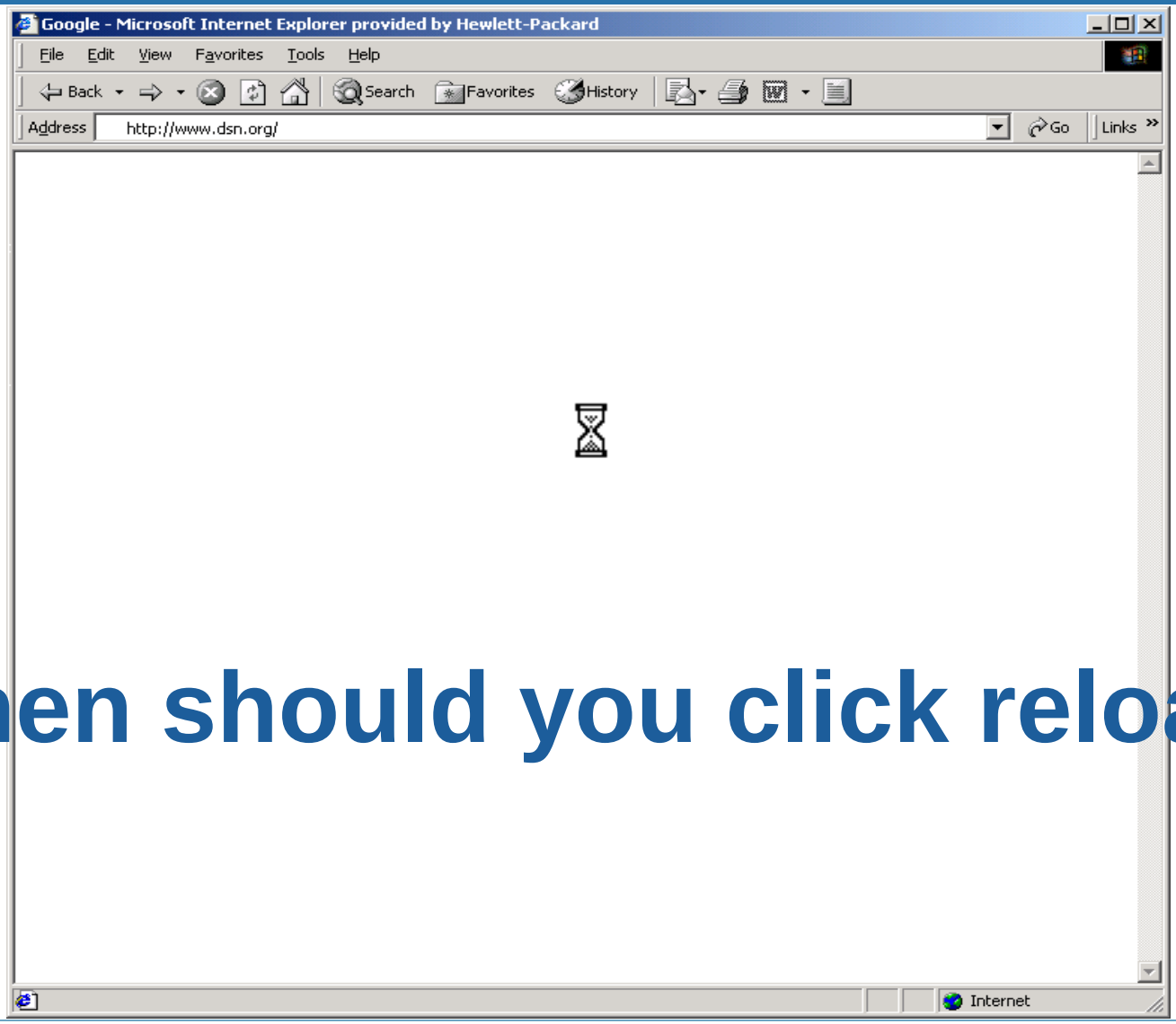
to diagnose or not to diagnose

Aad van Moorsel, Newcastle
aad.vanmoorsel@newcastle.ac.uk
with
Katinka Wolter
Philipp Reinecke, Humboldt, Berlin

the problem



when should you click reload?

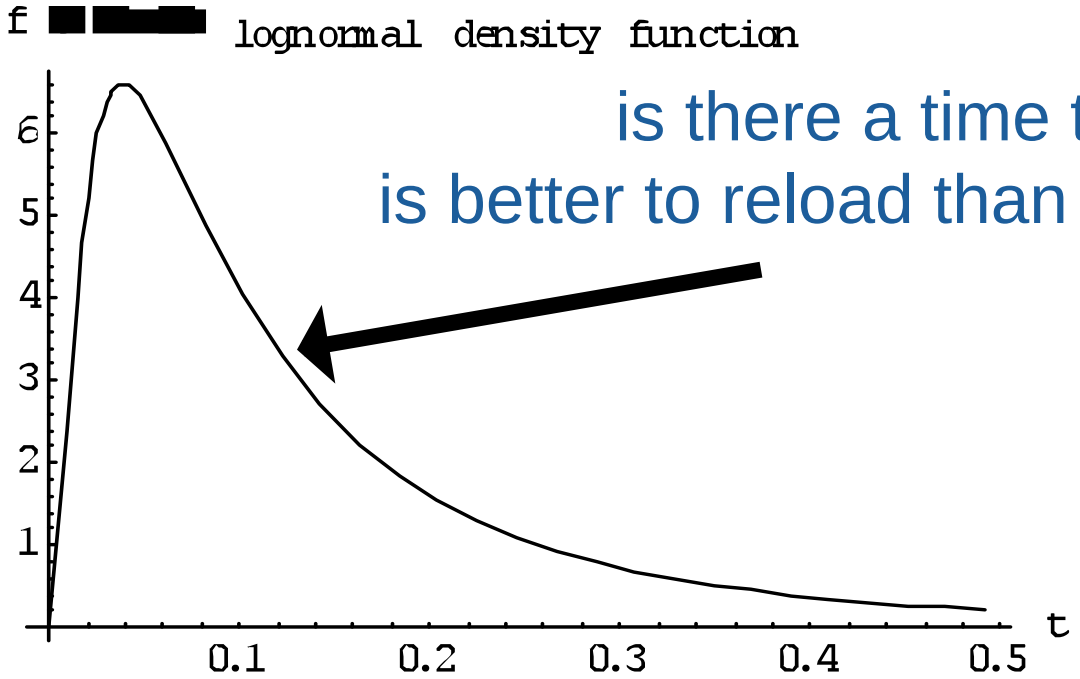
content

- optimal restart times to make deadlines: some results
- how does this relate to failure diagnosis?
- some data and some results

model

- retries are independent and identically distributed
- retries preempt the previous try

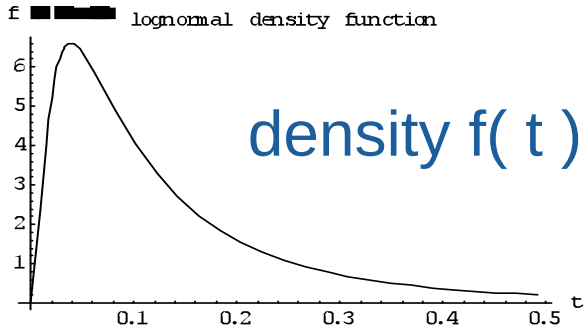
all we need is the probability distribution of the completion time T



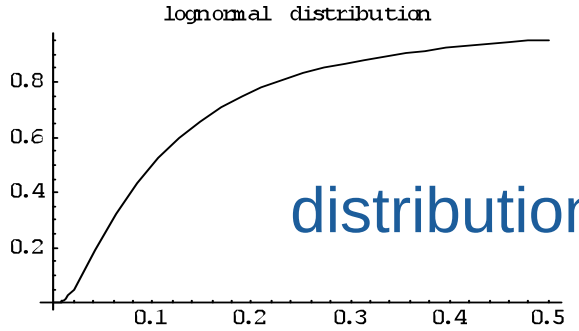
is there a time t where it
is better to reload than to keep waiting?

model

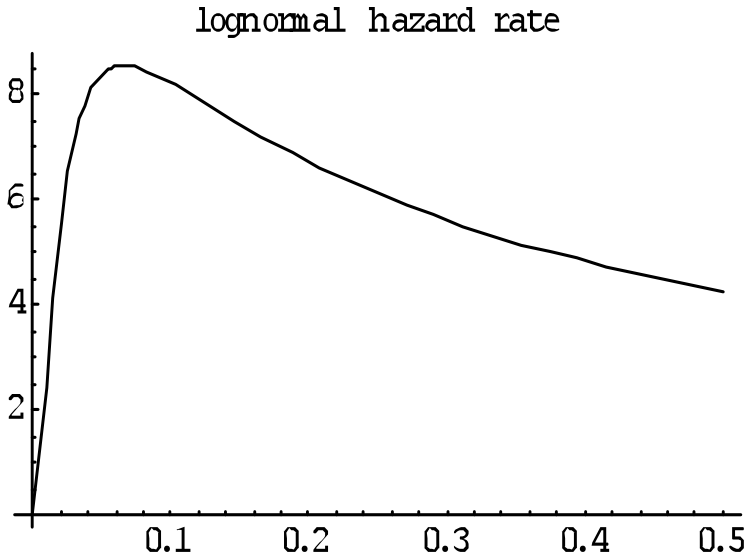
we use:



density f(t)



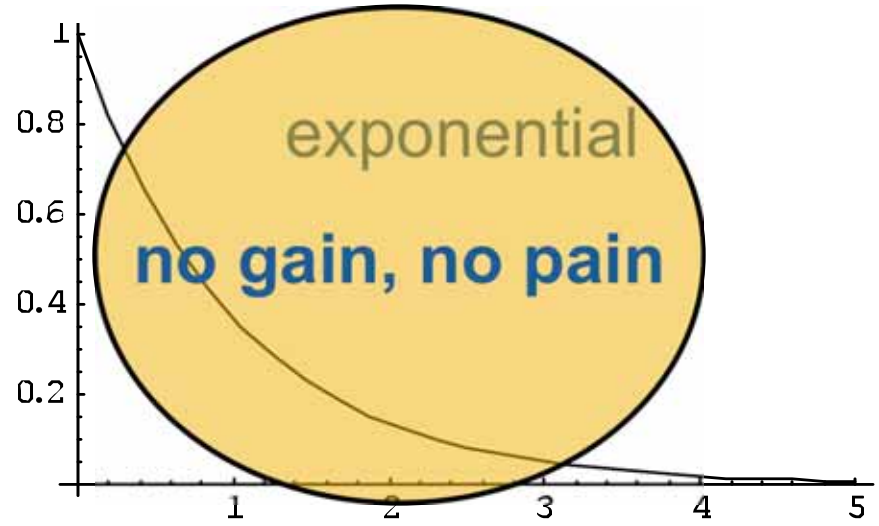
distribution F(t)

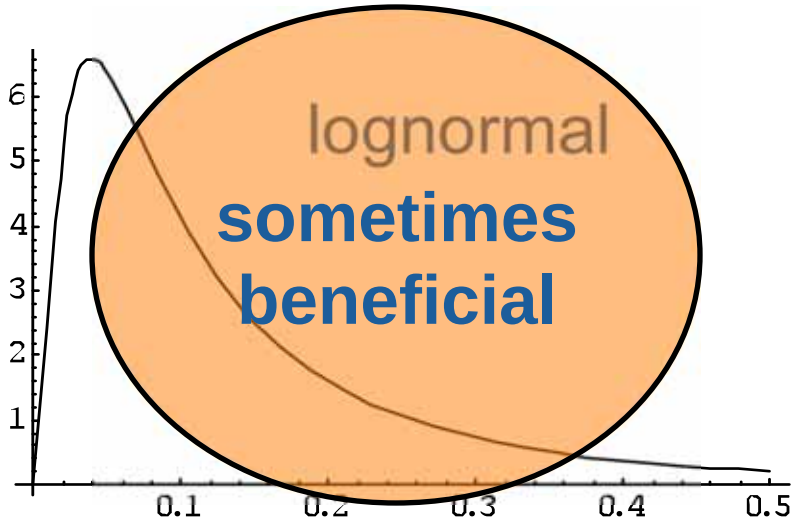


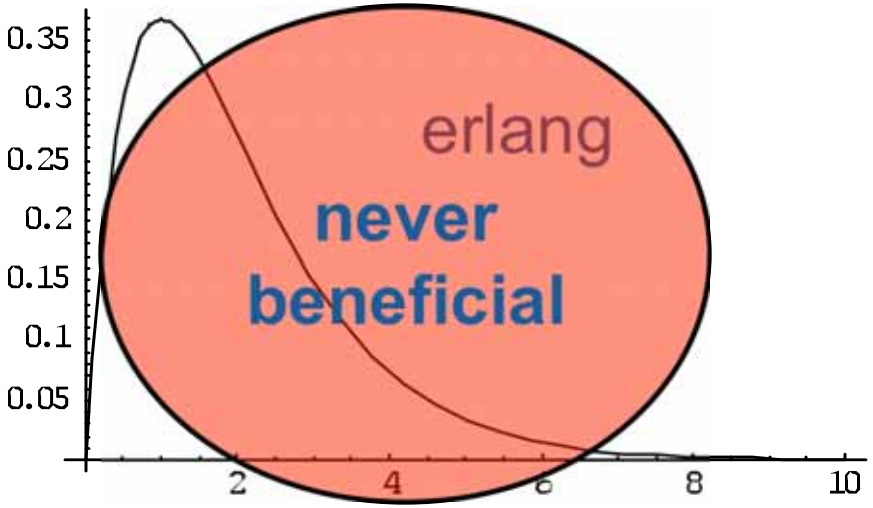
hazard rate

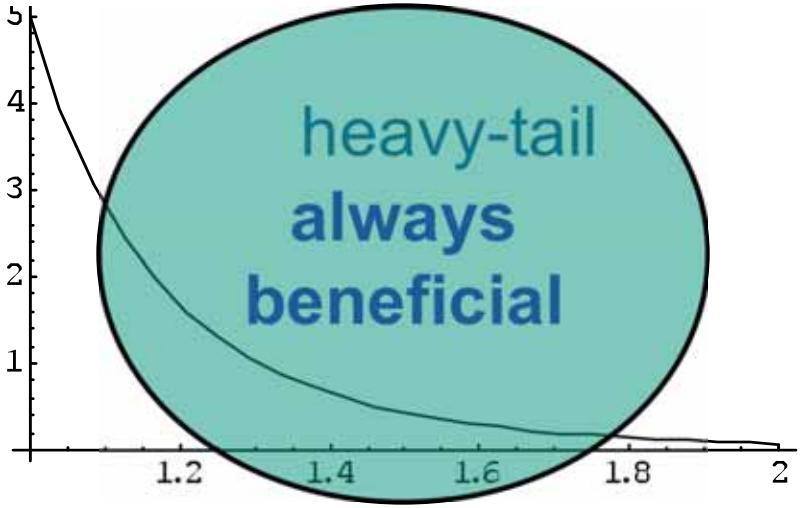
h(t) = f(t) / (1 – F(t))

which distribution is amenable to restart (densities)

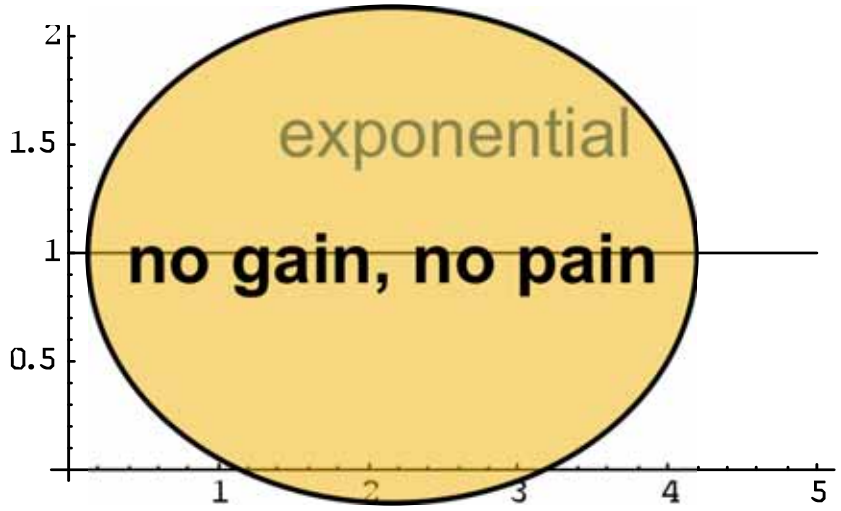


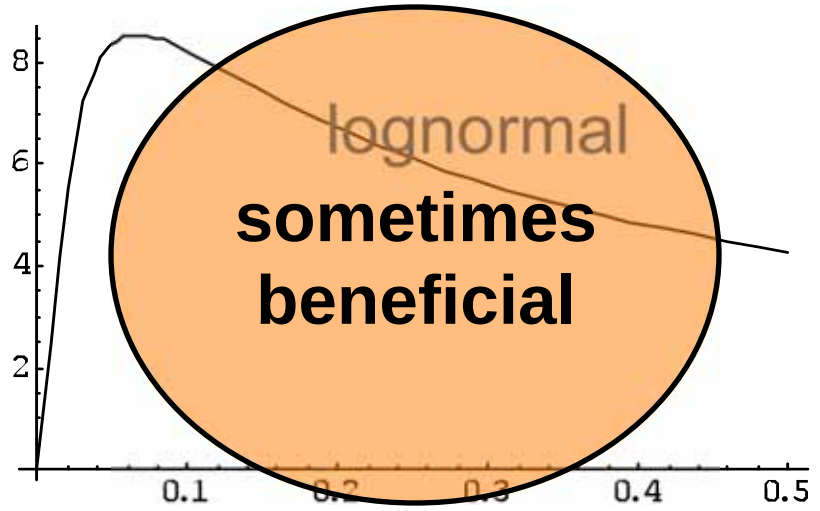


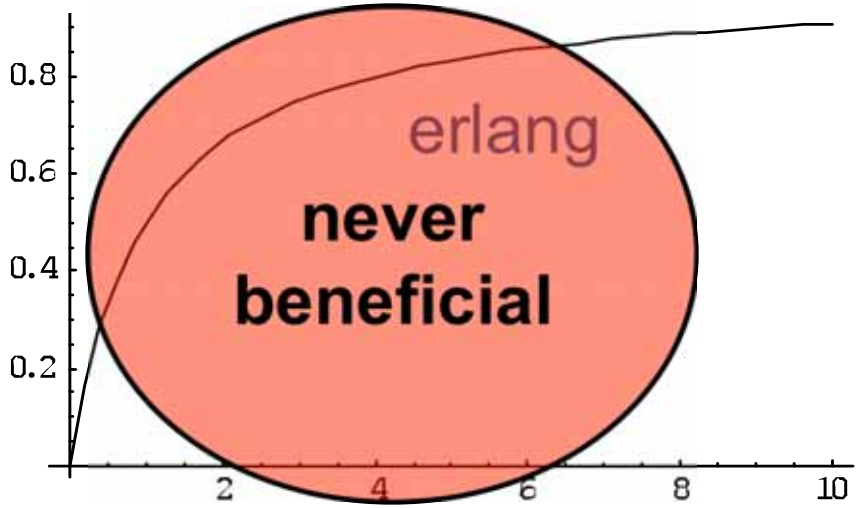


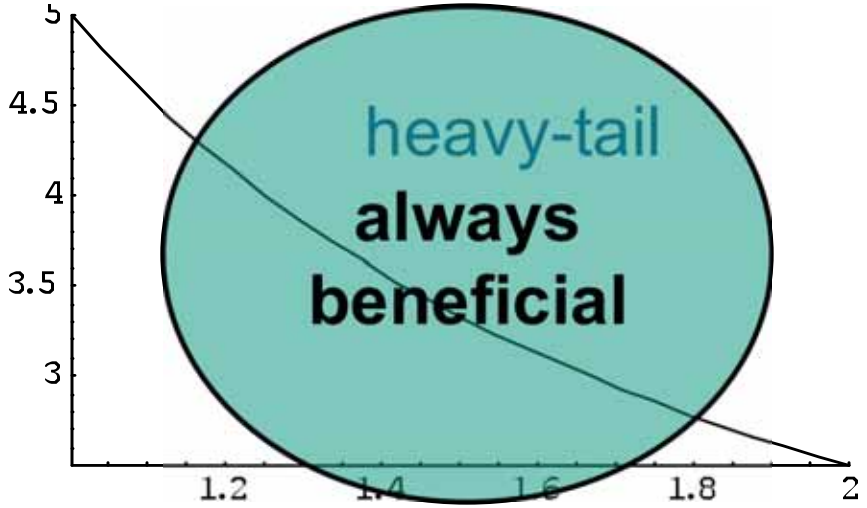


which distribution is amenable to restart (hazard rates)









model

metric: probability make the deadline d

without restart:

$$F(d)$$

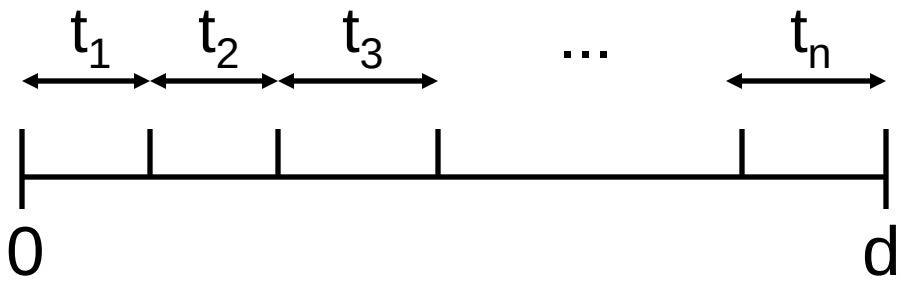
with restart at multiples of t: flip a coin with probability F(t)

$$F_t (d) = 1 - (1 – F(t))^{d/t}$$

restart at time t makes sense if

$$F_t (d) > F(d)$$

local extremes: equi-hazard points



minimise $(1 - F(t_1)) . (1 - F(t_2)) ... (1 - F(t_n))$

for local extremes we find:

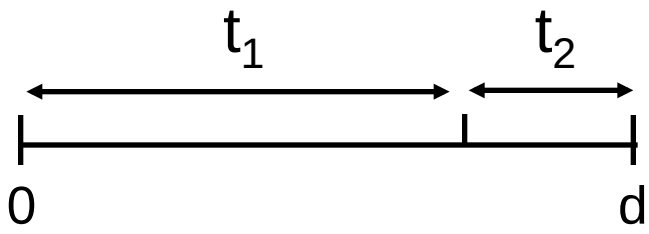
$$h(t_1) = h(t_2) = \ldots = h(t_n)$$

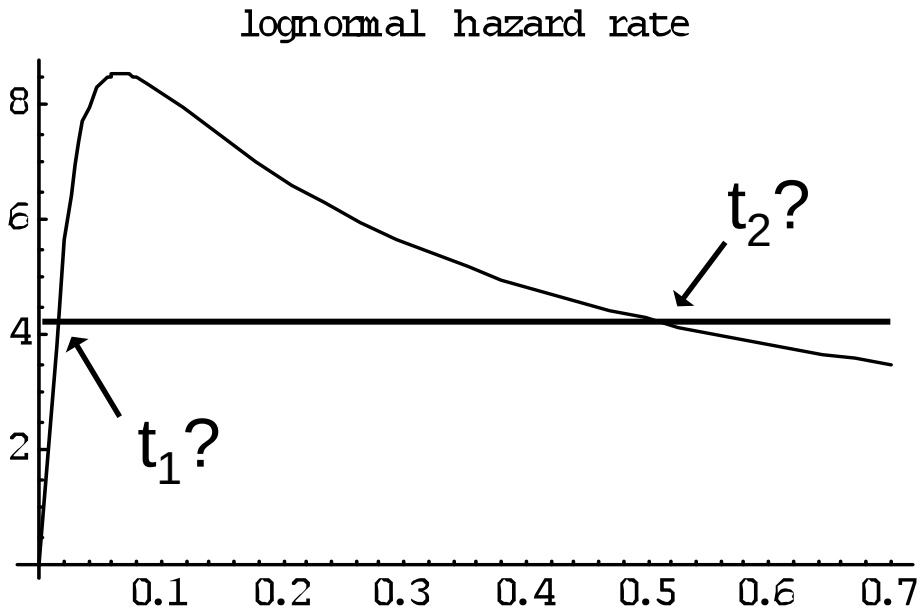
so, find the equi-hazard points!

local extremes: equi-hazard points

for one restart, find t_1 and t_2 such that:

- $t_1 + t_2 = d$
- $h(t_1) = h(t_2)$





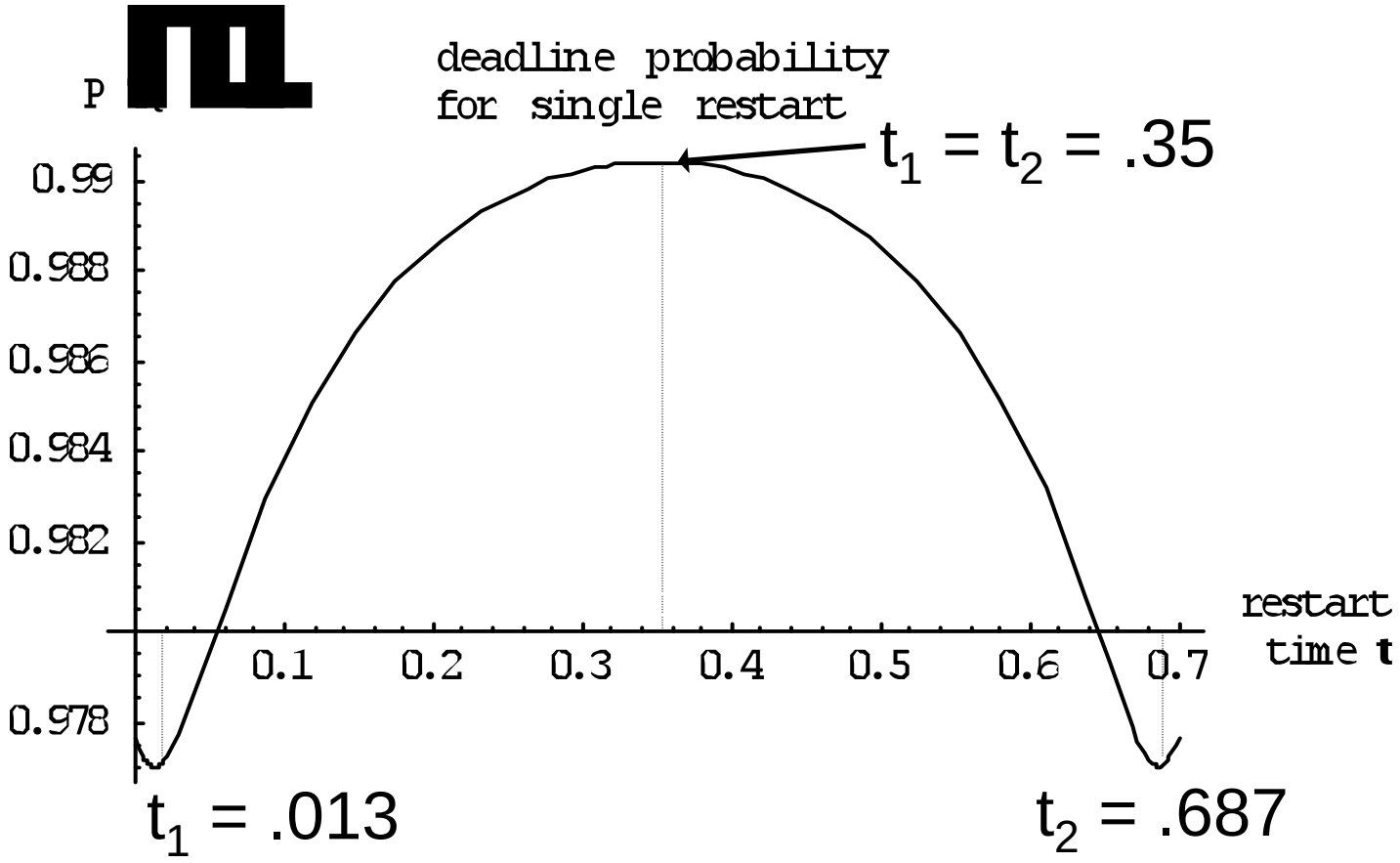
note, one solution:
$t_1 = t_2 = d / 2$

local extremes: equi-hazard points

# restarts	equi-hazard intervals	$P(T_{\{\tau\}} < d)$
0	—	0.978
1	0.35, 0.35	0.990
1	0.013, 0.687	0.977
2	0.23, 0.23, 0.23	0.993
2	0.019, 0.34, 0.34	0.990
2	0.013, 0.013, 0.674	0.976
3	0.175, 0.175, 0.175, 0.175	0.99374
3	0.024, 0.225, 0.225, 0.225	0.993
3	0.019, 0.019, 0.331, 0.331	0.989
3	0.013, 0.013, 0.013, 0.660	0.976
4	0.14, 0.14, 0.14, 0.14, 0.14	0.99366
⋮	⋮	⋮

lognormal task completion time, deadline d = .7

local extremes: equi-hazard points



lognormal task completion time, deadline d = 0.7

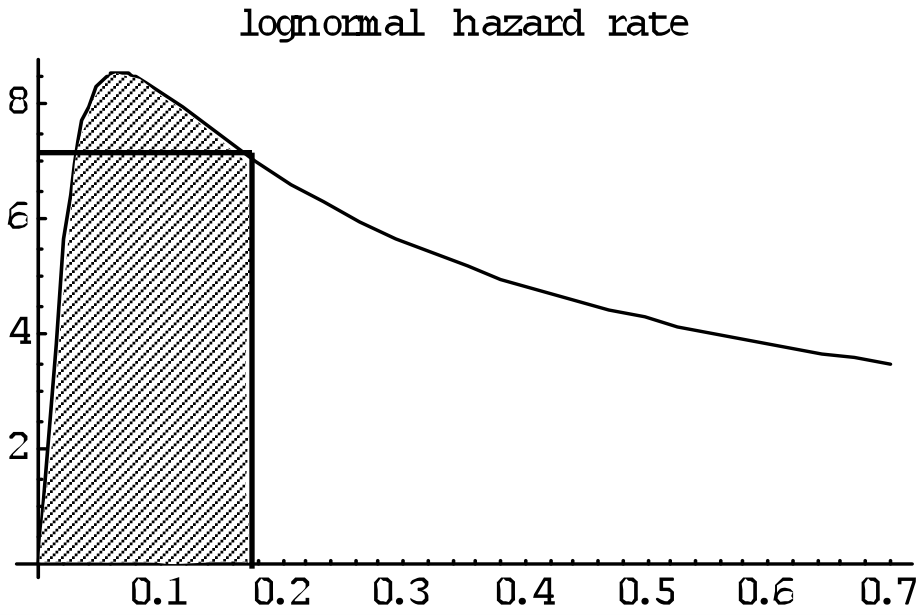
curve = rectangle hazard rule

relax integer requirement

derivative of $(1 – F(t))^{d/t}$ → optimal restart time:

$$t . h(t) = –\text{Log} (1 – F(t))$$

which is independent of d (!)



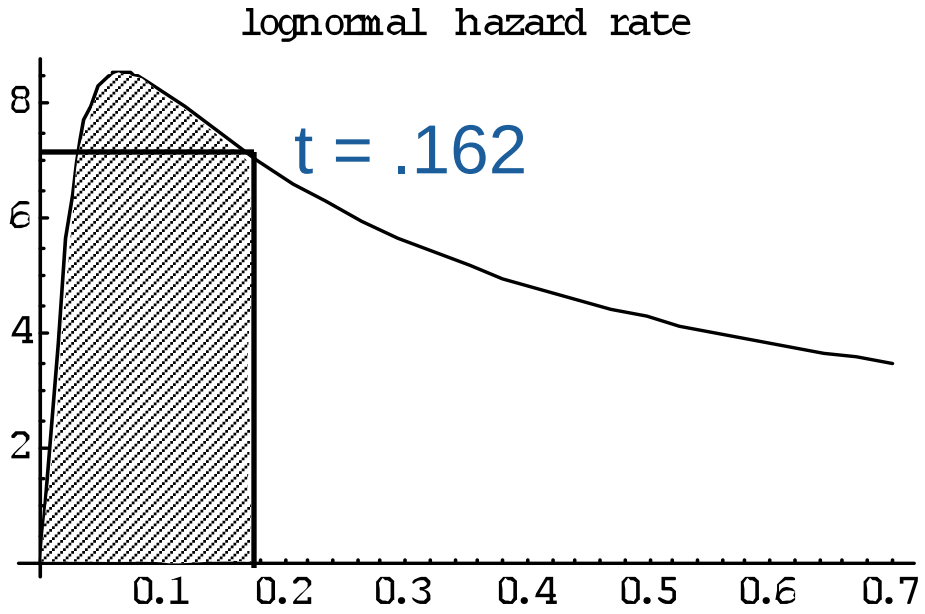
curve = rectangle hazard rule

example: t = .162 optimal (about 3.3 restarts)

integer optima, for d = .7:

3 restarts: t = d/4 = .175: F_t (d) = .99374

4 restarts: t = d/5 = .140: F_t (d) = .99366



conclusion model and optimal restart times

equi-hazard algorithm

- tailored to lognormal, finds global optimum, very fast if not too many restarts allowed
- nice…

equi-distant algorithm

- seems always (albeit unproven) to find global optimum, extremely fast
- nicer…

approximate optimum (curve = rectangle rule)

- independent of deadline d, no lognormal assumption, need a smart algorithm (Springer paper)
- nicest…

enter: engineering

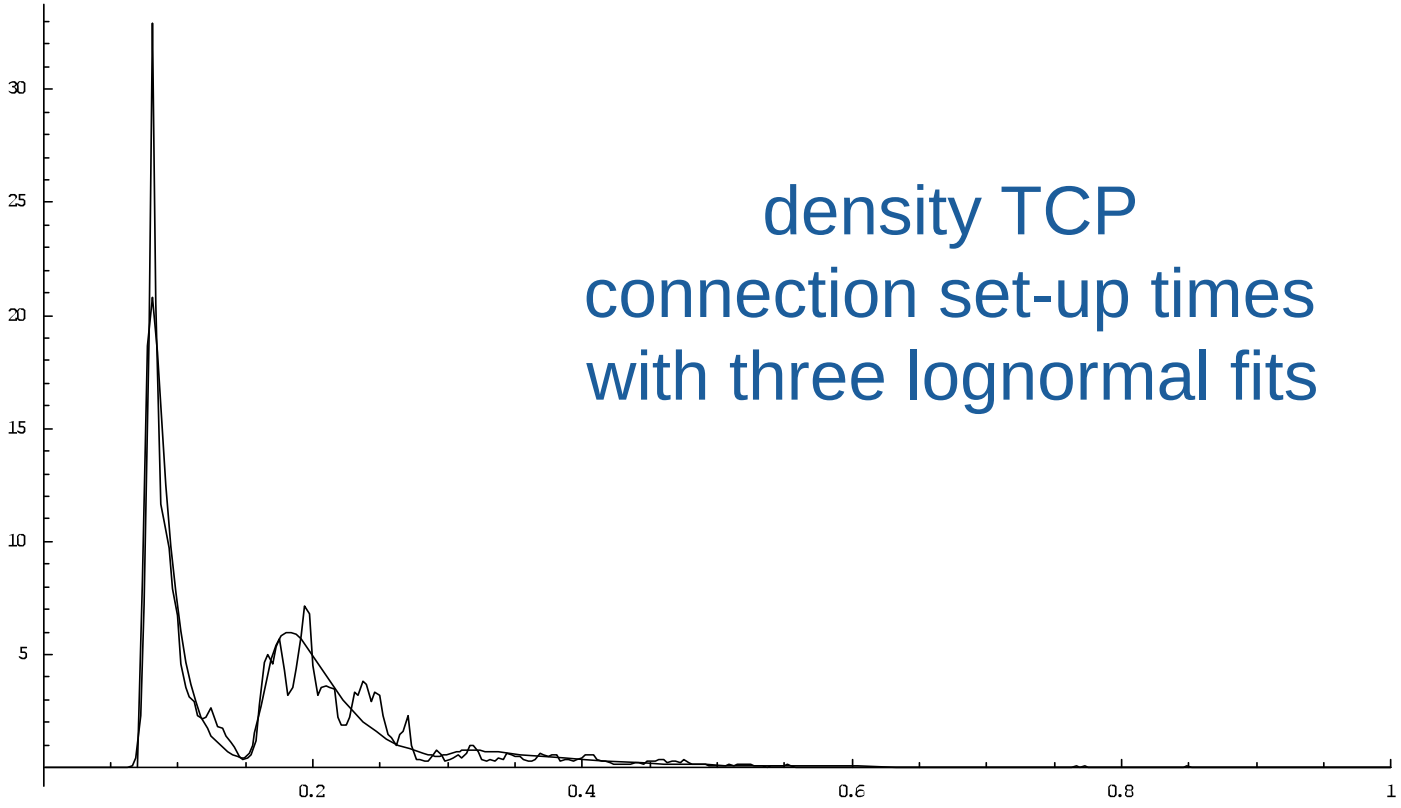
the math results are elegant, but:

- do the assumptions apply?
- do we gain much?
- is there better than black box? (ie. diagnosis)

let’s analyze some data

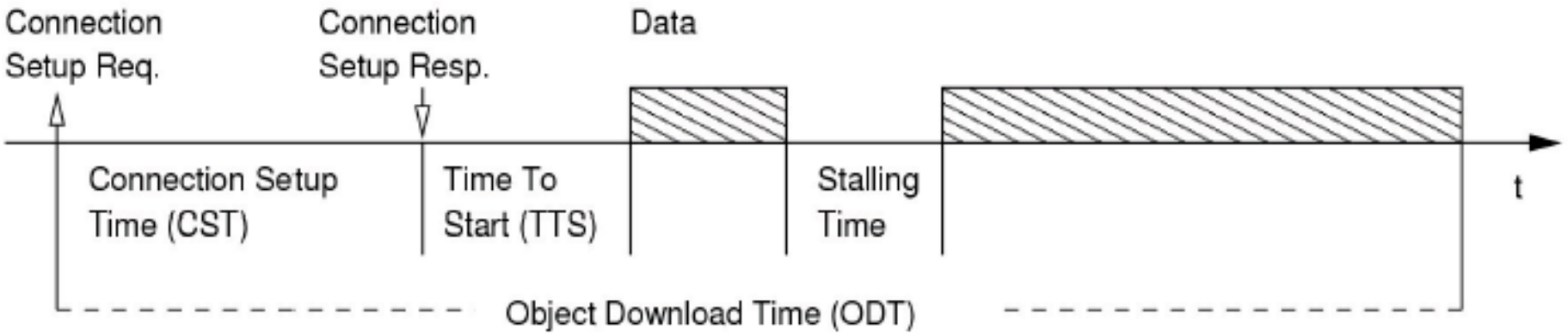
algorithm assumptions

- retries are independent and identically distributed
- retries preempt the previous try
- (`lognormal' shape of the hazard function)

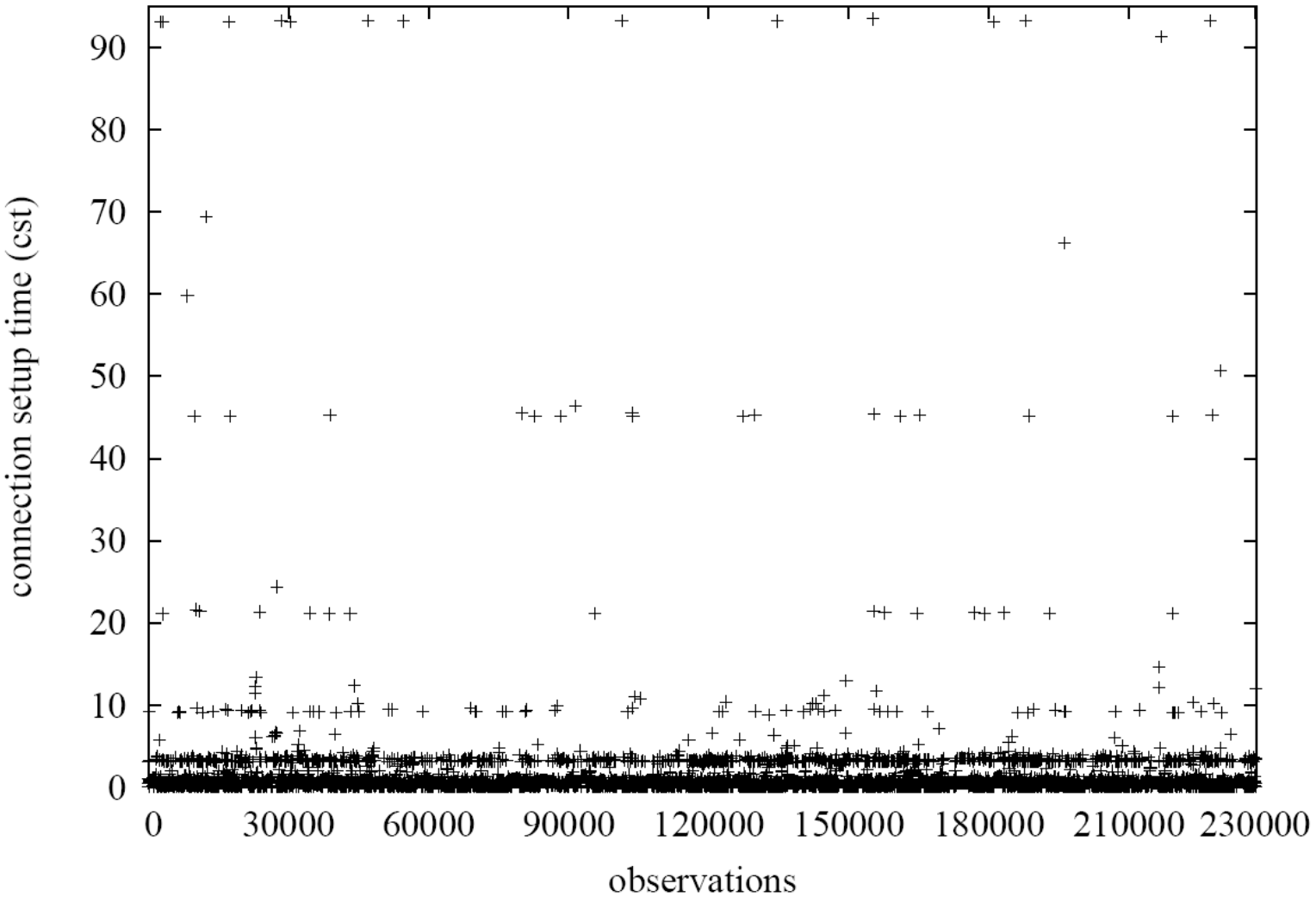


density TCP
connection set-up times
with three lognormal fits

HTTP Get

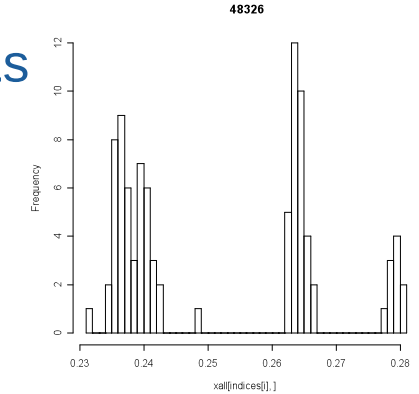


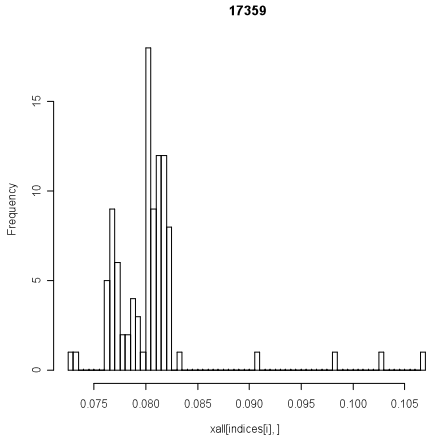
TCP connection set-up time

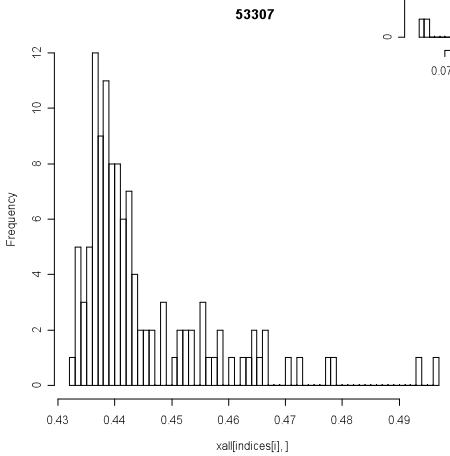


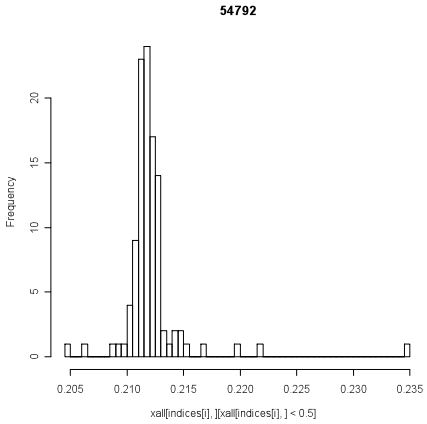
lognormal?

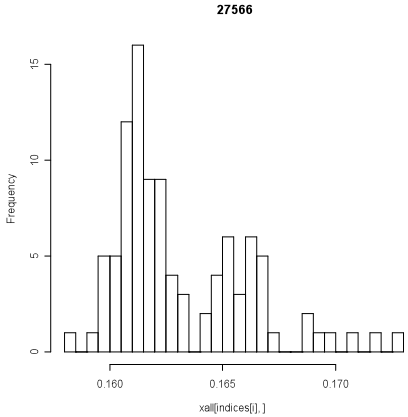
- lognormal holds across URLs
- algorithms require lognormal for individual URLs
- are these figures lognormal?
- note: approximation doesn't need it



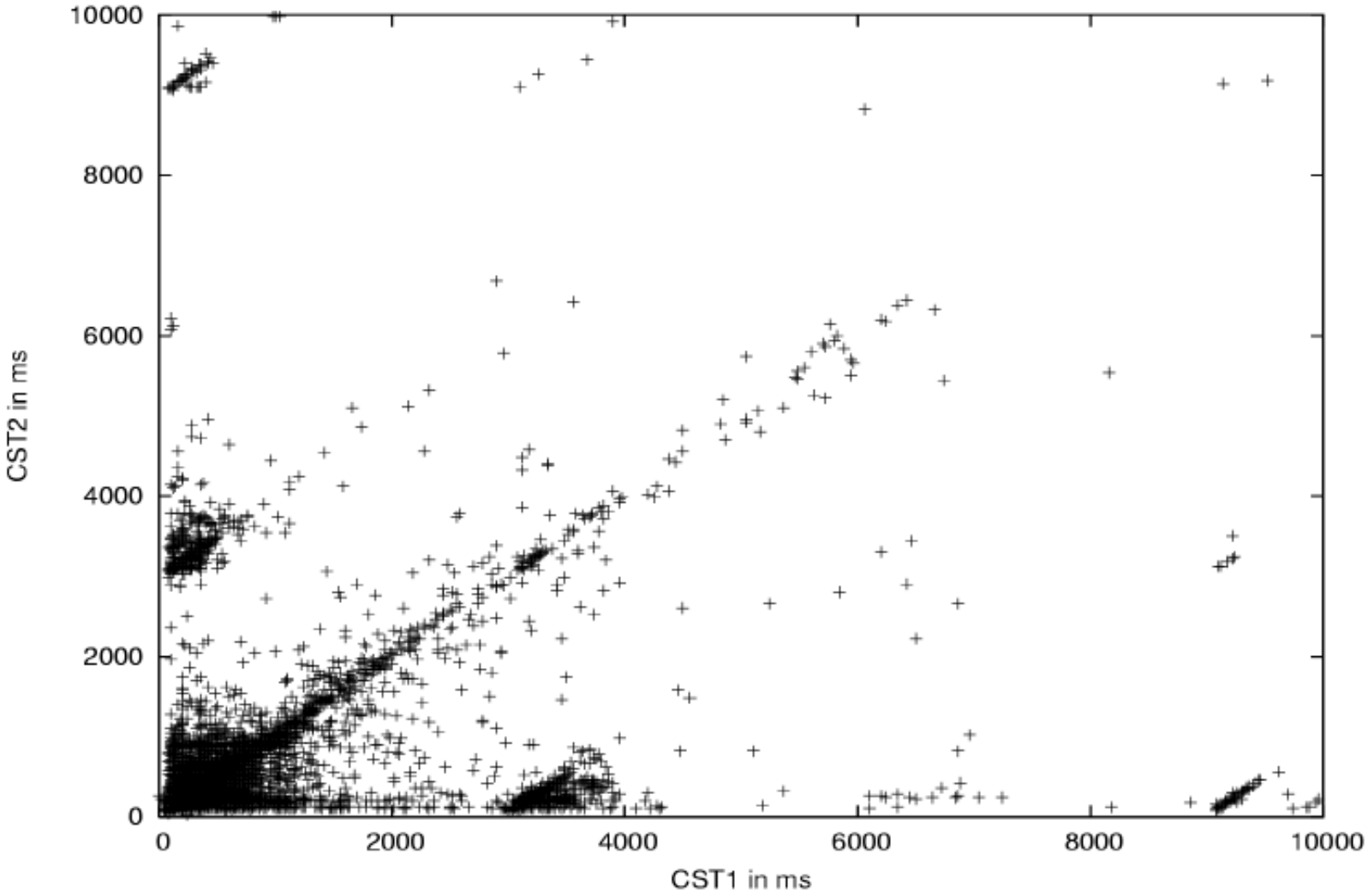




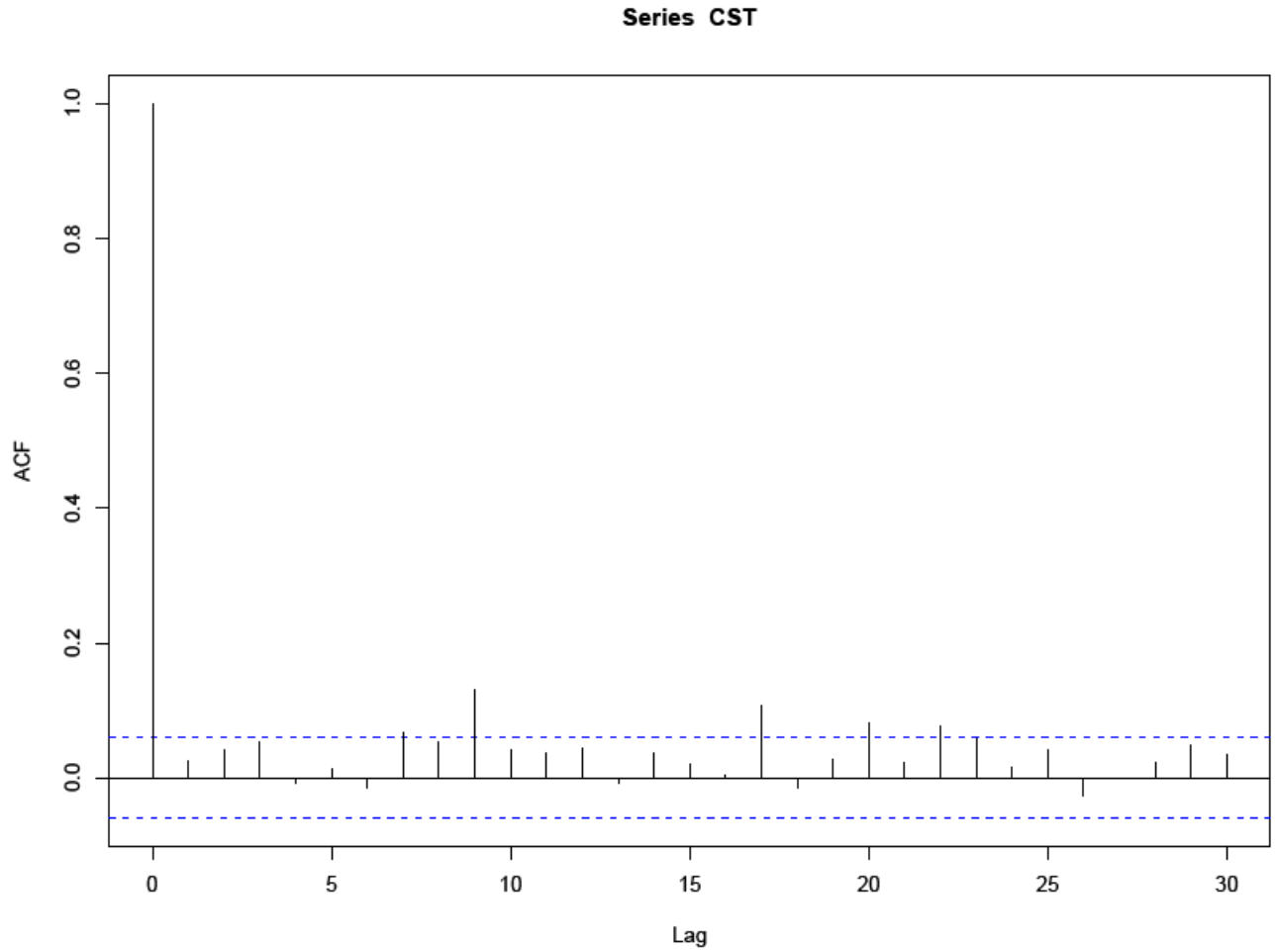




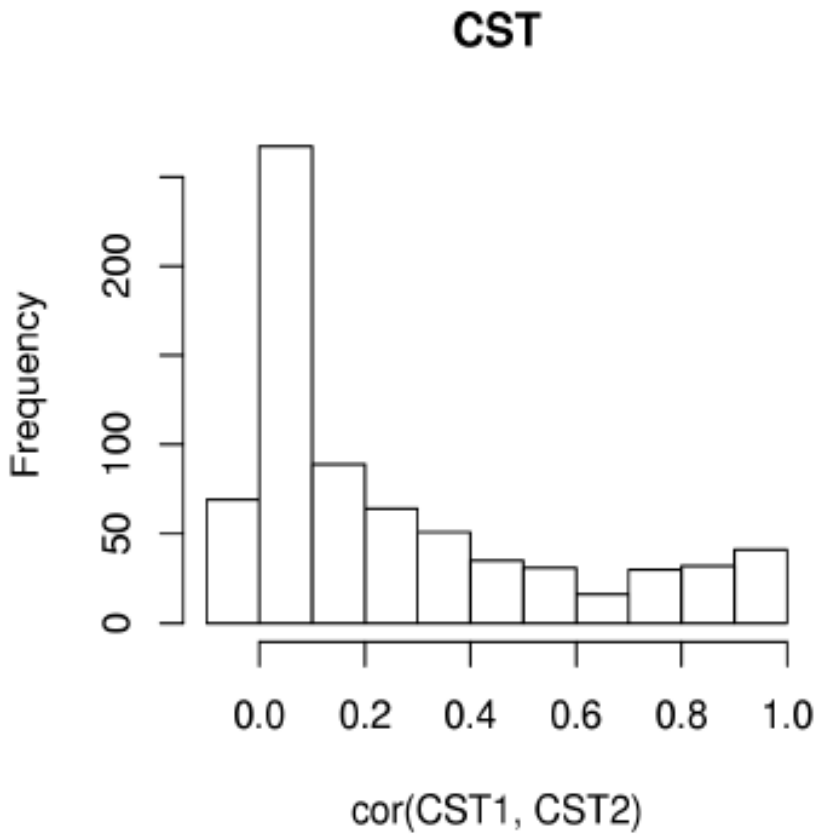
independence

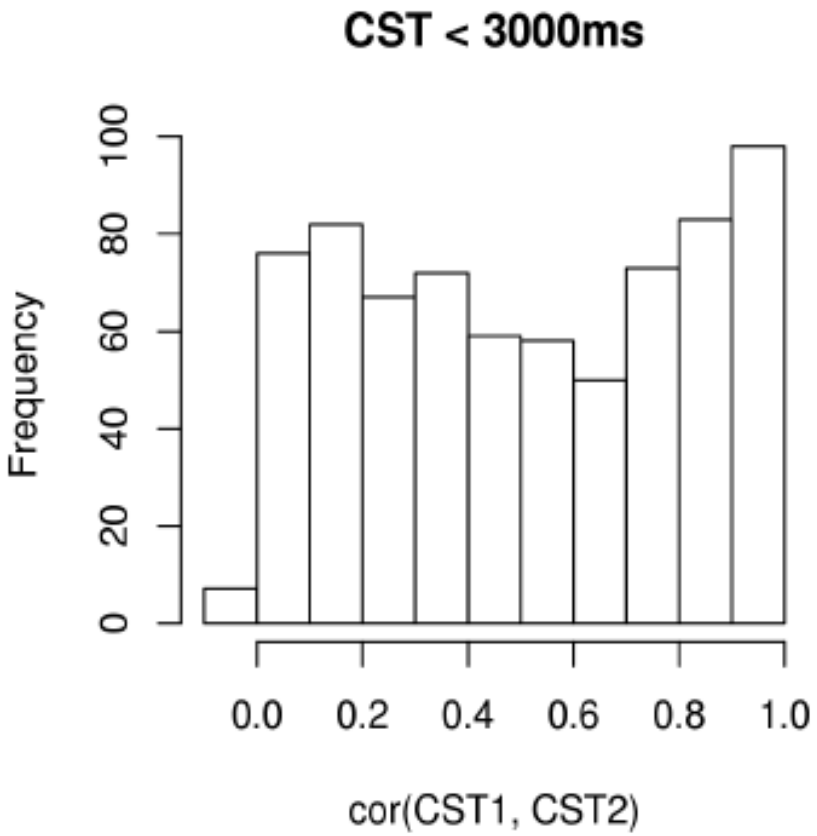


independence?
correlation of set-up times same URL

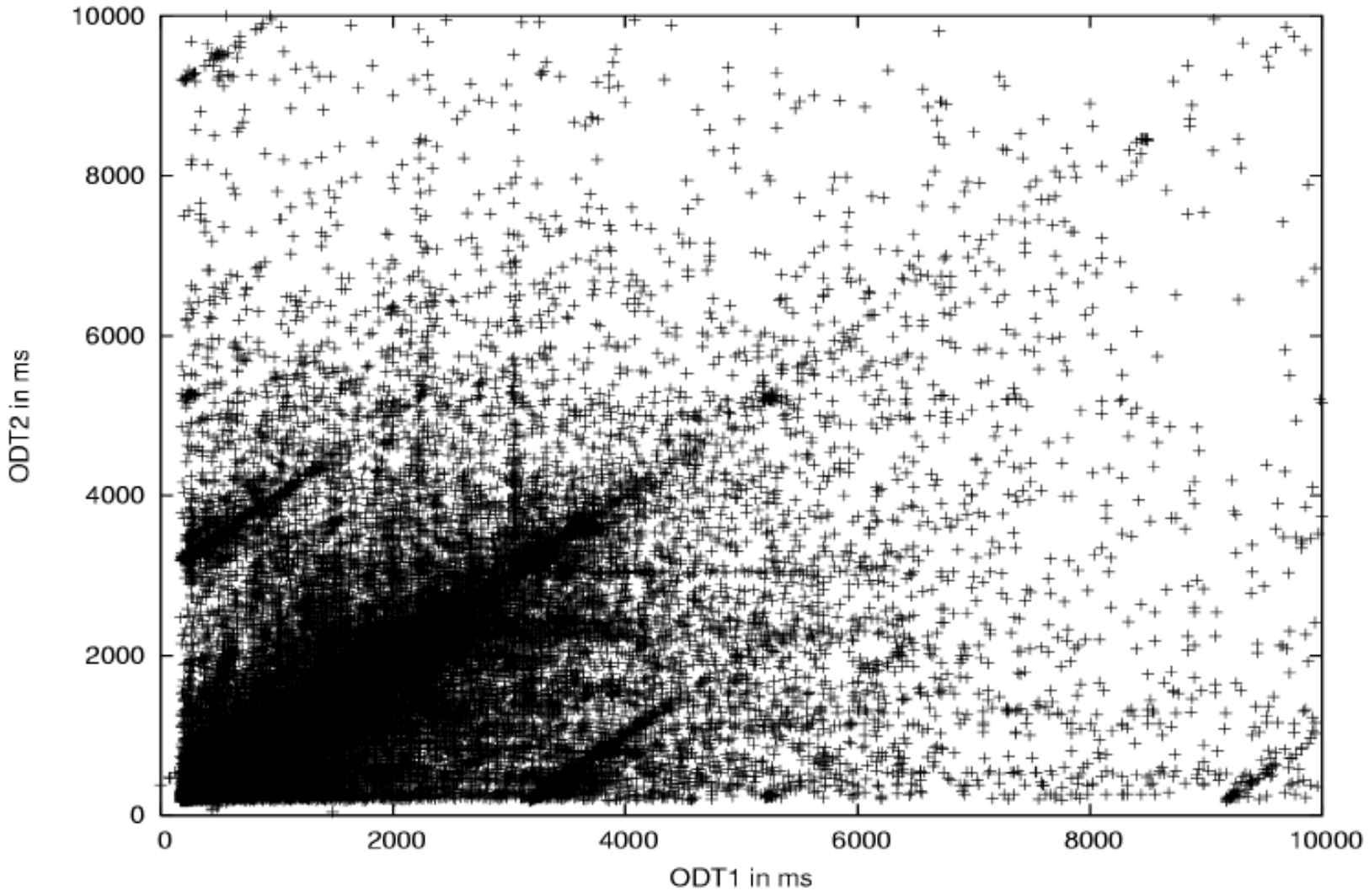


independence

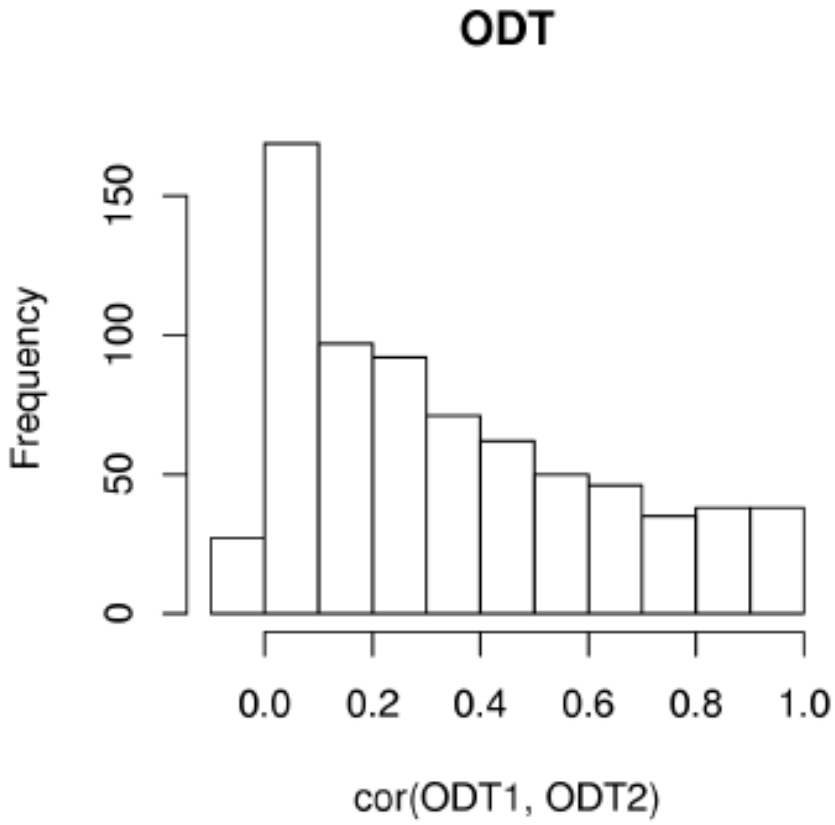


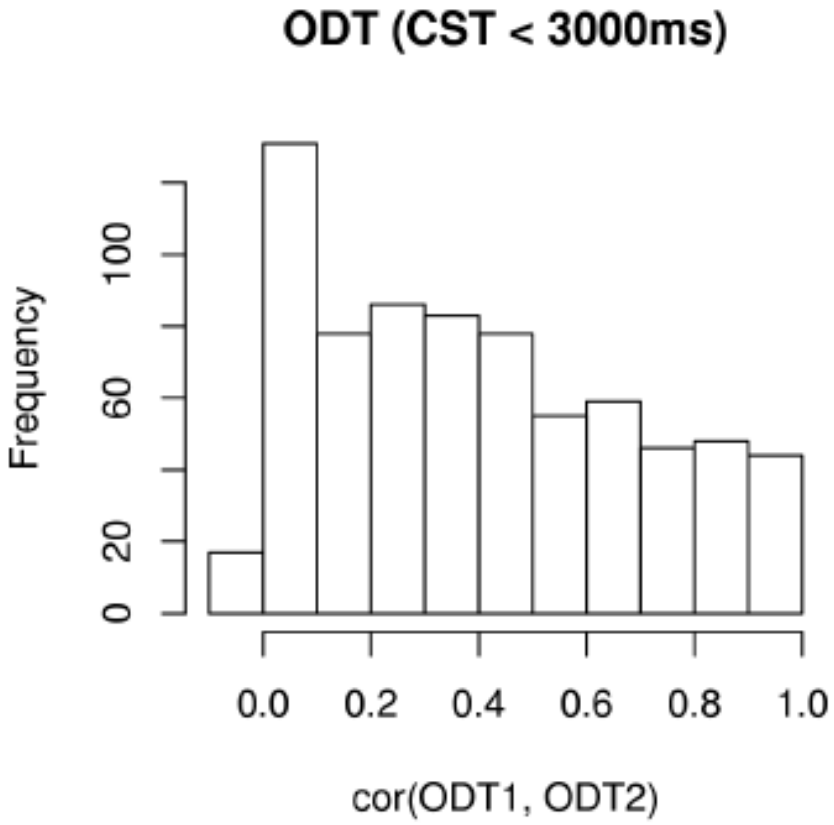


correlation page downloads



correlation page downloads





data

the low correlation remains surprising

- but is the reason TCP time-outs work

some questions:

- is there a lot to gain from setting time-outs based on run-time information (sort of failure diagnosis...)
- could we diagnose more and do better?
- how does it work in more complex systems?

complexity of WSRM

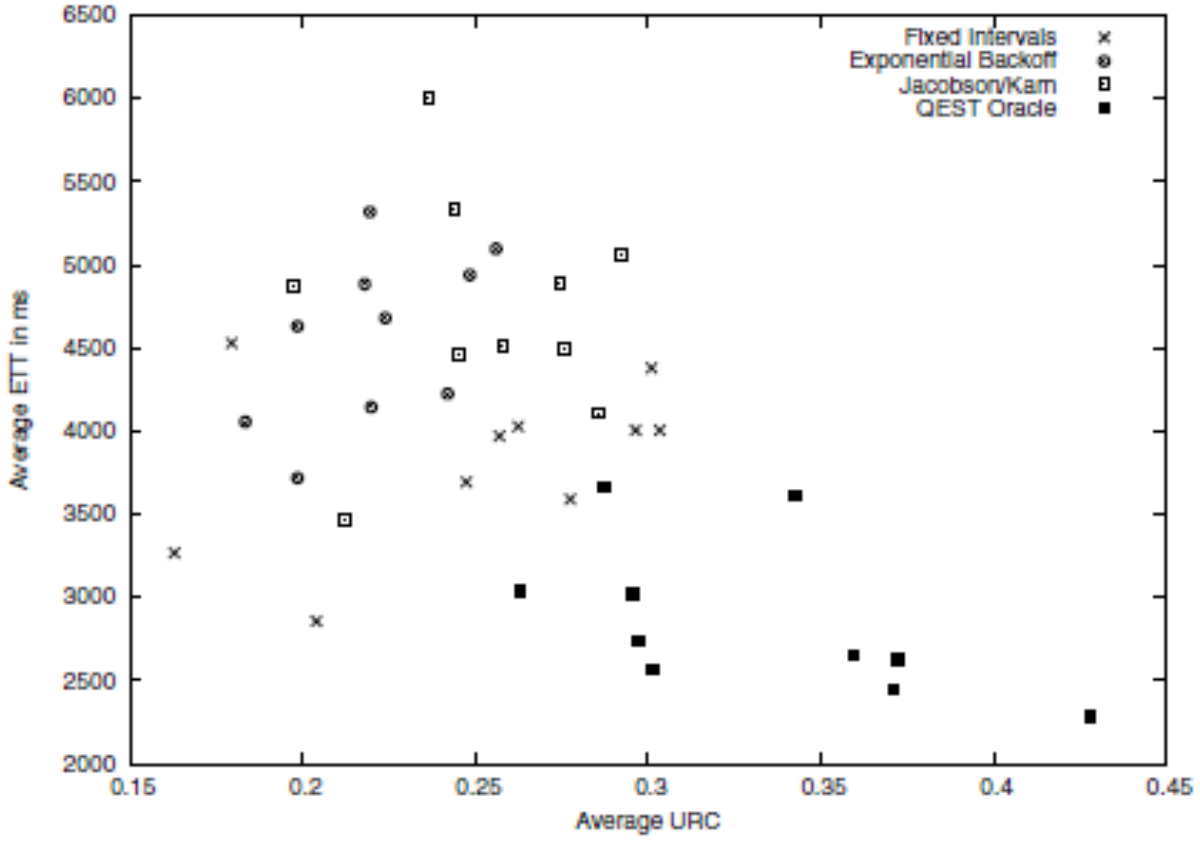


Figure 6. Performance with 10% packet loss (HTTP Transport).

complexity of WSRM

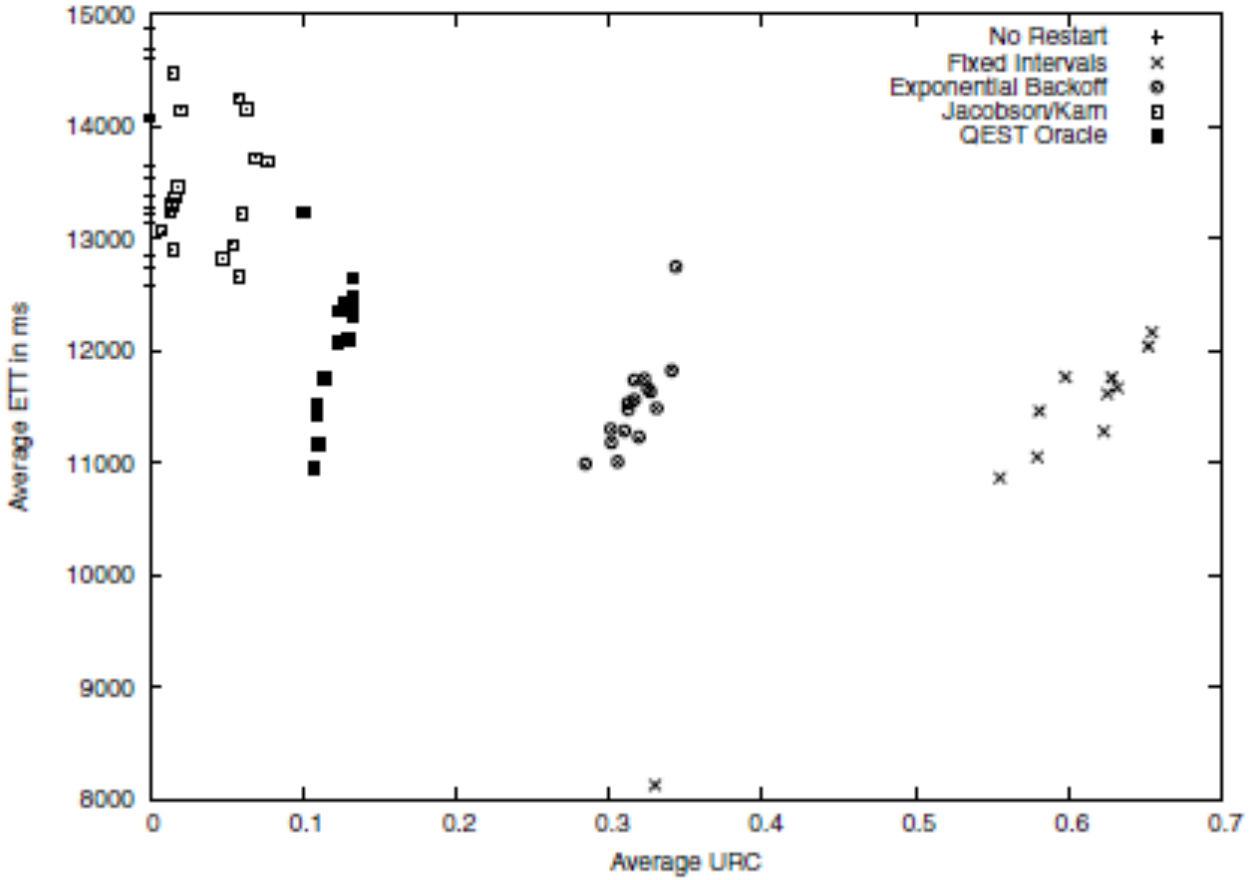


Figure 8. Performance for 60 s disruptions (Mail Transport).

failure diagnosis

some other observations:

- the improvement you can get in terms of average download time and making the deadline is minimal, since most tries don't benefit from restarting it

- so, instead of trying to improve overall, try to get rid of the worst cases

- the trick is: how to find identify you're looking at a 'worst case', what additional info could identify this?

suggestions?

the general case

for WSRM, and even web page downloads, the system is extremely complicated:

- retries at different levels
- other interfering fault tolerance, load balancing, rerouting, ..., mechanisms

how much can you do at the endpoint, and how can diagnosis help?

conclusion

we looked at retries, did some fun math

retries are surprisingly successful in the Internet

to exploit the power of retries fully, there are challenges:

- need (much) better diagnosis
- deal with interleaving mechanisms...

instead, simplify the engineering problem:

- need clever methods that diagnose and tolerate worst cases