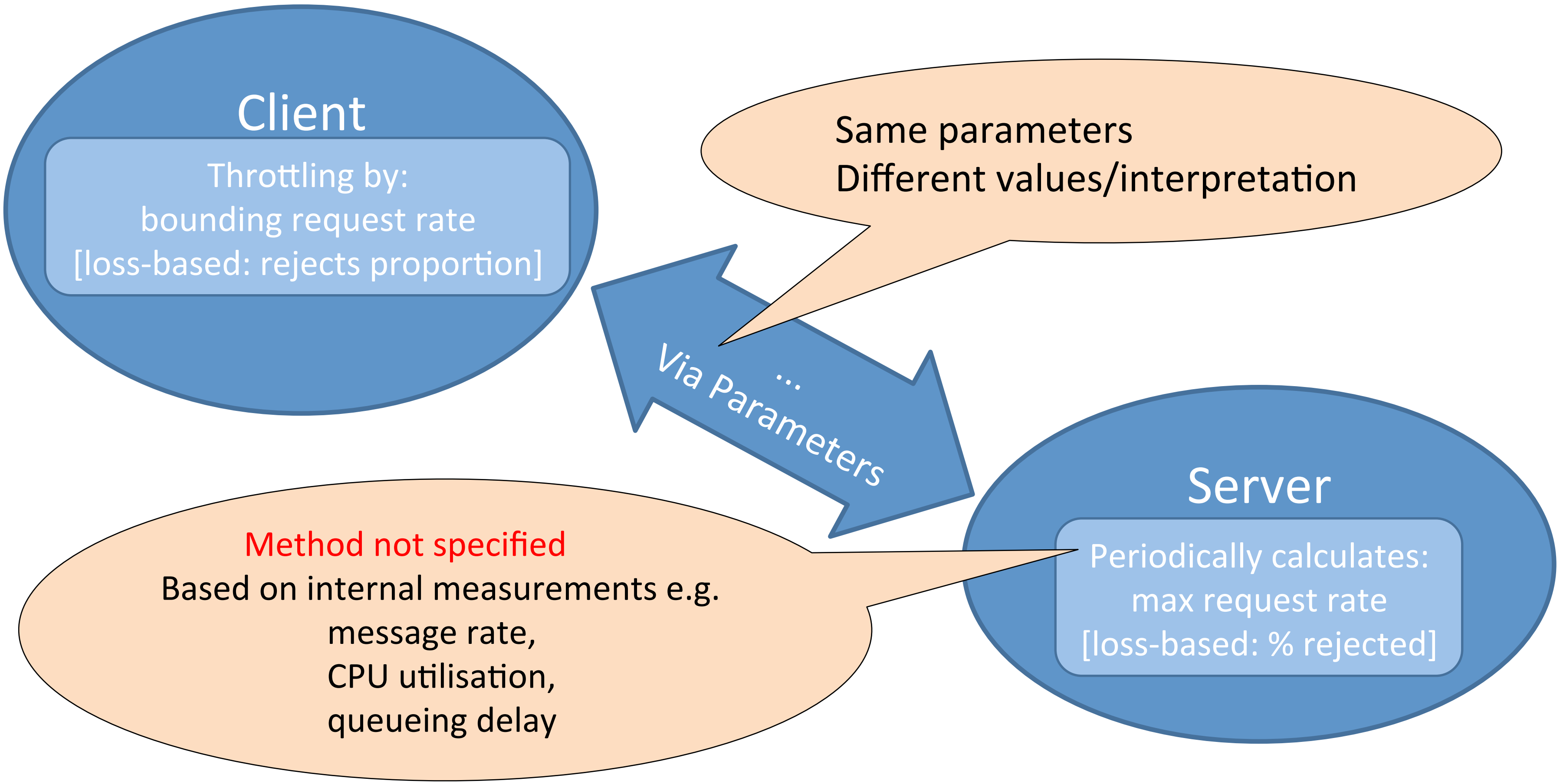
Overview of draft-ietf-soc-overload-rate-control

SOC Rate team:
Eric Noel, AT&T Labs, Inc
Janet Gunn, CSC
Philip Williams, BT

Introduction

- Last SOC meeting IETF81
  - Agreed: submit a rate-based contribution to complement the loss-based control in *draft-ietf-soc-overload-control*
- We propose a rate-based overload control approach
  - mitigates congestion in SIP networks
  - conforms to *draft-ietf-soc-overload-control* signalling scheme
  - draft-ietf-soc-overload-rate-control-01 available

Overview:
Commonality & Differences
loss-based, rate-based



Motivations

Loss-based & Rate-based client algorithms compared

- Behaviour between server updates:
  - Loss-based: admitted rate after control $\propto$ arrival rate before control
    - vulnerable to sudden increases in offered load at client sources
    - cannot guarantee bounded rate towards an overloaded server
  - Rate-based: constant rate bounds
- Deployment in simple/nascent networks
  - Loss-based: fixed rejection proportion not possible
    - although adaptation not difficult
  - Rate-based: static max rates simple
    - not efficient, but can be made adaptive later
- Support for precise capacity guarantees
  - e.g. communication provider boundaries
  - policy easier to realise & enforce
- Penalty: algorithmic complexity?
  - Server: must allocate portion of target offered load to each conversing client
    - max rate may not be attained
  - Client: leaky bucket more complex than proportional blocking
    - but incorporating priorities easy

Client and Server Rate-control Algorithm Selection

Client sends	Server returns
oc	oc = <rateValue>
oc-algo = “loss”, “rate”	oc-algo = “rate”
	oc-validity = <controlDuration>

Key oc Via parameter values

Server assignments		Client action	scope
value (oc)	**value (oc-validity)**		
> 0	> 0	T := 1 / value(oc)	example alg'm
= 0	> 0	reject all requests	rate-based (only)
any	= 0	stop throttling immediately	all (loss & rate)

NB: Other Via parameters are common (not shown)

Server operation overview

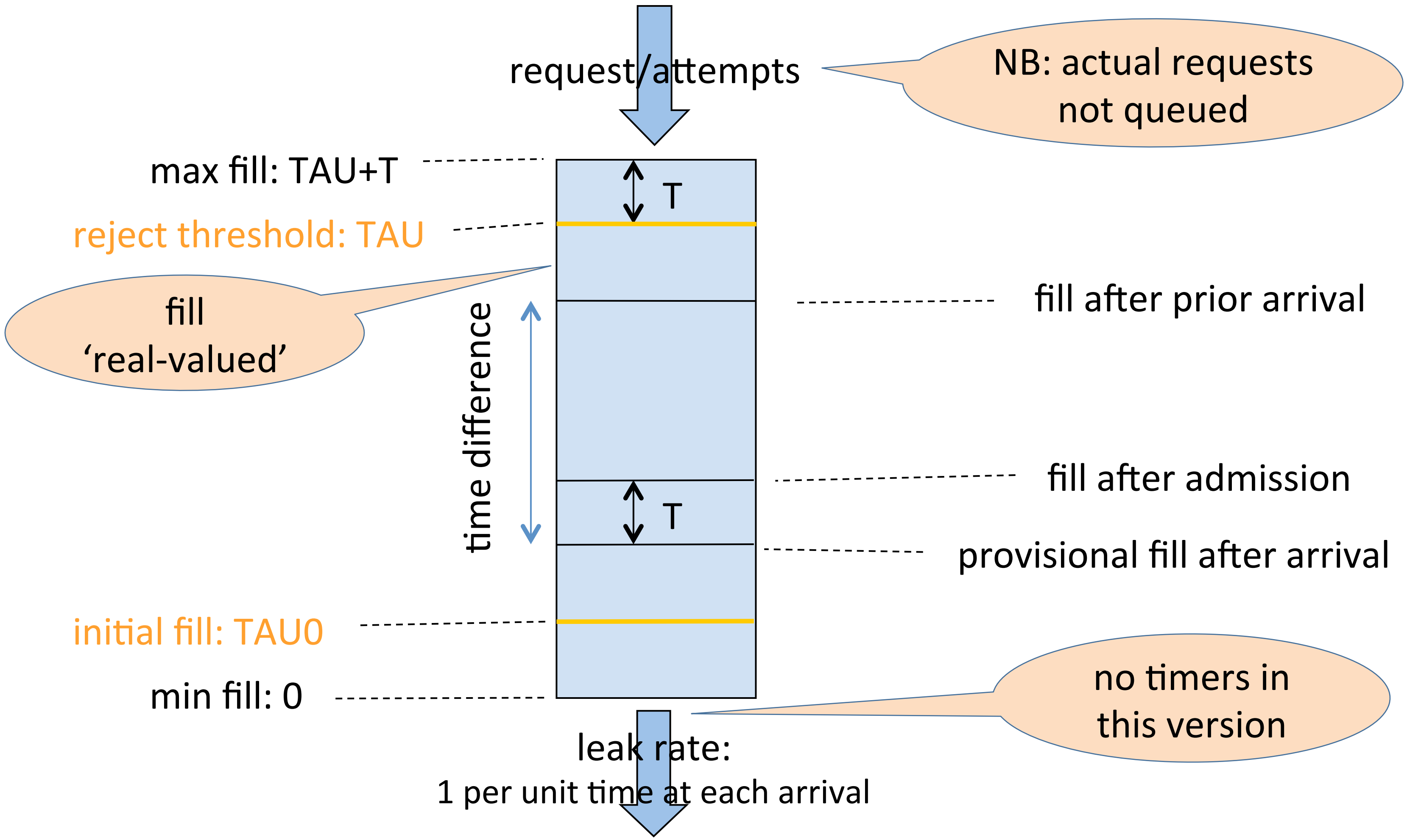
- Server MUST periodically evaluate its overload state and estimate a target SIP request rate
  - to avoid congestion collapse & maintain effective throughput
  - allocate portion of target SIP request rate to each client
    - max rate may not be attained by the arrival rate at the client
    - may be related to capacity guarantees
  - specific algorithm out of scope
- Per draft-ietf-soc-overload-control
  - oc restriction value applies to entire stream of SIP Requests
    - for rate-based: upper rate bound
  - Request prioritization is Client responsibility
    - Server does not know it explicitly
      - but may need to take into account effect this has on the load it receives

Illustrative Client Algorithms

- No mandatory algorithm
- Example Client algorithms included
  - may use others that comply with rate upper bound
- Range of approaches
  - basic scheme
  - priorities: two or more
  - avoidance of resonance

Client operation: basic example

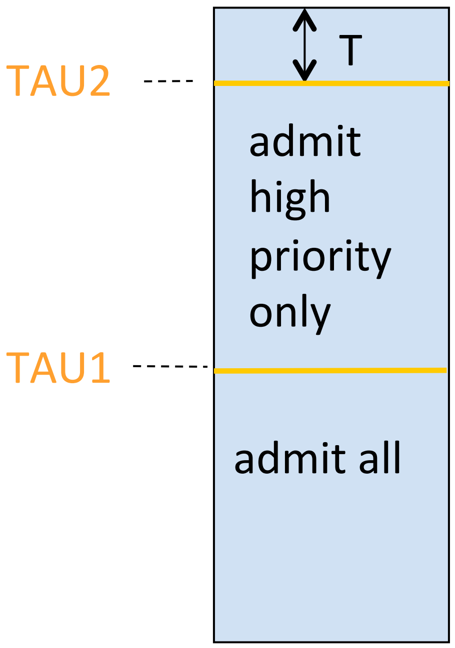
A client default algorithm based on [ITU-T Rec. I.371] Annex A Leaky Bucket algorithm



Priority scheme in client

- Client permitted to prioritize SIP requests based on local policy
  - RFC 5390: requirements
  - RFC 6357: design considerations
  - *draft-ietf-soc-overload-control*: 5.10.1 Message prioritization…
- E.g. two or more categories of requests
  - criteria not specified. Might be
    - Request method
    - SIP URI
    - Resource-Priority header field value

    …..
- Priority may be implemented in Leaky Bucket algorithm using two or more thresholds



Avoidance of resonance

- T becomes larger when:
  - number of client sources of traffic increases and the
  - throughput of the server decreases
- Fill of each bucket can become synchronized
  - e.g. due to traffic surges

  ⇒'peaky' arrivals at the server
- Solution: randomize bucket fill over *T*[1-½,1+½]
  - activation of control
  - admission after bucket empty

Conclusions

- Open discussion on draft-ietf-soc-overload-rate-control