding, caching, relay nodes)

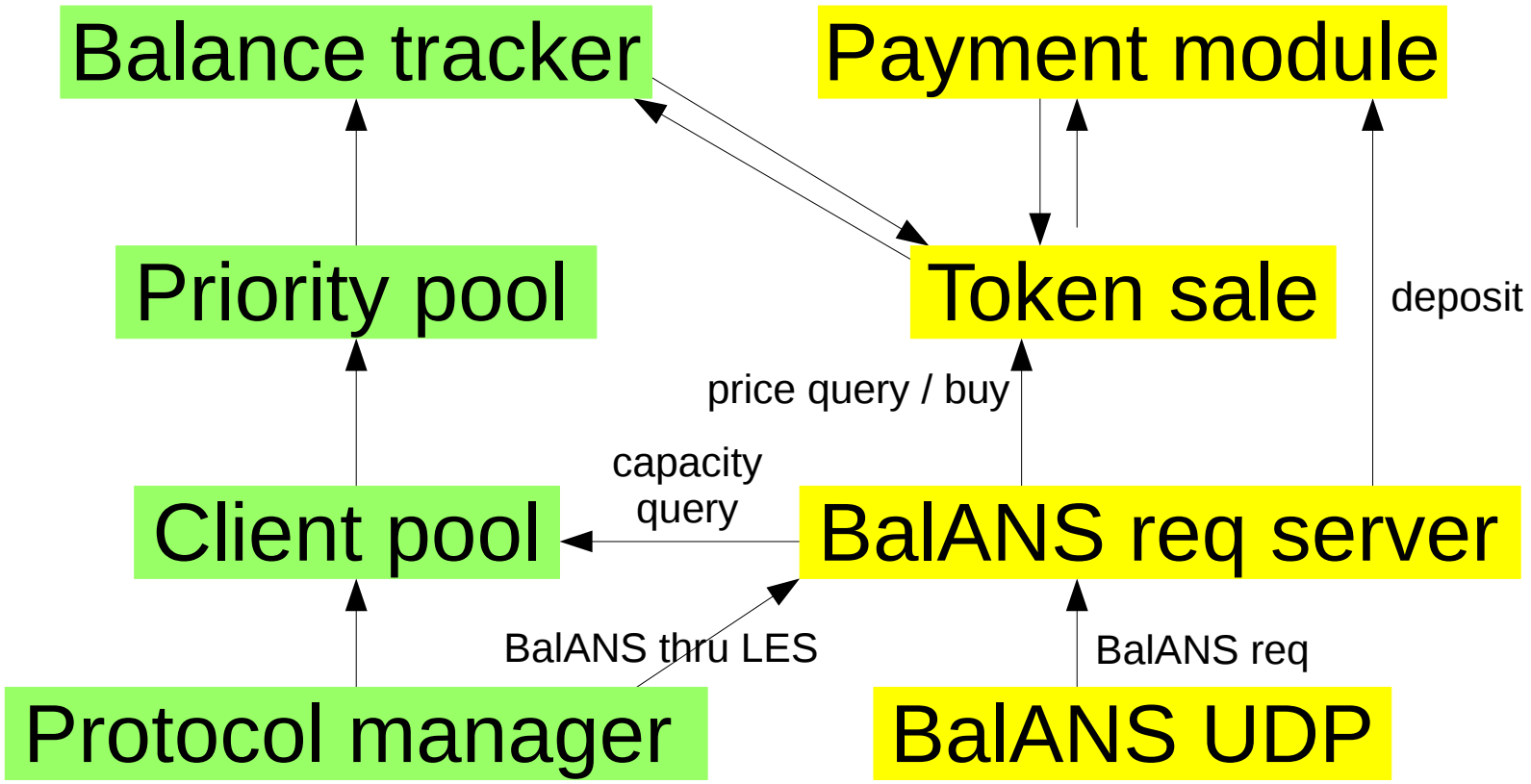
Protocol upgrades

- BalANS protocol (available through both LES/4 and discv5 UDP)
  - capacity query
  - price query
  - deposit
  - exchange
- LES/4 enhancements
  - serve BalANS requests
  - capacity request/update
  - request/reply meta fields
    - reply optionally includes real request cost and token balance (required for paid connections)

Integration and refactoring
(client side)



Integration and refactoring
(server side)



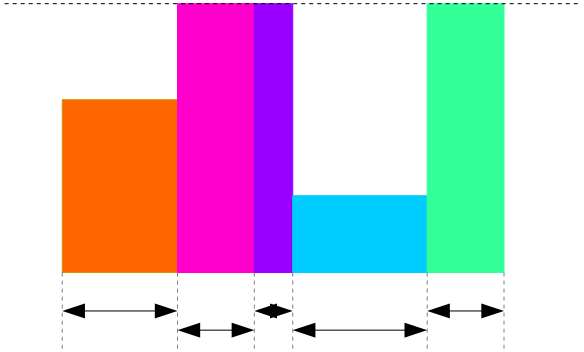
Economic model

"Free market planned economy"

- policies for the short term, market forces for the long term
  - adjustable policy parameters subject to market negotiation
  - two-level scalable priority model
    - capacity assigned to each client, total capacity limited
    - capacity guarantees requests being served
    - pre-purchased tokens guarantee capacity being granted
- "adjusted bonding curve" token sale mechanism
  - token utility is stable, token amount is limited
- flexible pricing
  - price tables based on worst case difficulty estimates
  - server rewards easier than worst case requests with discounts

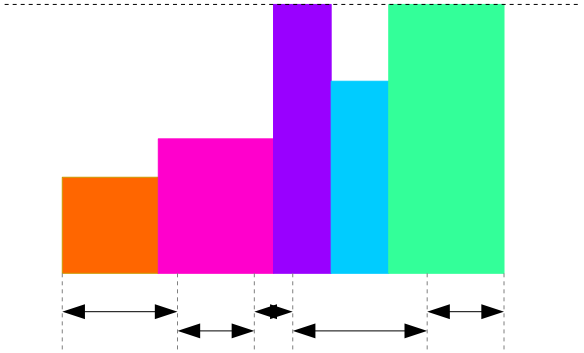
Server policy and token sale

Max capacity/balance ratio
(enforces max total capacity)

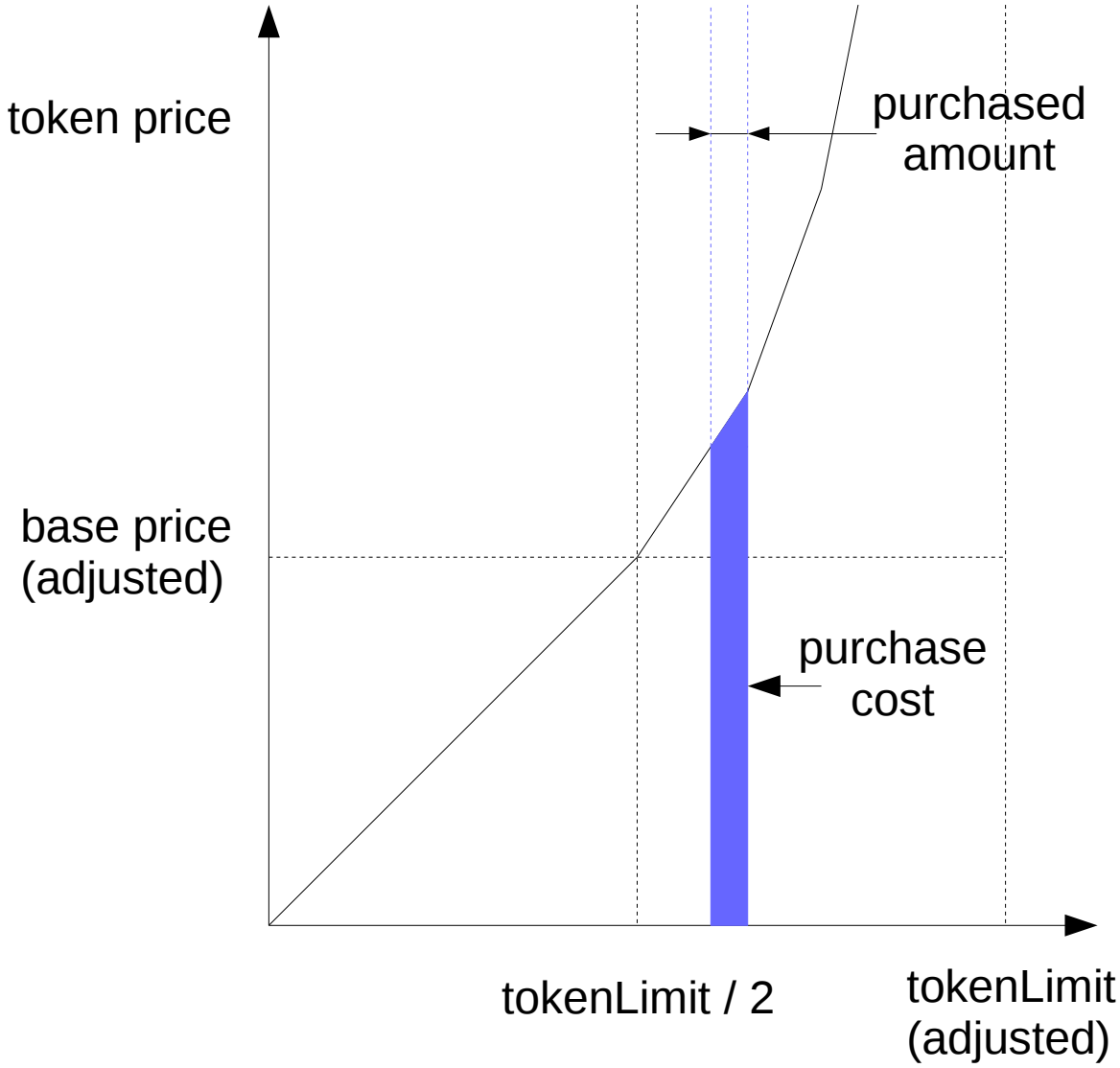


Token balance

Max req rate/capacity ratio
(enforces global req rate limit)



Capacity



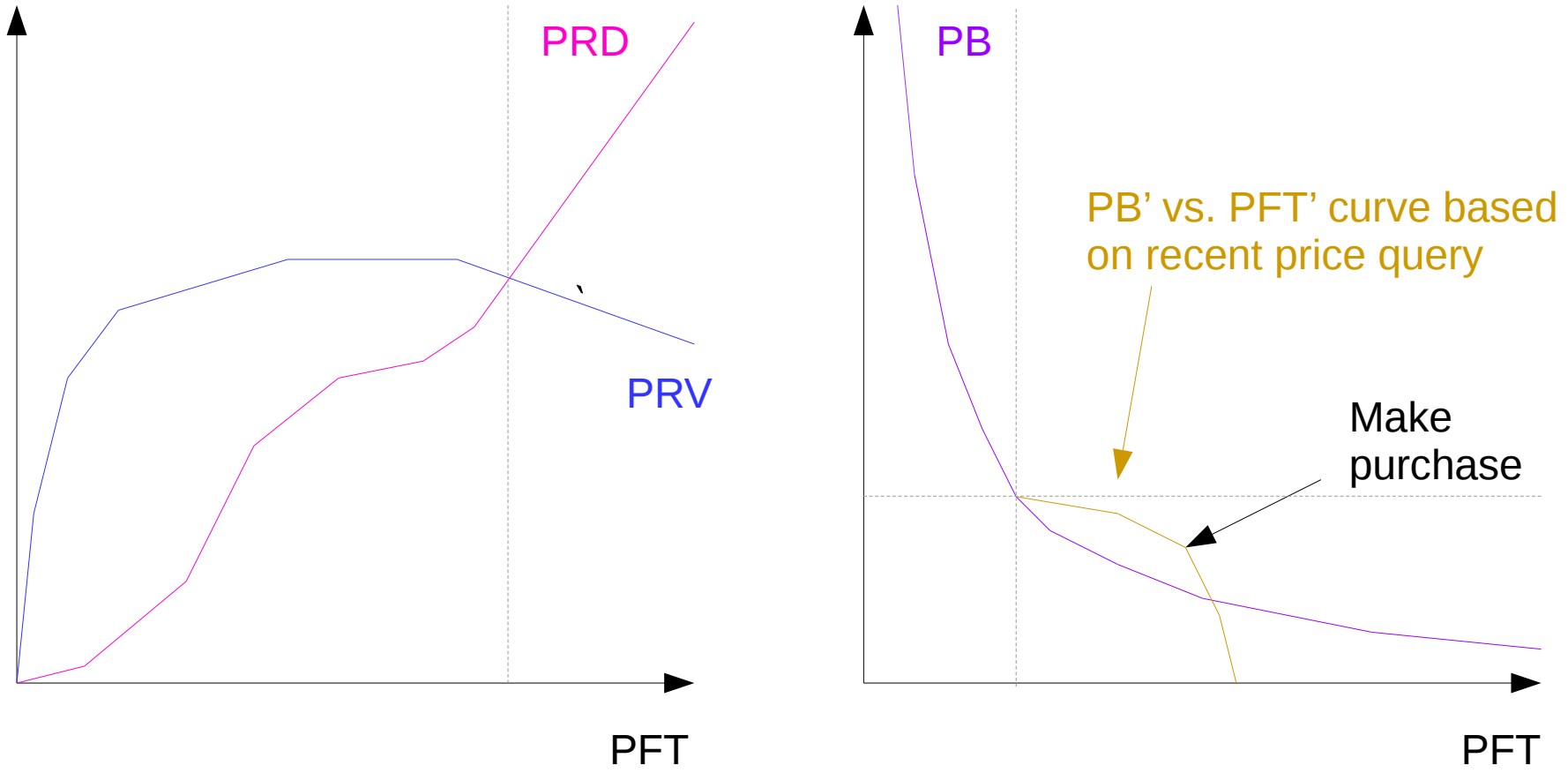
Decentralized trust model

- caveat emptor (nothing is enforceable)
- mutual strategy commitment builds trust
  - trust limits grow exponentially if both parties cooperate
  - kickstarted by free service
- smart clients, dumb servers
  - servers have a simple priority policy
  - clients build a probabilistic model of the known servers based on statistical data in order to optimize their decisions
    - this model predicts service value in the near future based on server policy parameters
  - service as a commodity (instead of a complex enterprise)
    - easy to provide, easy to access, easy to marketize
    - balances information asymmetry, more power stays with the smallest players
    - reduces entry barrier for server operation (risk is a bigger centralization factor than technological economies of scale)

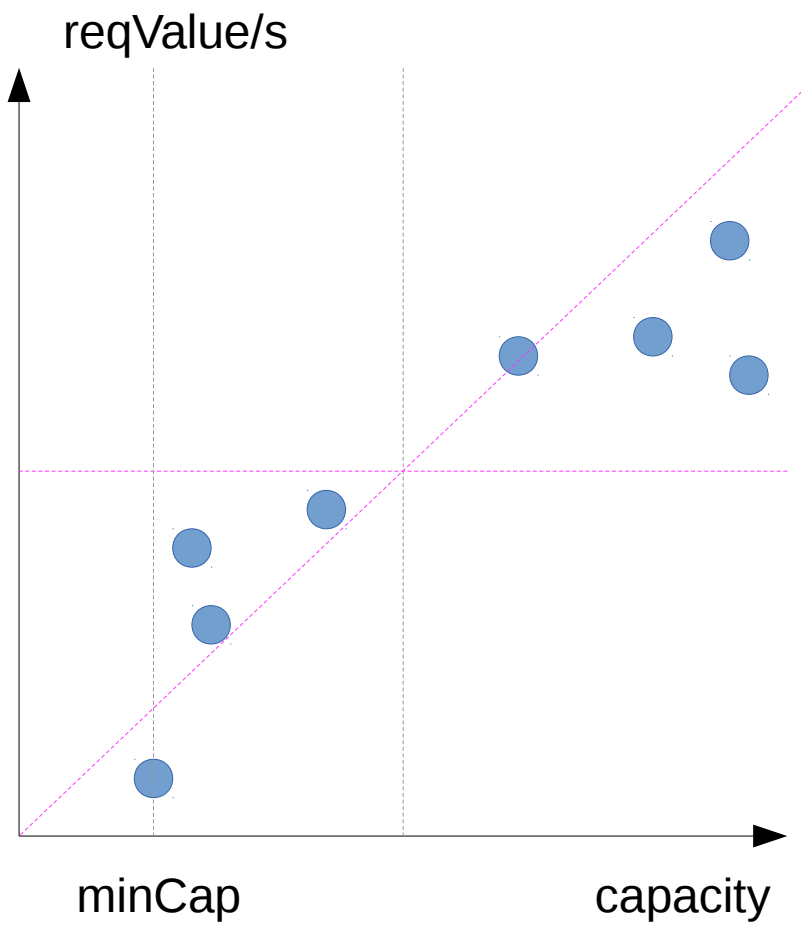
Client market strategy

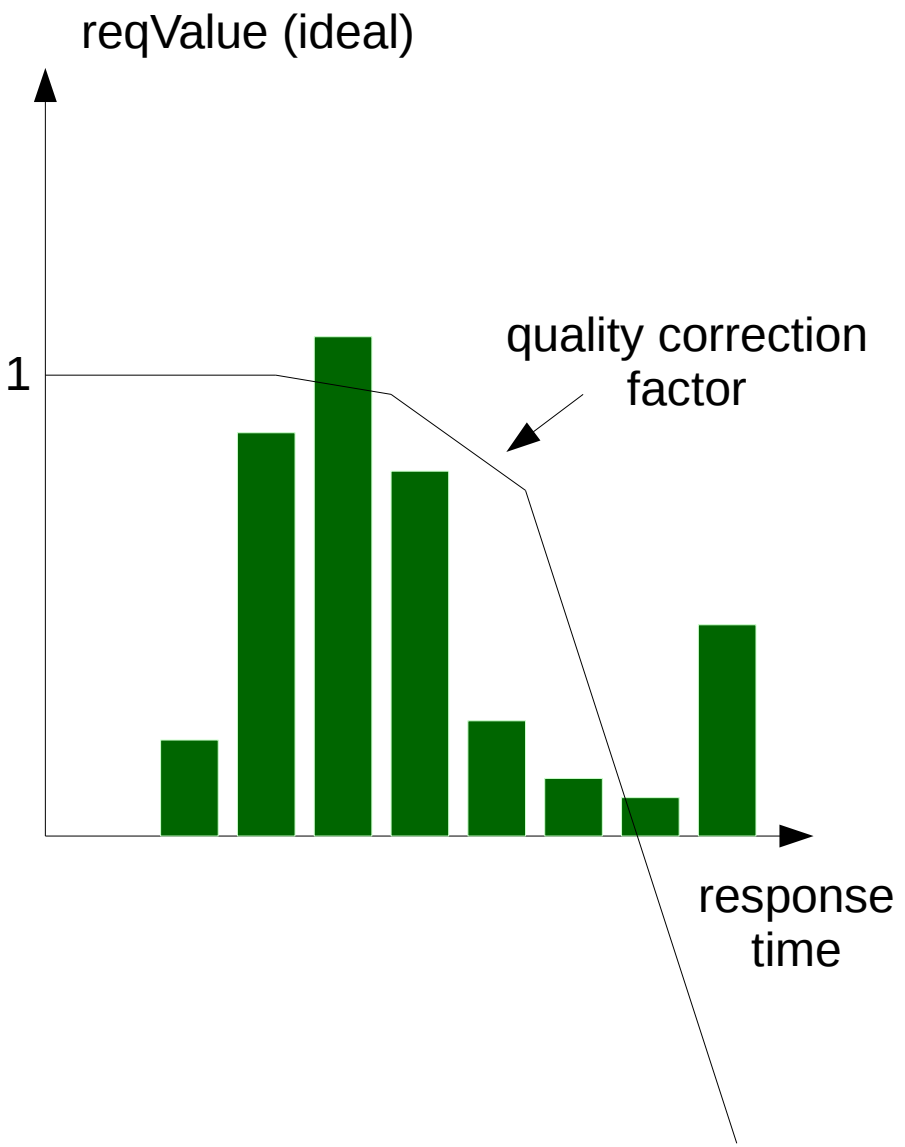
- a dynamic balance between allocated future time and money spent
- projectedTokenValue, PTV(dt): the amount of quality corrected request value expected from all servers if we spend all tokens in the next *dt* time
  - based on token balance, announced request and capacity prices, token expiration speed and capacity-dependent performance correction factors based on past statistics
  - PTV_exp(dt) is the amount of request value lost to token expiration in the above scenario
- projectedTokenDemand, PTD(dt): a rough estimate of future request value demand (simply based on the last *dt* time period)
- projectedFutureTime (PFT) is where PTV(PFT) = PTD(PFT)
- PaymentBuffer (PB) is charged with MaxPaymentRate (MPR, currency/time) if **PFT < FutureTarget && PTV_exp(PFT) / PTV(PFT) < MaxLossRate**
- TokenBuyer makes price and capacity queries, calculates PFT’ and PB’ for different possible token purchase amounts
- A purchase is made if **PB’ * PFT’ > PB * PFT**

Client market strategy



Service quality modeling





Advanced topologies

- flat topology
  - all servers provide the same service (access the entire chain)
- layered topology
  - “service map” allows advertising and looking for access to subsets of the whole dataset
  - some servers provide quick access to a subset of all data (full node on a shard)
  - other servers do request relaying and data caching of data belonging to the entire dataset or a larger subset (light client on many shards)
- “holistic” topology
  - all servers provide access to all data, with different delays on different subsets (at least a light client on every shard)
  - service map adds a “distance” dimension (proportional to maximal guaranteed delay)
  - recommendation service (gives the ENR of a proven efficient node providing a lower distance access to a given subset)
    - practically an advanced, incentivized discovery service based on real performance