Guaranteeing Access in Spite of Distributed Service-Flooding Attacks

Virgil D. Gligor

gligor@eng.umd.edu

Security Protocols Workshop
Sidney Sussex College
Cambridge, April 2-4, 2003

I. Focus

- Large, Open Networks
  - *public services :* application and infrastructure services (e.g., security, naming)
  - *all clients* are *legitimately authorized* to access a public service

  => *cannot distinguish* the "good" (legitimate clients), "bad" (adversaries), and "ugly" (flash crowds)

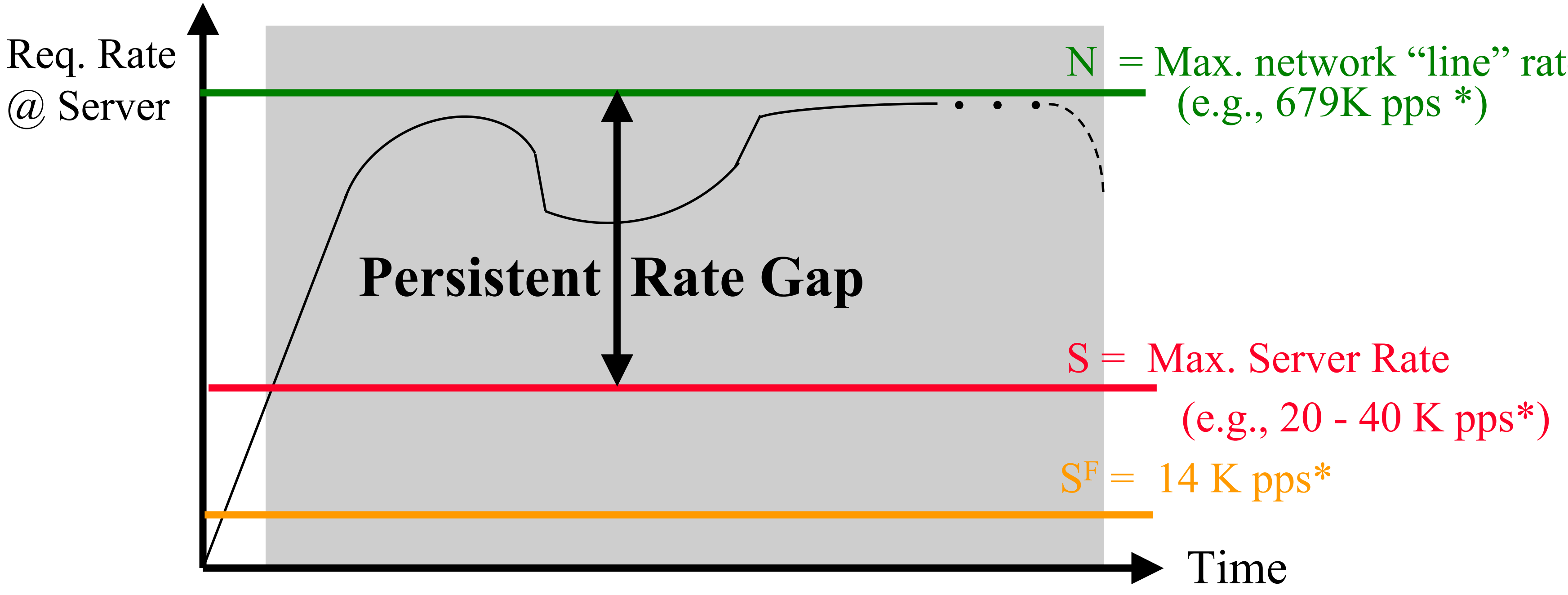
  => *bounds* on number of clients and their capabilities are practically *unknown*

- Flooding Vulnerability of Public Servers
  - persists after *all* other types of DDoS attacks are handled
  - *cause*: E2E Argument => *rate gap* (network "line" rate >> public server rate)
  - *rate-gap persistence/increase over time* => persistent flooding vulnerability
  - economic analogy of service flooding: "tragedy of commons"

- E2E Solution: simple "*user agreements*"
  - behavior constraints: client-server, client-client, or both
  - definition and verification: (1) outside the service, and (2) at "line" rate
  - economic analogy: regulation of resource over-consumption by "user norms"

E2E Solution: Public Service Flooding cannot be prevented by ISPs

- ISPs: no unusual traffic observed in ‘01 cnn, ebay, yahoo! flooding attacks
- **Network economics:**
  - *Public Services : pricing model =/= access model*



* packets per second (Moore, Voelker, Savage, Usenix Security 2001)

* requests (= packet) per second

* firewalls for TCP SYN flood protection

II. GOALS

- **Server Protection -** a necessary but very weak goal
  - Weakest Guarantee: server responds to *some* requests

- **Client Guarantees => Server Protection**
  - *waiting-time bounds for access* to Server
    - scope: *per request*, *per service*
    - bound quality: *variable-dependent, -independent of attack, constant*

  MWT - *maximum waiting time*

  FWT - *finite waiting time*

  PWT - *probabilistic waiting time*

- **Threat:** coordinated service-flooding attacks by
    - an *unknown number* of client "zombies"
    - with *bounded* but *unknown* computational capabilities

Non-Goals:

Protection against "*men-in-the-middle"

QoS guarantees (e.g., aggregate throughput, cost)

Definitions

For *all* client requests,

MWTr – *maximum waiting time* ([IEEE S&P ‘83, TSE’84, ICDE ‘86])
client request is accepted for service in time **T**,
where **T** is known at the *time of the request*.

PWTr – *probabilistic waiting time* ([Millen, IEEE S&P ‘92])
Pr [client request is accepted for service in time **T**] $\geq \theta$,
where **T** is known at the *time of the request*,
θ **=/= 0** and is ***independent* of attack**.

FWTr – *finite waiting time* ([IEEE S&P ‘88])
client request is accepted for service ***eventually***

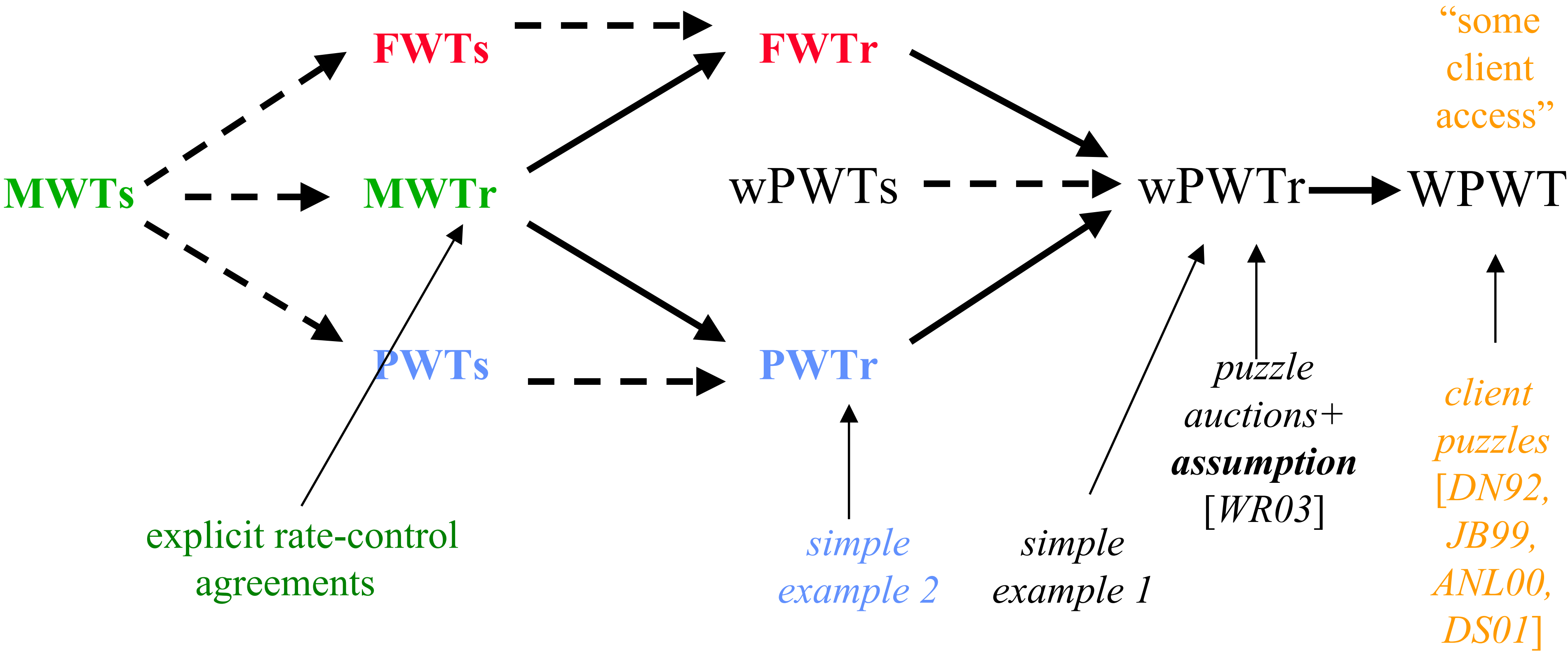
wPWTr – ***weak** probabilistic waiting time*
Pr [client request is accepted for service ***eventually***] $\geq$ **p**,
where **p =/= 0.**

WPWT – wPWT w/o the constraint that **p =/= 0.**

Similar definitions for *Per-Service* Waiting Times:
MWTs (e.g., real time), PWTs (FWTs, wPWTs)
Per-Service Waiting Time => Per-Request Waiting Time guarantee

Relationships among Waiting-Time Definitions

Examples of User Agreements



FWTs

FWTr

"some client access"

MWTs

MWTr

wPWTs

wPWTr

WPWT

PWTs

PWTr

puzzle auctions+ ***assumption*** [*WR03*]

client puzzles [*DN92, JB99, ANL00, DS01*]

explicit rate-control agreements

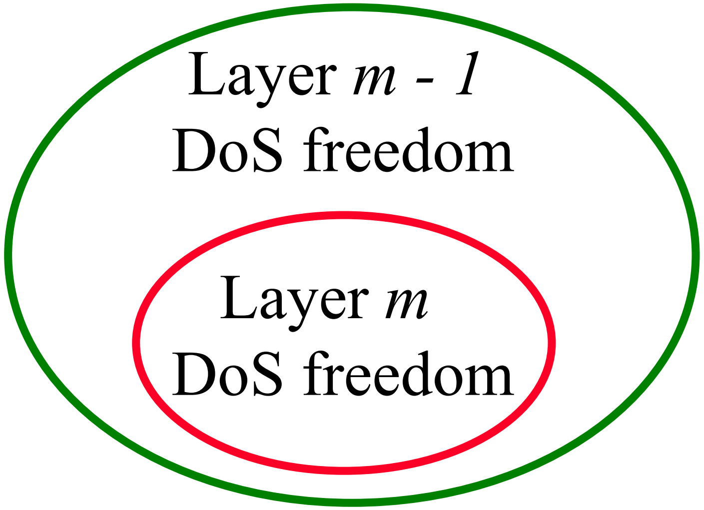
simple example 2

simple example 1

Legend: → ....► = implies

FWTr <=/=>PWTr

General Observations


Layer $m - 1$
DoS freedom

Layer m
DoS freedom

- *layering :*
DoS freedom at layer m-1 cannot be implemented from layer m

(1) DoS freedom at layer m ==> DoS freedom at layer m-1
(not an E2E solvable problem, even if the "Ends" cooperate)

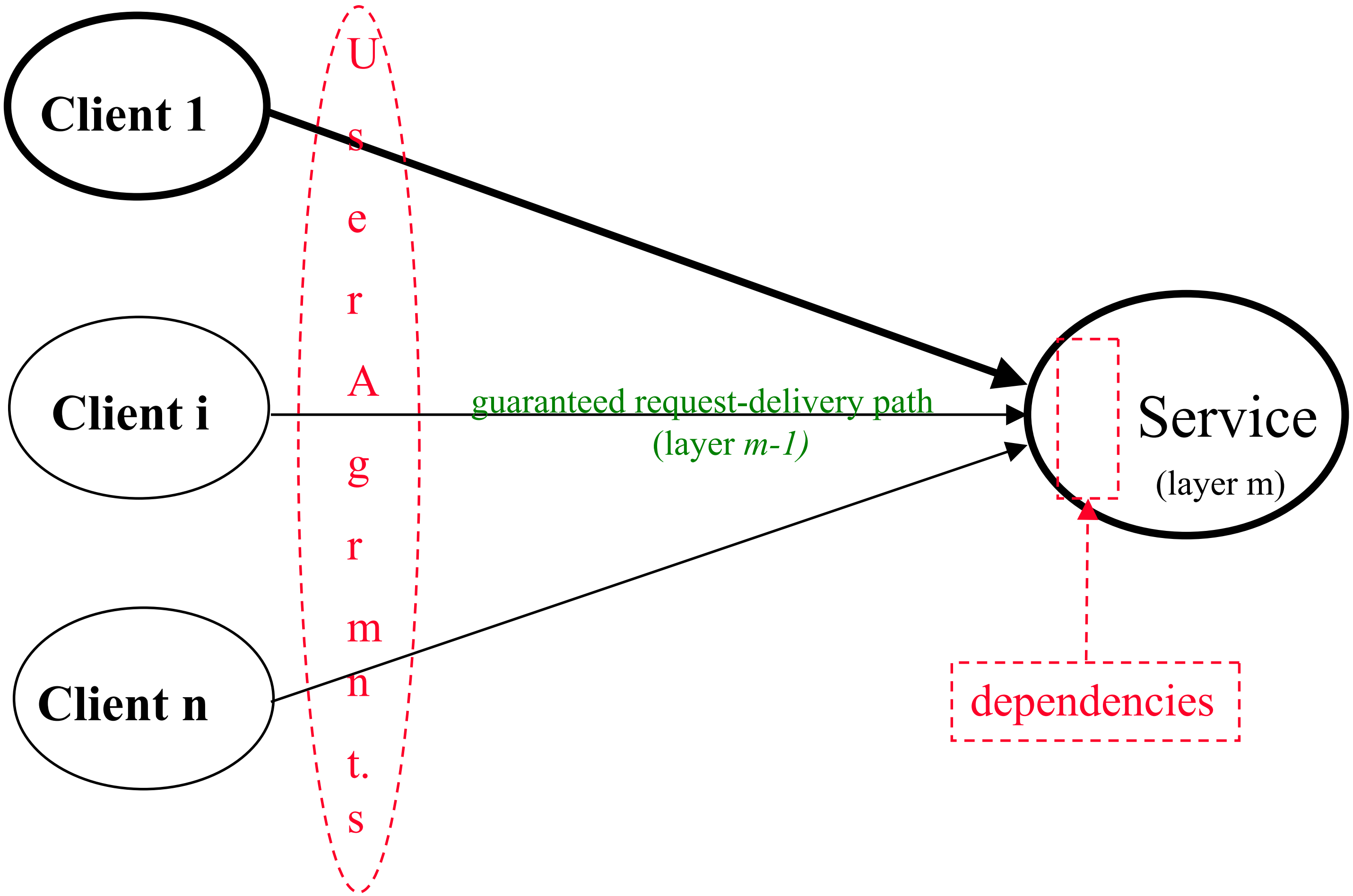
(2) DoS freedom at layer m <=/= DoS freedom at layer m-1
(need a solution for layer m defense *even if layer m-1 is DoS free*)

(3) Solution for DoS freedom at layer m-1 cannot always be replicated at layer m
(likely to need a distinct solution; e.g., no server "pushback" of clients)

Challenge: assuming that layer m-1 is DoS free,
provide a solution that assures DoS freedom to a service at layer m

III. User Agreements

(1) Rate Gap => Undesirable Dependencies among Clients [IEEE S&P ‘83]:
(viz., “the tragedy of commons”)



Client 1

Client i

Client n

User Agrmnt.s

guaranteed request-delivery path
(layer *m-1)*

Service
(layer m)

dependencies

(2) User Agreements [IEEE S&P ‘88] counter undesirable dependencies,

“User-Agreements”

1. Examples in Other Areas

2. What do Client “Puzzles” Achieve ?
   - only that *some* clients get access to the server

3. Explicit Control of Client Request Rate
   - time-slot reservation, total ordering (e.g., a “Bakery Mechanism”)

4. General Request Controls

1. Examples of “User Agreements” in Other Areas

(per user) local state information required

- *binary exponential back-off* agreement for (slotted) Ethernet collision handling
- *splitting algorithms* for collision handling in slotted multi-access protocols
- *two-phase locking* agreement of distributed transactions for maintaining data consistency
- *ordered resource request* agreements for deadlock prevention

global state information required

- *self-stabilization* agreements in distributed control problems
  (e.g., prevent “starvation” in Dijkstra’s dinning philosophers problem)

stateless

- *client-side, packet-filtering*; *pushback* agreements in routers

1. “Client Puzzles” based on Hash Functions

1. *Challenge*: given **k, find X**

   Response: Message **X**

Verification:

h(Message **X**)

$$\underbrace{\boxed{00\ldots0}}_{\mathbf{k}\text{ bits}}\underbrace{\boxed{\textbf{don't care}}}_{\mathbf{m\text{-}k}\text{ bits}}$$

$1 \leq k \leq 64$ $m = 128$

2. *Challenge*: given **k**, **h(X),**

   Response: Message **X**

$$\underbrace{\boxed{00\ldots0}}_{\mathbf{k}\text{ bits}}\underbrace{\boxed{\textbf{don't care}}}_{\mathbf{m\text{-}k}\text{ bits}}$$

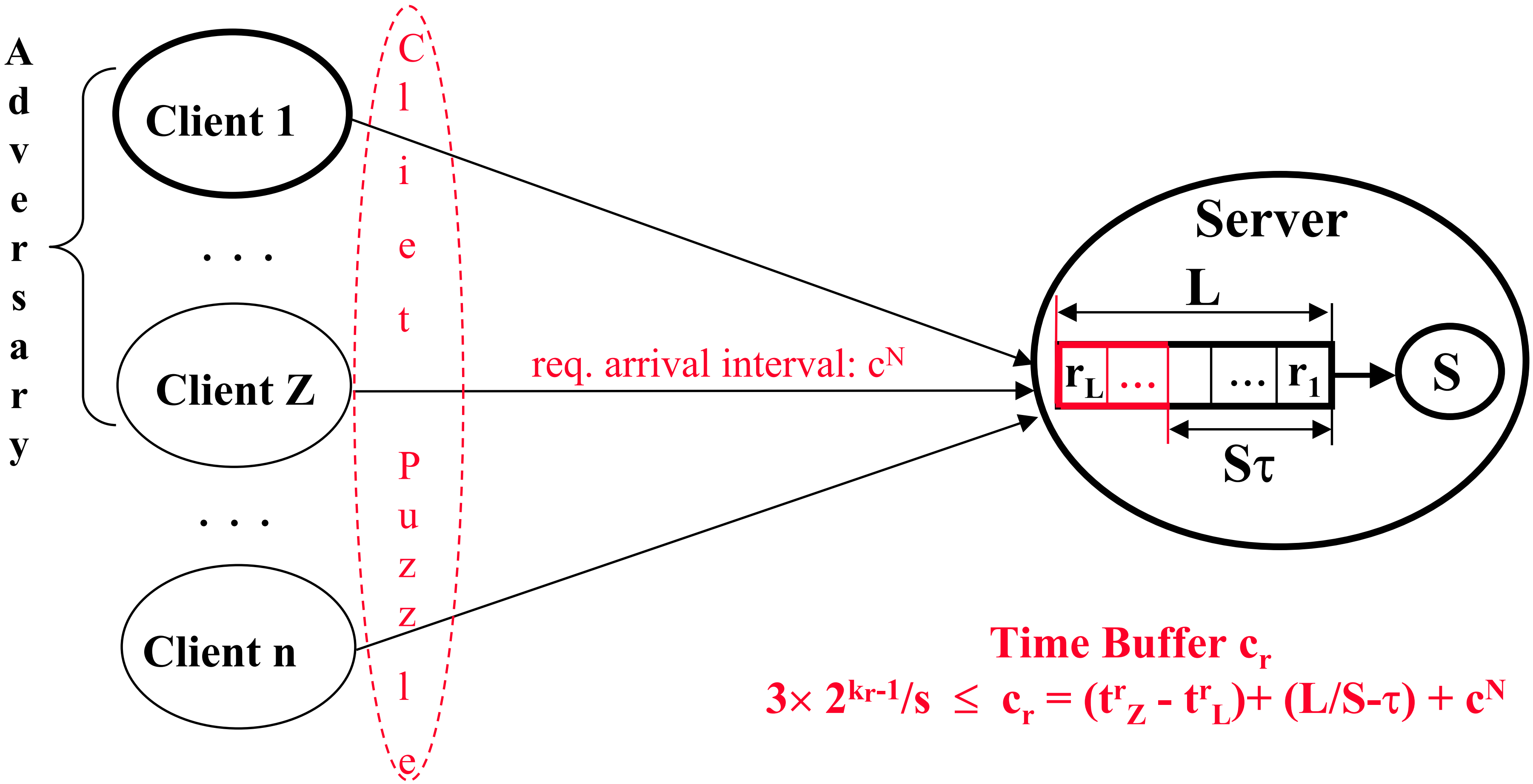
$1 \leq k \leq 64$ $m \geq 512$

Verification:

h(Message **X**) = **h(X)**

Average Latency per Client: 2^k steps

“Client Puzzle” Model


Adversary

Client 1

. . .

Client Z

. . .

Client n

Client Puzzle

req. arrival interval: c^N

Server

L

r_L … … r_1

S

Sτ

Time Buffer c_r

$3\times 2^{kr-1}/s \ \le\ c_r = (t^r_Z - t^r_L) + (L/S-\tau) + c^N$

Property 1: Solution Latency

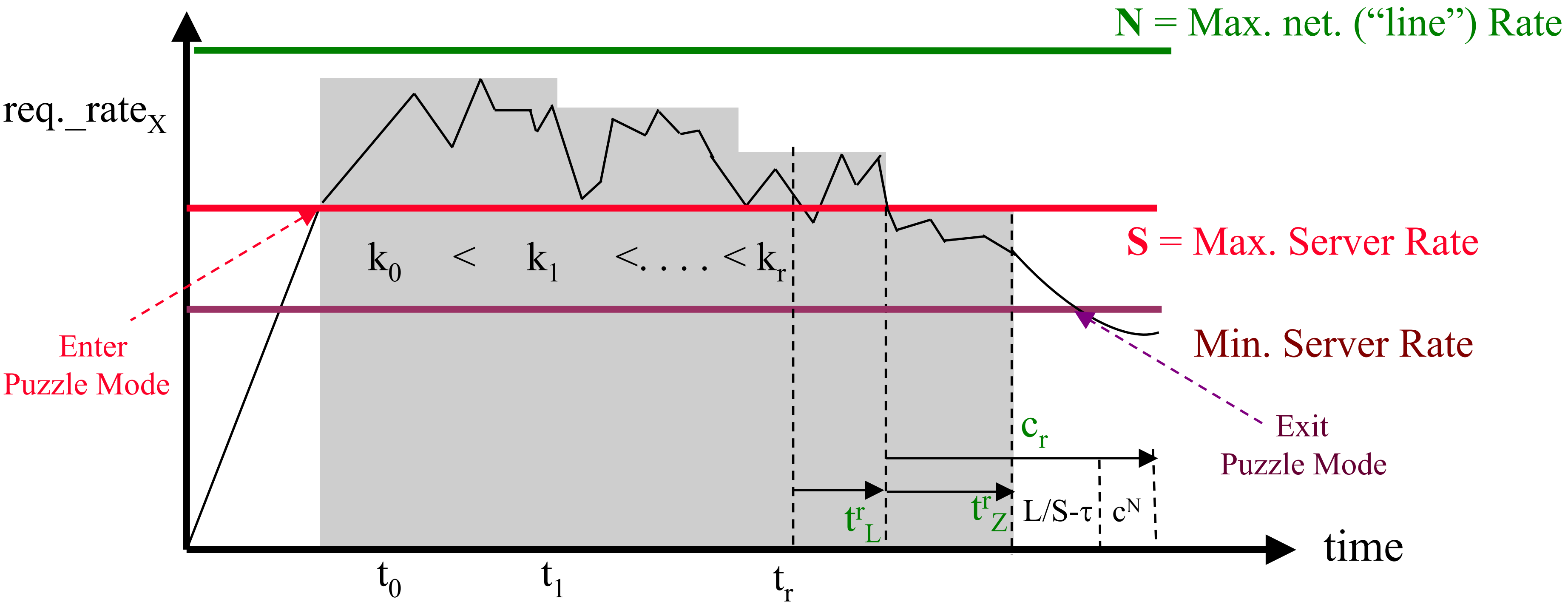
With high probability

a) $Z \geq 2L+2\sqrt{(6L+9)}+6$ *clients solve at least* L *puzzles in* $2^{kr-1}Z$ steps (in time $2^{kr-1}/s$)

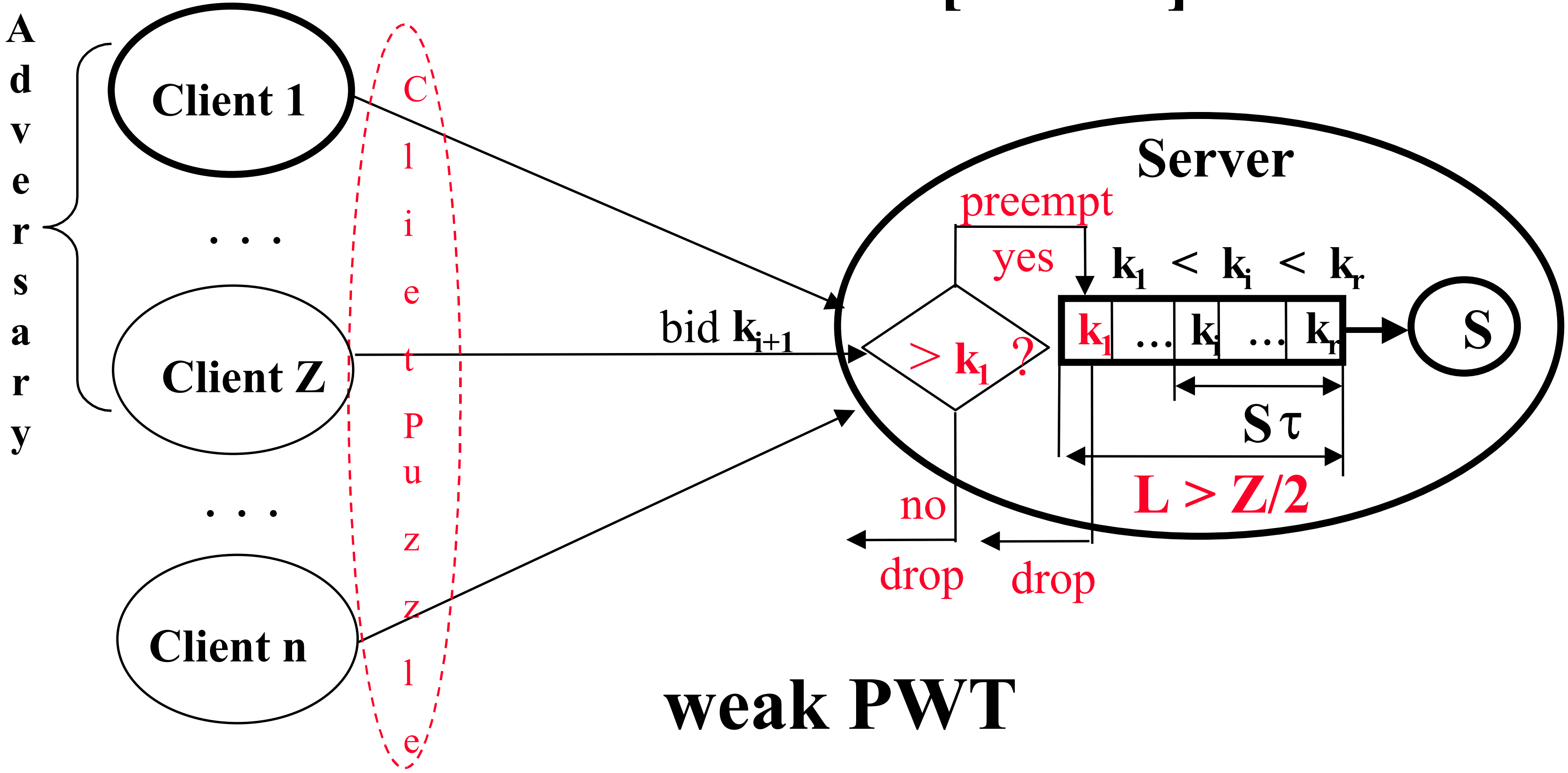
b) Z *solve at least* Z *puzzles in* $2^{kr+1}Z$ steps (in time $2^{kr+1}/s$)

Property 2: Request-Rate Control (WPWT):

$N_Z^{k_r} \leq S$ over interval $\mathbf{t^r_L + c_r}$ <=> $\mathbf{k_r \geq 1 + \lceil \log(Z/S - c_r)s \rceil}$, where $\mathbf{c_r < Z/S}$



Puzzle Auctions [WR03]



weak PWT

Pr [any client C’s request is accepted for service in time ***T***]

= **Pr** [any client C’s request is accepted for service in ***R+1*** rounds $k_0, k_1, \ldots, k_r$]

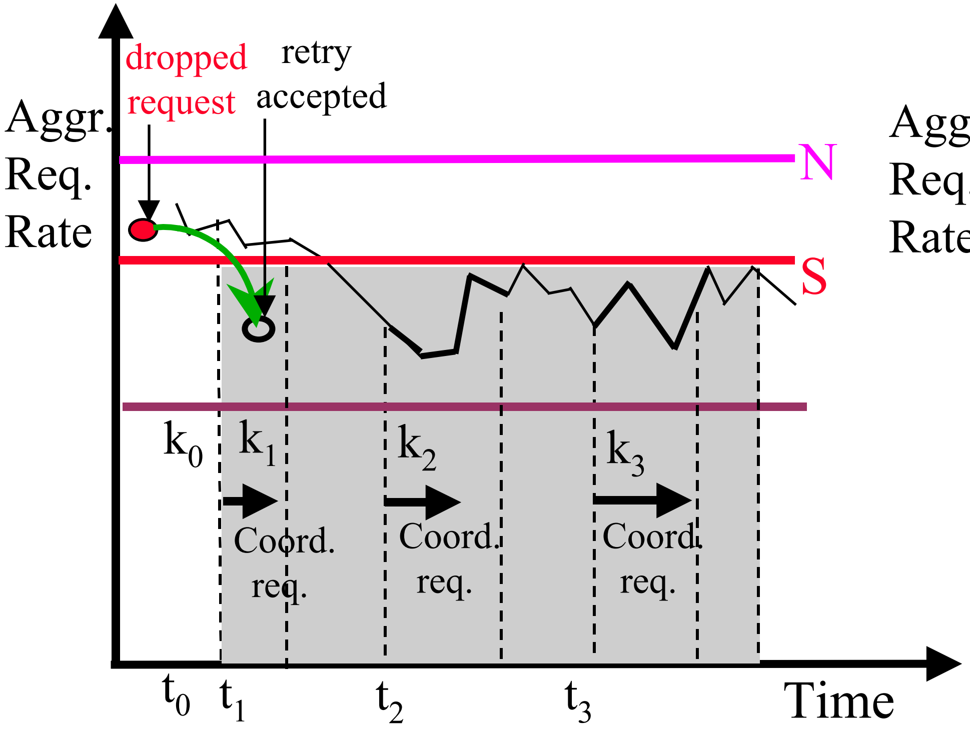
= 1- Pr [any client C’s request is denied at round ***R+1***]

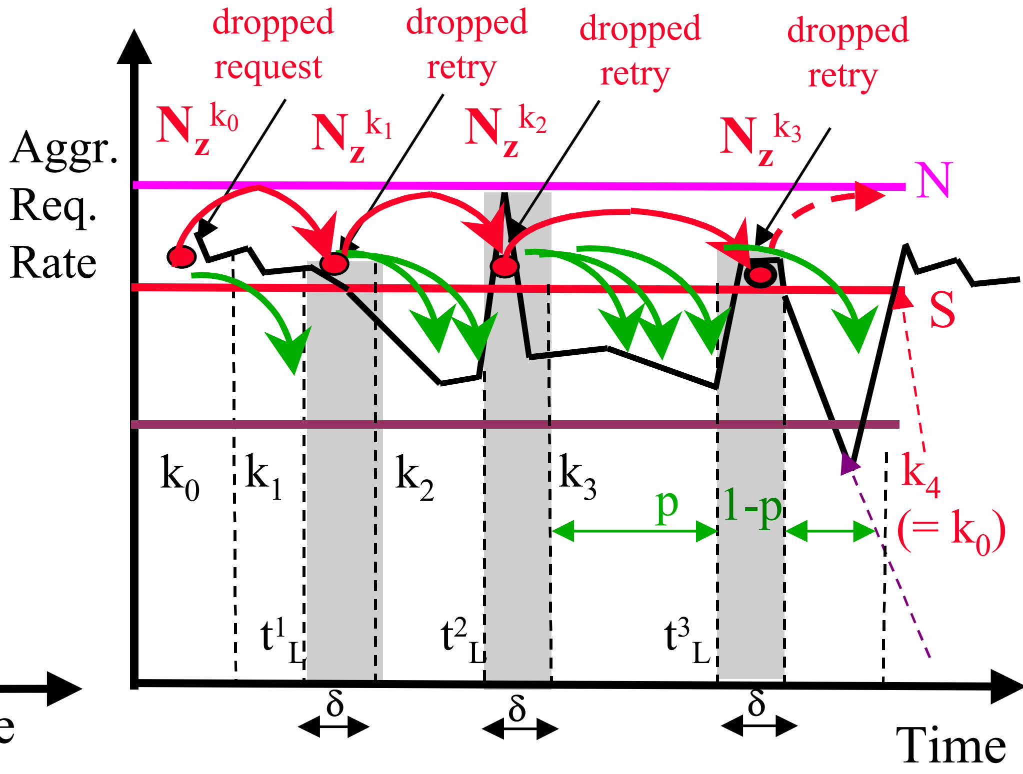
$$\geq \mathbf{1 - (1 - 2^{-k_0})}^{2^{k_0-1}L/Z - \tau s} \prod_{i=1}^{R} \mathbf{(1 - 2^{-k_i})}^{(2^{k_i-1} - 2^{k_{i-1}-1})L/Z} = p > 0$$

Dependency on attack parameter Z

Attack Coordination

Goal: Deny Strong Guarantees (FWTr, PWTr, MWTr)





Coordinated Attack **for a** $k_0 < k_1 < k_2 < k_3$ **sequence**

$$L/N_z^{k_i} < \delta < Z/S$$

$p = \max(p_i), i = 1,\ldots,m$

Pr [client req. is accepted within *m* retries] $< p \sum_{i=0}^{m} (1-p)^i = 1-(1-p)^{1+m} < 1$

What Do “Client Puzzles” Achieve ?

… *very weak* client guarantees *at high* …

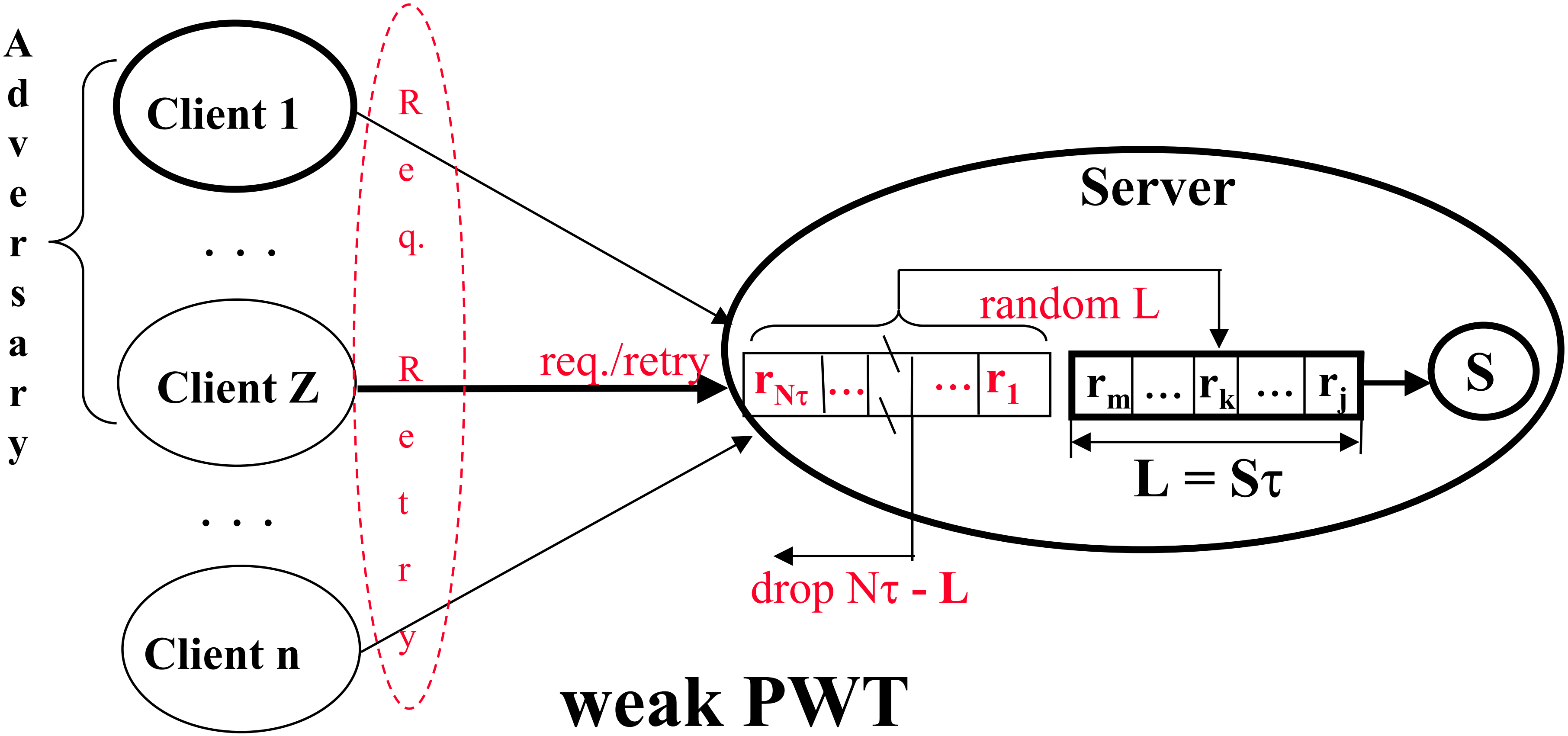
Client Guarantees ?

- WPWT (by P2)
- ***w****PWT* (with assumption L > Z/2)
- *no PWT, no FWT* => *no MWT*

… and *unnecessary* request overhead.

- ***random scheduling (with preemption) achieves wPWT (PWT)***

Example 1: Random Sτ = L < Nτ (w/o preemption)



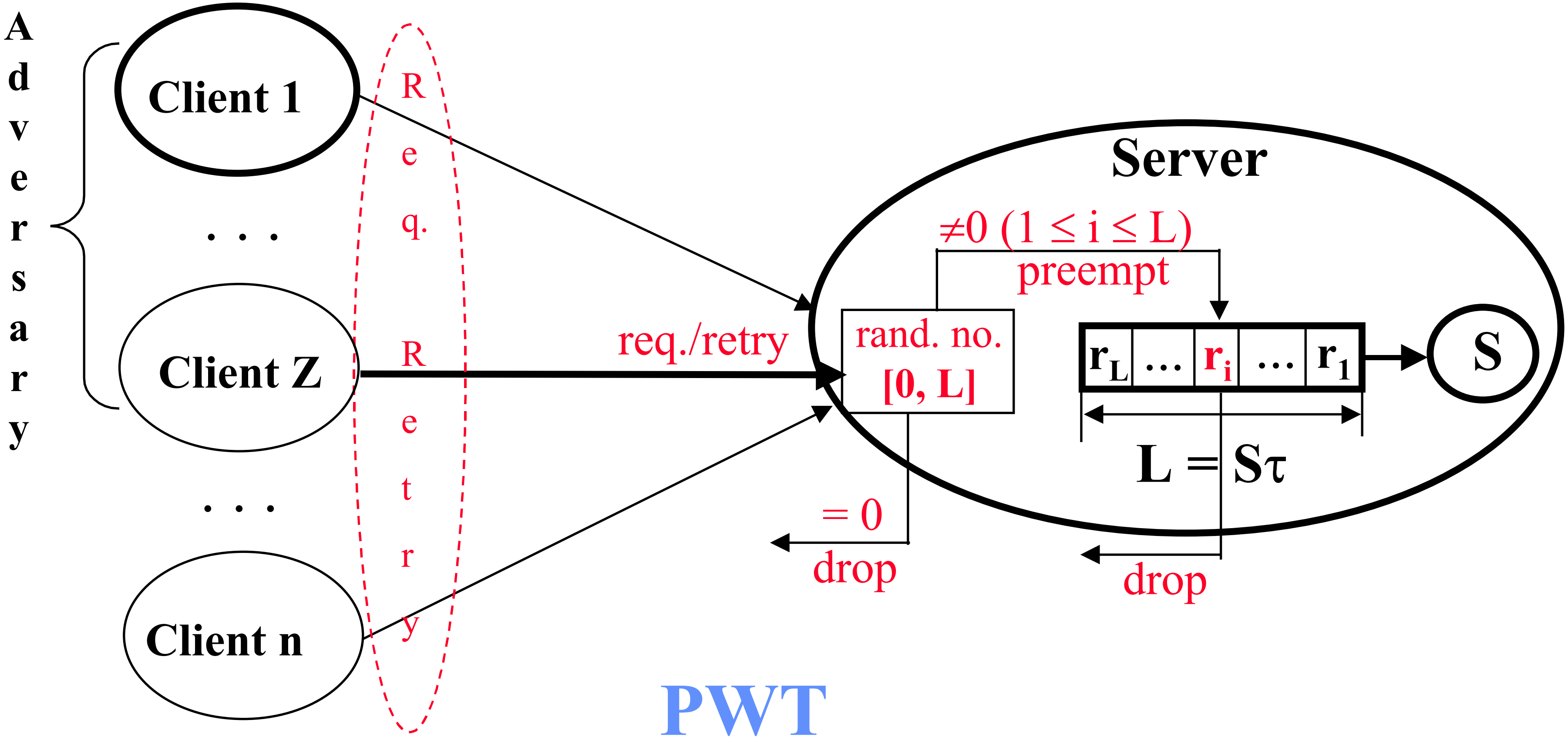
weak PWT

$n_i / S\tau$ = no. of requests received / processed at *round i*; $S/N \leq \min\{S\tau/n_i\}$, $i = 1, \ldots, r$

Pr [client request is accepted for service *eventually*]
≥ **Pr** [client request is accepted for service in *r* rounds]
= 1- Pr [client request delayed to round *r*] $\geq p = \mathbf{1 - (1 - S/N)^{r} \rightarrow 1}$

Dependency on attack parameter r

2. Example 2: Random $L = S\tau$ with Preemption



Adversary

Client 1

. . .

Client Z

. . .

Client n

Req.

Retry

req./retry

Server

$\neq 0 \ (1 \leq i \leq L)$

preempt

rand. no.

[0, L]

r_L … r_i … r_1

S

$L = S\tau$

= 0

drop

drop

PWT

Pr[*req./retry* is accepted by Server in $T \geq \Delta+\tau$]

= **Pr**[*req_buffer*[1…L] ← *req./retry in* Δ] x **Pr**[*req./retry* not dropped in τ]

$\geq [1-1/(L+1)] \times [1/(L+1)+(L-1)/(L+1)]^{n} = [L/(L+1)]^{1+n}$

$\geq [S\tau/(S\tau+1)]^{1+N\tau} = \rho$ =/= 0

(independent of the number and aggregate request rate of "zombies").

3. Idea: Explicit Control of Client Request Rate
+
Maximum Waiting Time Guarantees

Phase 1: Client-Proliferation Control

(Stateless Session) Cookie => Reverse Turing Test (e.g., CAPTCHA) passed

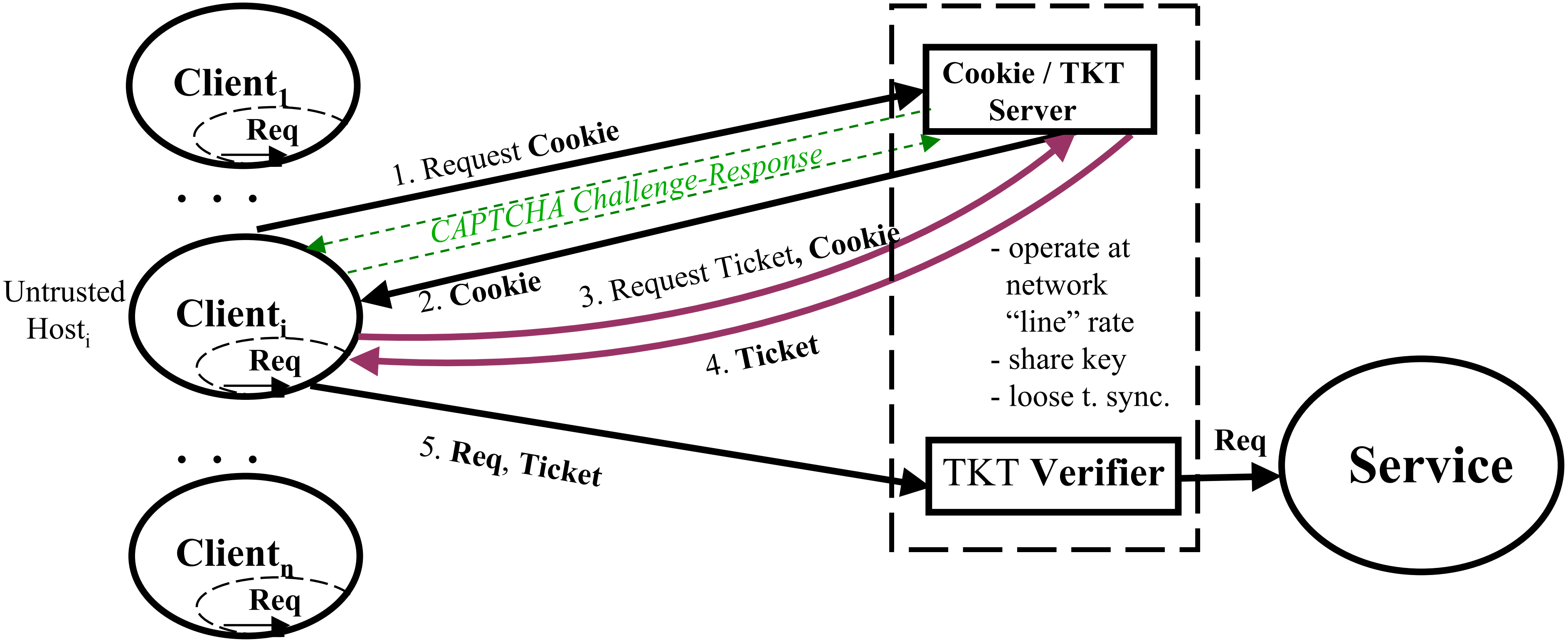
- forces *human-level collusion* and *coordination* on global scale

Phase 2: Request-Rate Control for Individual Clients

Service Req. => Valid Rate-Control Ticket => Valid Cookie
(=> solved puzzle, no Phase 1)

- ticket: time-slot reservation, total ordering
(e.g., a "Bakery Mechanism")

Phase 1: Client-Proliferation Control

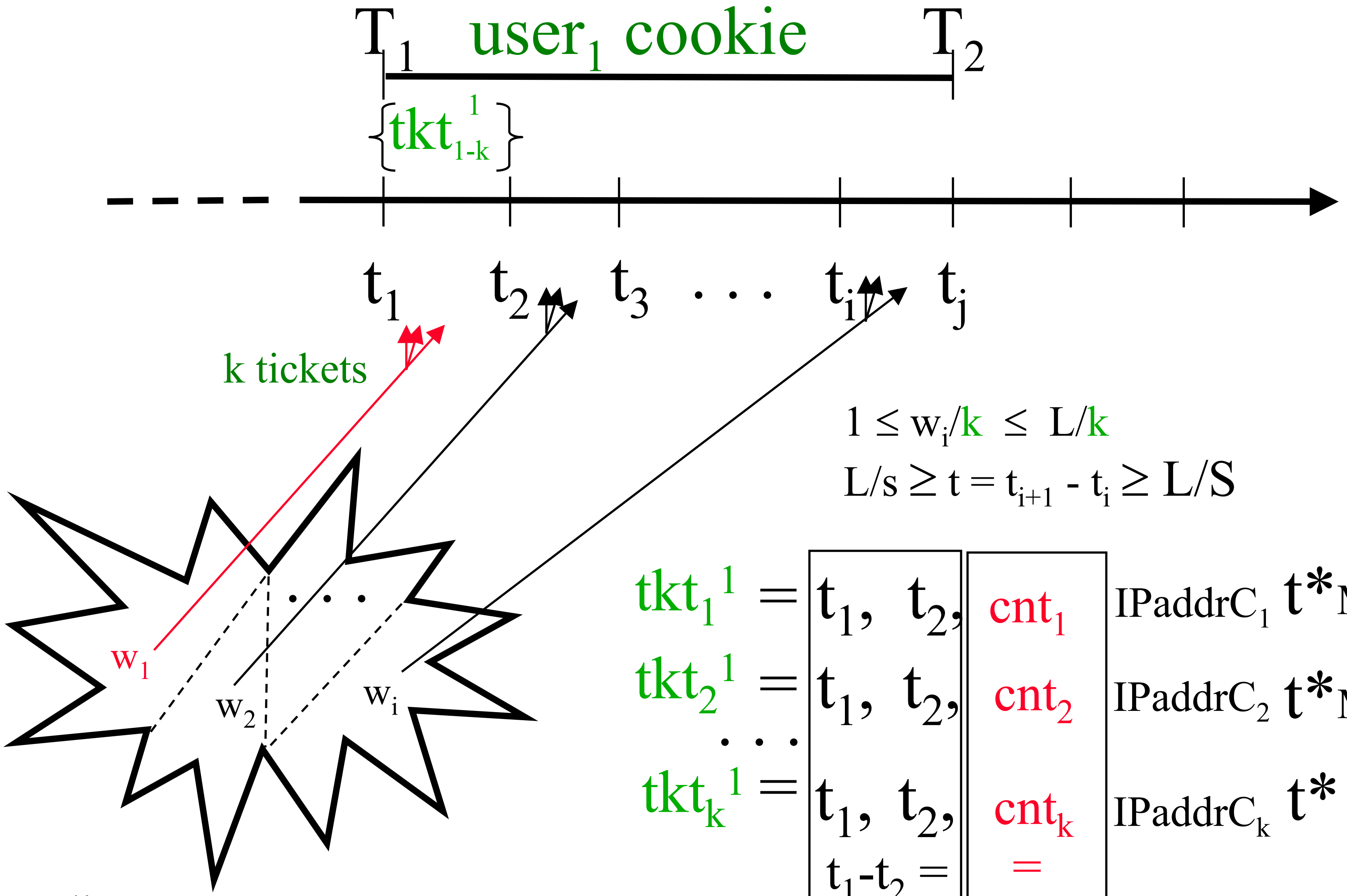


- operate at network “line” rate
- share key
- loose t. sync.

Phase 2: Request-Rate Control for Individual Clients

Cookie / Ticket duplication by Clients ? theft, replay by Clients ?

Client Request-Rate Control: *Time-Slot Reservation*



$1 \le w_i/k \le L/k$

$L/s \ge t = t_{i+1} - t_i \ge L/S$

$tkt_1^{1} = t_1,\ t_2,\ cnt_1$ $IPaddrC_1$ t^* MAC_1

$tkt_2^{1} = t_1,\ t_2,\ cnt_2$ $IPaddrC_2$ t^* MAC_2

. . .

$tkt_k^{1} = t_1,\ t_2,\ cnt_k$ $IPaddrC_k$ t^* MAC_k

$t_1 - t_2 = t \ge L/S$

$cnt_k = w_1 \le \frac{L}{k}$

client requests = Σw_i

* t = time at server

- ***Cookies and TKTs*: similar function, different time scale**

  e.g., cookie = … T_i, T_j, tkt.cnt, IPaddr_list, t, MAC

- ***Sliding Time Window caches* of TKTs use @ Verifier**

  of Cookies @ TKT Server

- ***Packet filtering in Access-Point Routers***

  (counters large-scale IP spoofing; already deployed)

- ***Optimization*: Ticket Count w_{opt} ; Window $t_{opt} = t_{i+1} – t_i$?**

  1. Effect of unused reservations => *small* $t_{i+1} – t_i$=L/S.

  w = 1, k = 1 => *Total Ordering* of Requests

  (low impact TGS traffic; e.g., content distribution, protocol exchanges)

  2. Reducing Client – TKT Server communication

  => all L requests in one ticket and *large* $t_{i+1} – t_i \geq$ L/S.

  (high-impact TGS traffic; e.g., high-speed, bursty transactions)

  w = L, k =1 => *Server Underutilization (by zombies not issuing requests)*

Simple Optimization: $\mathbf{w_{opt}}$, $\mathbf{t_{opt}}$

$C_{total} = C_{client} + C_{server} = c_1 A_r/w + c_2(1\text{-}r)w,$ where

w = total number of requests in a window (for all that window's tickets)
$\mathbf{c_1}$ = communication cost for getting a ticket from TGS
$\mathbf{c_2}$ = server-utilization cost of waiting for a request not issued within **w**
$\mathbf{A_r}$ = average number of Application Requests (Client -> Server requests)
r = percentage of legitimate clients $\mathbf{(0 \leq r < 1)}$

$$\delta C_{total}/ \delta w = 0 \Rightarrow \mathbf{w_{opt}} = \sqrt{\frac{\mathbf{c_1 A_r}}{\mathbf{c_2(1\text{-}r)}}}, \text{ constrained by } \mathbf{1} \leq \mathbf{w_{opt}} \leq \mathbf{L}$$

$$\mathbf{L/S \leq t_{opt} = w_{opt}/S \leq L/s}$$

Simulations

Parameters: $\mathbf{c_1/c_2}$, **r**, $\mathbf{A_r}$
Processes: client request, service response
Attack characterization: *low* inter-arrival times of client requests to TGS, *low* **r**, *high* A_r

What can General Request Constraints Achieve ?

Additional constraints on Client Requests

- Examples
    - MWT for coordinated requests from Clients to Servers under attack
        - Client requests to multiple Servers
        - application-related Clients requests to Servers (e.g., is Σ MWT$_i$ for Client$_i$ requests to Server$_i$ within ΔT ? in [t$_1$, t$_1$] ?)
    - patches: safety constraints not enforced in Server (e.g., parameter constraints)

SUMMARY

1) ***problem reduction**: flooding freedom of a simple (distributed) service*

- RCS Service (*Server 1,…, Server k*) has specialized, simple function
  - ⇒ max. service rate of TKT *Service* is *at network rate or above*
  - ⇒ *flooding is impossible*

2) ***maximum waiting time (MWT) per request***

- request-rate control for *individual* clients (e.g., client puzzles for TKT requests)
- protection against TKT theft
  - packet filtering on IP addr. at access-point routers, sliding-time-window caches of TKT use
- problem: long MWT

3) ***reasonable MWT for legitimate clients***

- control of client proliferation
  - reverse Turing tests (CAPTCHAs), stateless cookies
  - protection against cookie theft (same as for TKT theft)