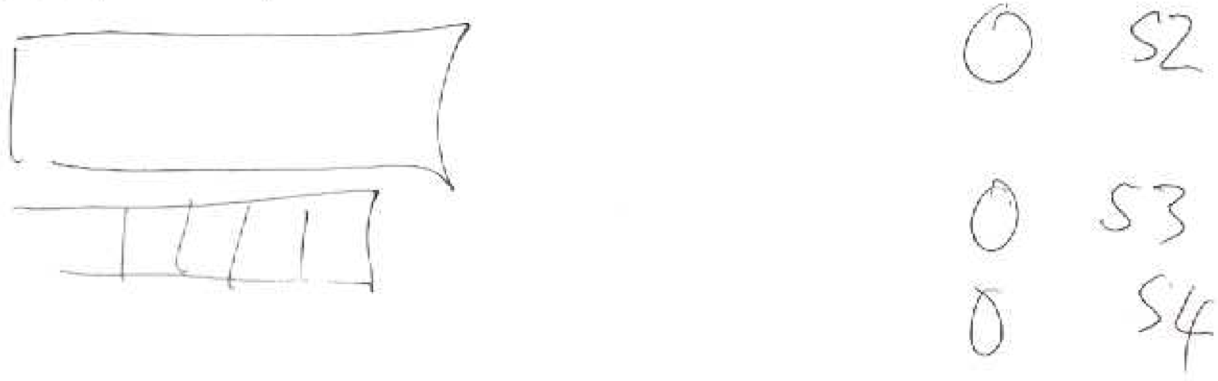
Load balancing

* LAN

Task size distribution
- Classical systems (exponential dist. $P(X > x) \simeq 0$
- New systems (Web servers) Paretodist. $P(X > x) \neq 0$

Task size
- KNOWN (CLASSICAL SYSTEMS)
- UNKNOWN.

CENTRAL DISPATCHER



$P(X > x)$ = probability that a given task has a size greater than x

$x \uparrow$

① $x \uparrow$ $P(X > x) = 0 \Rightarrow$ exponential distribution (ie most of the tasks have similar size

L7-1

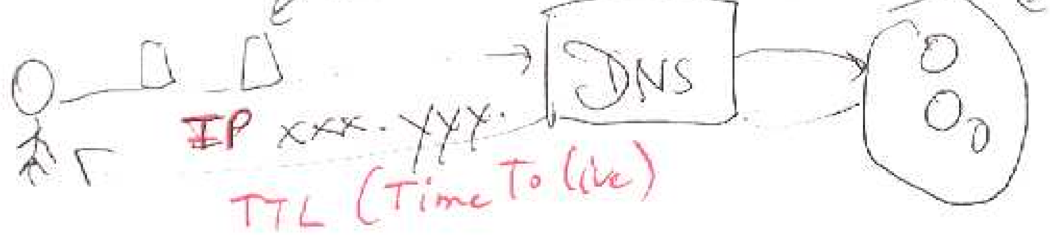
② $x\uparrow$ $P(X>x) \neq 0$ – Pareto distribution (ie high variability in the task size. 10-20% are very large tasks)

In case ①, the Least Loaded First (LLF) is THE optimal solution for balancing the load, where the load of a given server (S_i) is defined as the # of tasks in S_i's queue

* WAN (Geographically distributed Web Servers)

Intermediate DNS (I-DNS)

Cluster of web servers

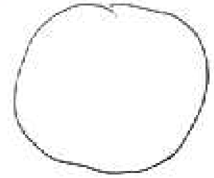


DNS has to balance the load (dispatcher)

DNS controls ONLY a limited % of traffic to clusters (because the mappings are stored in the I-DNS)

Slicing of tasks

TTL – zero



Replication of # geographical locations

L7-2

TTL-zero (when you assign a very small value to TTL)

↳ ~~to~~ overload DNS (which is <u>not</u> appropriate)

Hot domain(s) — [generate a lot of traffic] → small TTL

Cold domain(s) ← Large TTL

↳ most of the requests coming from hot domains will be controlled by DNS

↳ Come back to the techniques ~~[illegible]~~ based on centralised dispatcher.

TAGS (Task Assignment by Guessing the Size)

① – Pareto distribution (for task size variation)

② – task size is not known

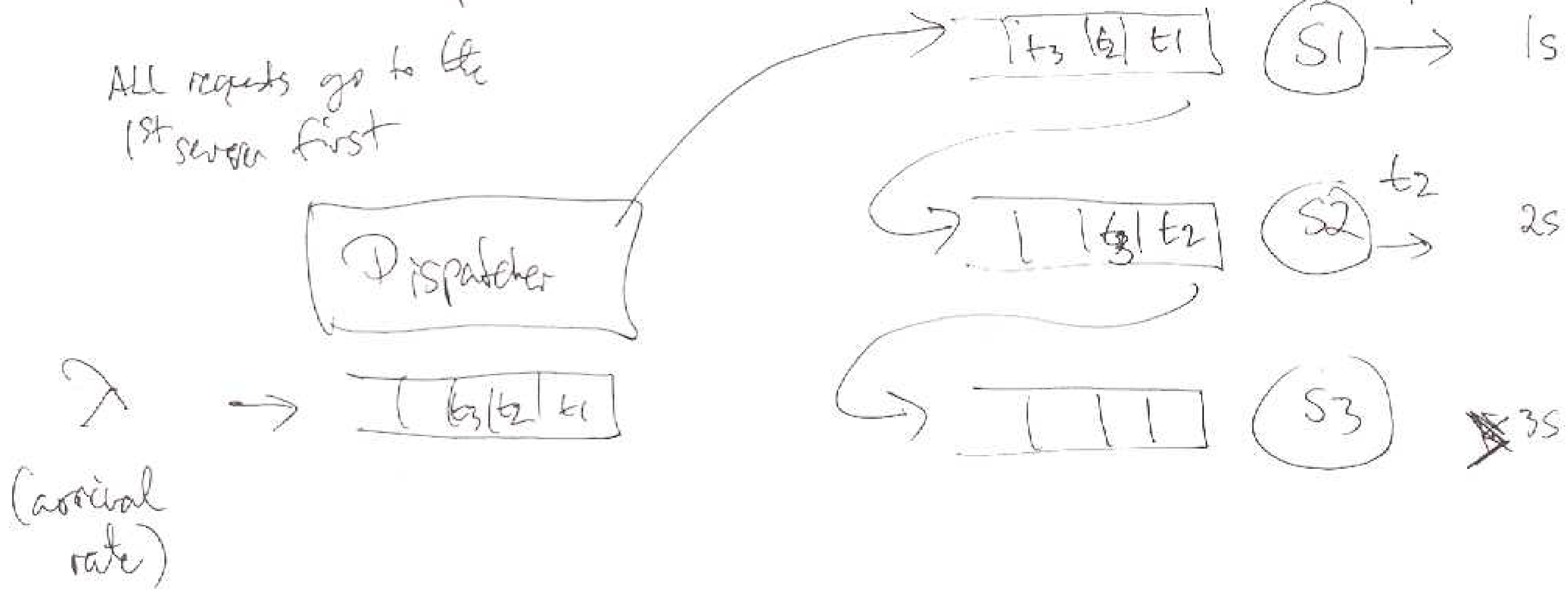
↳ developed for LAN

$t_2 = 1.5s$

$t_1 = 0.5s$

$t_3 = 4s$

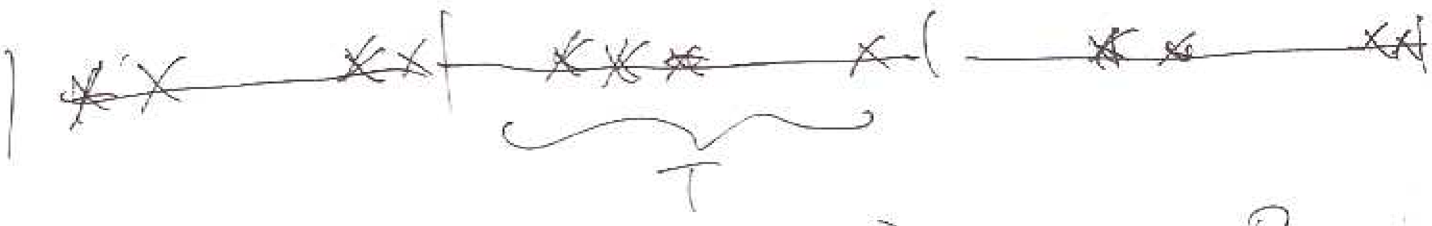
ALL requests go to the 1st server first



- Poison Distribution for the arrival (STABILITY IN THE SYSTEM ≡ "same" behavior of arrival across different periods)
- Non-Poison ("non normal behavior") approximation)

L7-4

e



(every hour)

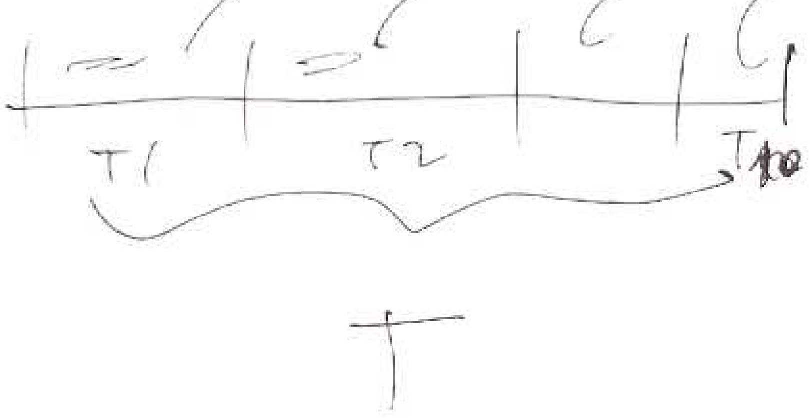
Poison

$$P(X > x) = 0$$

$x \uparrow$ X: distance between two events

exponential distribution

Non-Poison



Problems with TAGS

(1) A lot of restarting of "large" tasks

e.g. task t_3 has been restarted 2 times

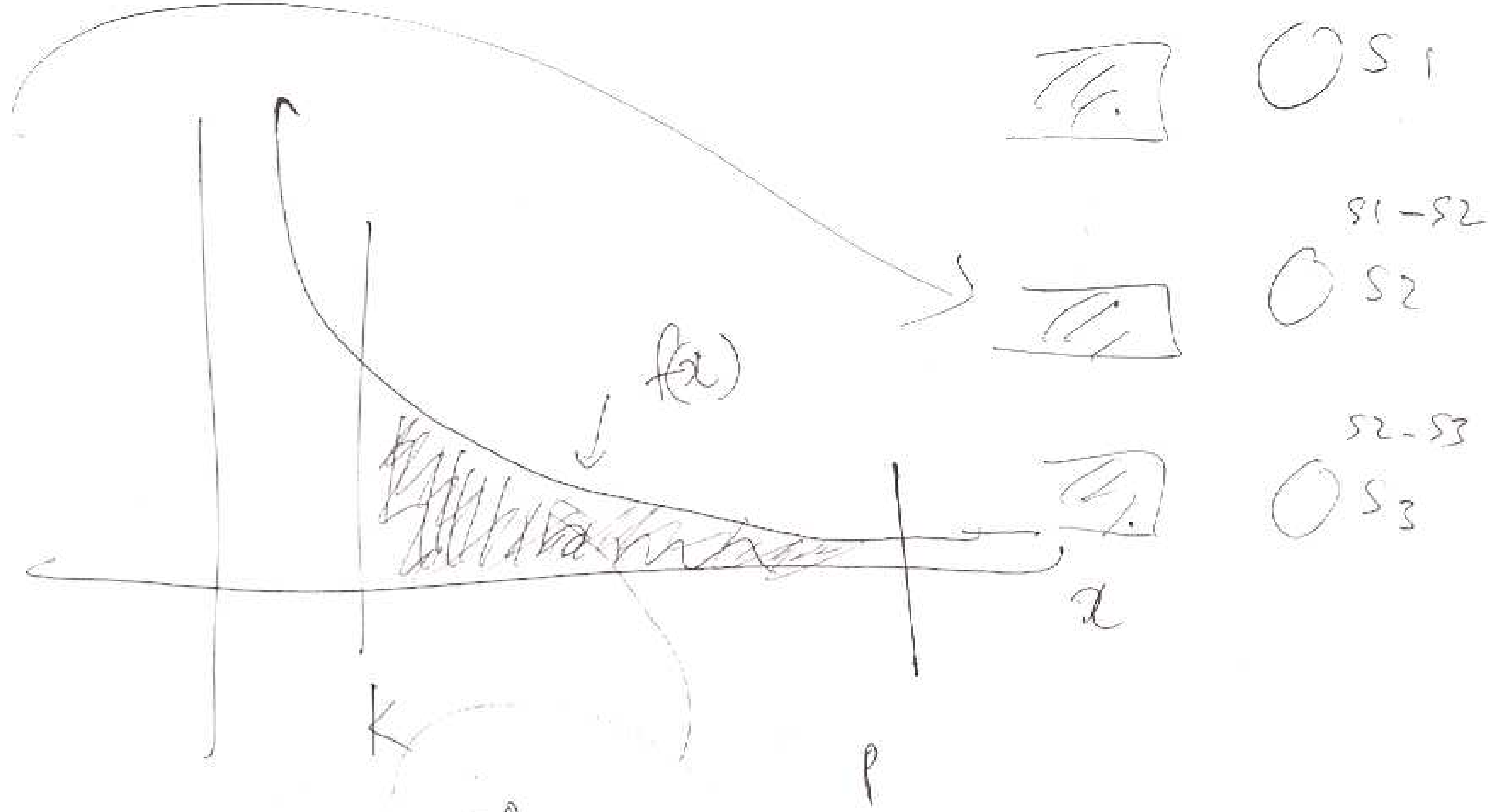
$\text{RESTART}(t_3) = 3s$ cost of restarting

(2) After a task is re-started at another queue, it has to wait (excess in waiting times)

$\text{EXCESS\_WAITING\_TIME}(t_3) = 2s + 1s = 3s.$

$2s$: processing of t_2 @ S_2 − processing of t_3 @ S_1

$$\int_{S_1}^{S_2} x f(x)$$



$$E(X) = \int_{K}^{P} f(x)\,dx = \text{average task size}$$

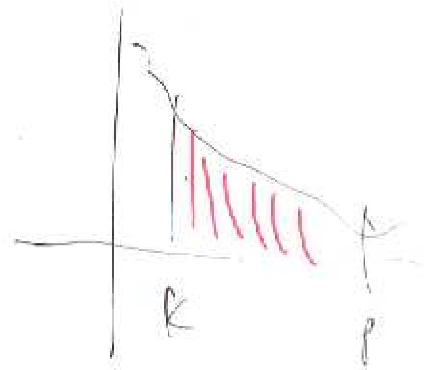
$$E(X^2) = \int x\, f(x)\, dx \quad \text{second one}$$

L7-7

$f(x) = \_\_\_\_\_\_$ (density)

$E(X^j) = j^{th}$ moment

$= \int_K^P x^{j-1} f(x)\,dx$



$j=1$ first moment (Task size)

AVERAGE TASK SIZE $E(X) = \int_K^P x^{1-1=0} f(x)\,d(x) = \int_K^P f(x)\,dx$

$j=2$ $E(X^2) = \int_K^P x\, f(x)\,dx$

second moment

IS USED IN WAITING TIME

$= \frac{\lambda E(X^2)}{2(1-\rho)}$

L7-8

$E(S1)$ = slown down in S1

$E(S2)$ = ______ in S2

$$E(S_1) = \frac{WT(S_1)}{|S1|}$$

↖ average task size

$$E(x) = \int_0^{S1} f(x)\,dx = $$

$$WT(S_1) = \frac{\lambda E(x^2)}{2(1-\rho)}$$

$$E(x^2) = \int_0^{x_0} x f(x)\,dx$$

$$E(S_2) = \frac{WT(S2)}{|S_2|} = \frac{\lambda E(x^2)}{2(1-\rho)}$$

$$\int_{S1}^{P} f(x)$$

$$E(x^2) = \int_{x_0}^{P} x f(x)$$

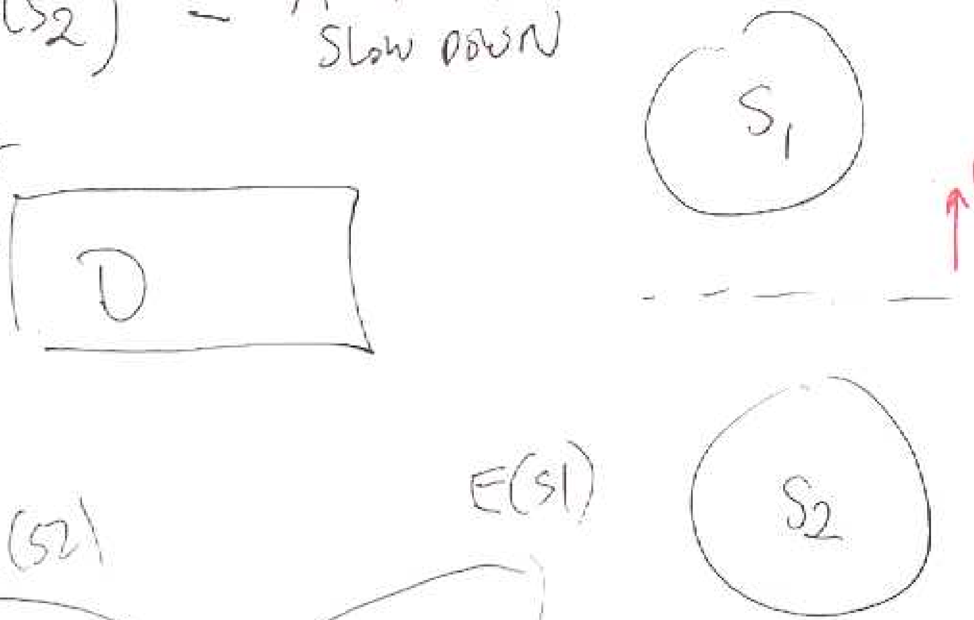
L7-9

SITA-V (Crovella)

$p_1 E(S_1) + p_2 E(S_2)$ = AVERAGE SLOW DOWN

p_1: proportion of tasks going to S_1

p_2: proportion of tasks going to S_2



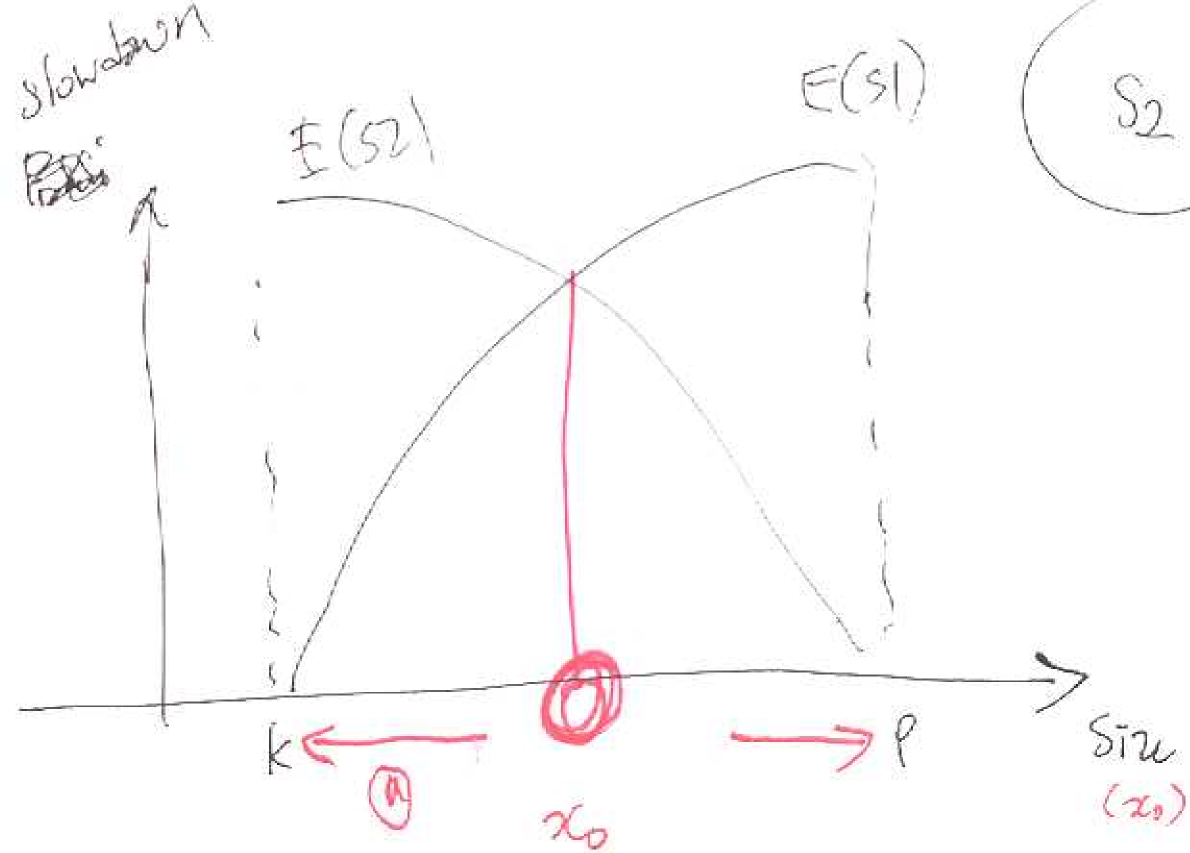
(a) x_0 (cut off)

OPTIMISE x_0 in such a way you get better performance.

(here: optimise SLOW DOWN)

=

$\dfrac{WT = \text{WAITING TIME}}{TS = \text{TASK SIZE}}$



t_3 t_2

(b)

L7-10

$$\frac{WT(t_3)}{TS(t_3)} = \frac{WT(t_2)}{TS(t_2)}$$