Performance Modeling of Computer Systems and Networks

Prof. Vittoria de Nitto Personè

The multi-server queue

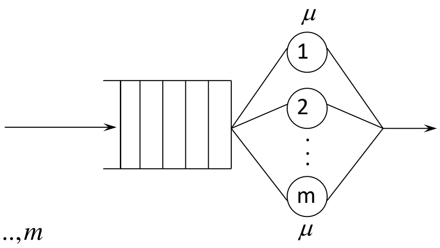
Università degli studi di Roma Tor Vergata
Department of Civil Engineering and Computer Science Engineering

1

Erlang, 1917

M/M/m abstract scheduling

$$E(N_Q)_{Erlang}$$



$$p(n) = \begin{cases} \frac{1}{n!}(m\rho)^n p(0) & for\ n = 1,\ldots,m \\ \frac{m^m}{m!}\rho^n p(0) & for\ n > m \end{cases}$$

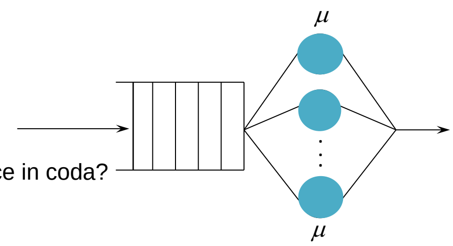
ci sono 'n' job

$$p(0) = \left[\sum_{i=0}^{m-1}\frac{(m\rho)^i}{i!} + \frac{(m\rho)^m}{m!(1-\rho)}\right]^{-1}$$

probabilità che il sistema sia vuoto

2

The *Erlang*-C formula



probabilità che quando un job arriva finisce in coda?

$$P_Q \cong \Pr\{n \geq m\} = \sum_{n=m}^{\infty} p(n)$$

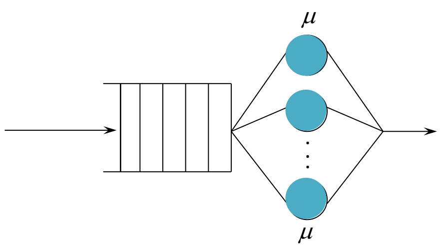
$$= \sum_{n=m}^{\infty} \frac{m^m}{m!} \rho^n p(0) = \frac{m^m}{m!} p(0) \sum_{n=m}^{\infty} \rho^n$$

$$= \frac{m^m}{m!} p(0) \sum_{n=0}^{\infty} \rho^{n+m} = \frac{m^m}{m!} p(0) \rho^m \sum_{n=0}^{\infty} \rho^n$$

$$\frac{1}{1-\rho}$$

serie nota

The *Erlang*-C formula



probabilità che siano tutti pieni,
dipende da 'm' e da 'rho'

$$P_Q = \frac{(m\rho)^m}{m!(1-\rho)} p(0)$$

$$E(N_Q)_{Erlang} = P_Q \frac{\rho}{1-\rho}$$

simile al caso servente singolo

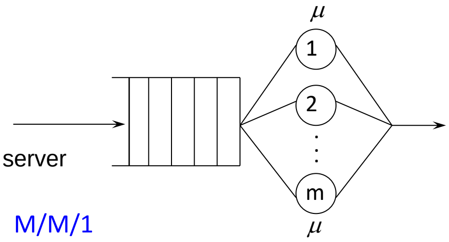
$$E(N_S) = P_Q \frac{\rho}{1-\rho} + m\rho$$

sommo quelli serviti mediamente

Little's law

$$E(T_Q) = \frac{E(N_Q)}{\lambda} \qquad E(T_Q) = P_Q \frac{\rho}{\lambda(1-\rho)} = \frac{P_Q E(S)}{1-\rho}$$

The *Erlang* formula



tempo medio per liberare uno qualsiasi dei server

M/M/m

$$E(T_Q)_{Erlang} = \frac{P_Q E(S)}{1-\rho}$$

M/M/1

$$E(T_Q)_{KP} = \frac{\rho E(S)}{1-\rho} = \frac{E(S_{rem})}{1-\rho}$$

tempo per far sì che se ne liberi uno, devo metterci lei!

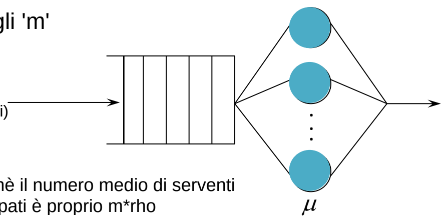
$$E(S) = \frac{E(S_i)}{m}$$

5

The Multi Server Queue

numero server occupati tra gli 'm'

c ≅ busy servers



(quando non sono tutti pieni) + (quando sono tutti pieni)

$$E(c) = \sum_{n=0}^{m-1} np(n) + \sum_{n=m}^{\infty} mp(n) = m\rho$$

perchè il numero medio di serventi occupati è proprio m*rho

definizione di Pq della C-Erlang (prob tutti occupati)

$$\rho = \sum_{n=0}^{m-1} \frac{n}{m} p(n) + \sum_{n=m}^{\infty} p(n) = \sum_{n=0}^{m-1} \frac{n}{m} p(n) + P_Q \Rightarrow \rho \geq P_Q$$

dato un certo carico, lambda e mu, la probabilità che siano tutti pieni è più piccola della probabilità che sia pieno solo uno.

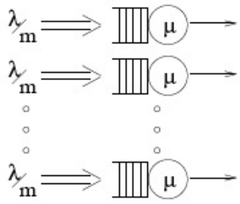
$$E(T_Q)_{Erlang} = \frac{P_Q E(S)}{1-\rho} \leq E(T_Q)_{KP} = \frac{\rho E(S)}{1-\rho}$$

6

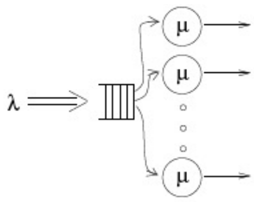
Nel multiserver ho più "sedie" su cui far sedere i job, se devo ottimizzare l'attesa, conviene distribuire la capacità, avere ad esempio 10 server meno potenti che uno 10 volte più potente, perchè dal punto di vista dell'attesa rho>Pq
Se devo minimizzare tempi di attesa è meglio la soluzione distribuita!

Server Organizations

m server disgiunti che si dividono il traffico, potrei avere punti vuoti e punti in cui si creano code.



m server divisi ma con traffico convogliato tutto insieme, quindi non posso avere punti vuoti e punti con code



server m volte piu veloce che si prende tutto lambda.

$$\rho = \frac{\lambda}{m\mu}$$

Communication systems

1
2
3
m

communication line

independent Poisson packet streams

each with an arrival rate of λ/m packets per second

the transmission time for each packet Exponential(1/μ)

Frequency-division Multiplexing

Statistical Multiplexing

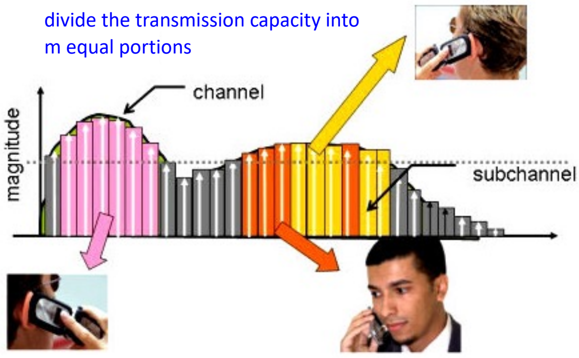
Communication systems

Frequency-division Multiplexing

divide the transmission capacity into
m equal portions

1
2
3
m

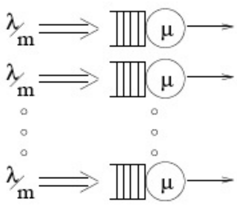

separated streams



tengo gli 'm' flussi separati quando si divide la capacità trasmissiva in 'm' porzioni uguali.

Communication systems

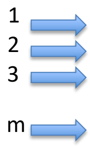
Frequency-division Multiplexing



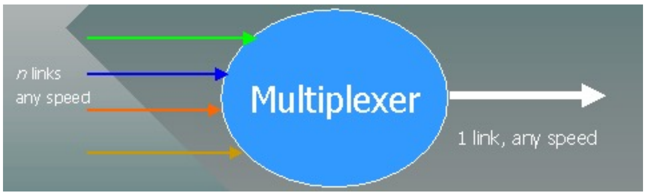
Communication systems

Statistical Multiplexing

keep the transmission capacity as a whole



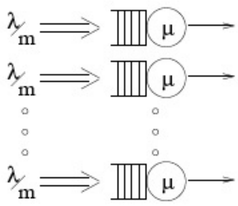
merge into a single stream



Qui gli 'm' flussi vengono convogliati in un unico flusso, fa un 'merge' e li manda su un unico link.

Communication systems

Frequency-division Multiplexing

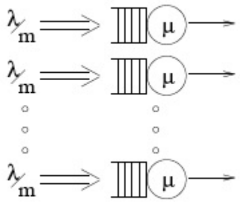


Statistical multiplexing

How do the two approaches compare with respect to mean response time?

Communication systems

Frequency-division Multiplexing



Statistical multiplexing

$$E(T_S) = \frac{\rho E(S)}{1-\rho} + E(S) = \frac{E(S)}{1-\rho}$$

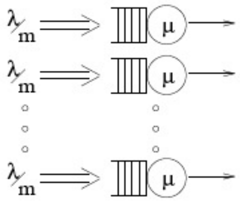
$$E(T_S) = \frac{1}{\mu\left(1-\frac{\lambda}{\mu}\right)} = \frac{1}{\mu-\lambda}$$

M/M/1

$\lambda \quad \mu$

Communication systems

Frequency-division Multiplexing



Statistical multiplexing

$$E(T_S)^{FDM} = \frac{1}{\mu-\frac{\lambda}{m}} = \frac{m}{m\mu-\lambda}$$

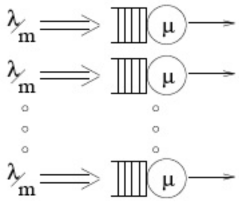
$$E(T_S)^{SM} = \frac{1}{m\mu-\lambda}$$

FDM shows a response time *m* times greater then for *SM* !

FDM però garantisce a ciascun flusso una specifica frequenza di servizio!

Communication systems

Frequency-division Multiplexing



1 QoS guarantee for each stream: a specific service rate to each stream

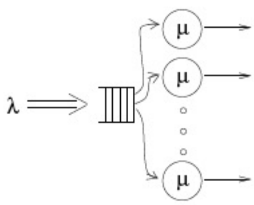
Statistical multiplexing

No QoS guarantee

2 If the original m streams were very regular (not Poisson), i.e., they were much less variable than Poisson, by merging them, we introduce lots of variability into the arrival stream. This leads to problems if the application requires a low variability in delay, e.g., voice or video.

15

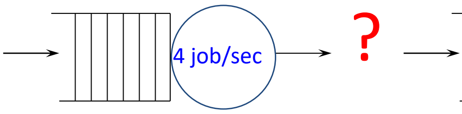
Server Organizations

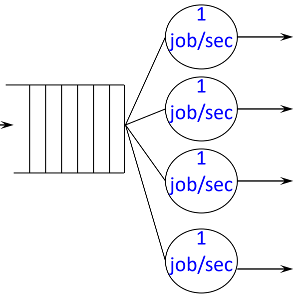


16

Modelling power: Design tool

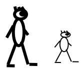
Design Example 4: One machine or many?





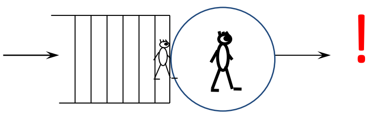
Goal: minimizing mean response time

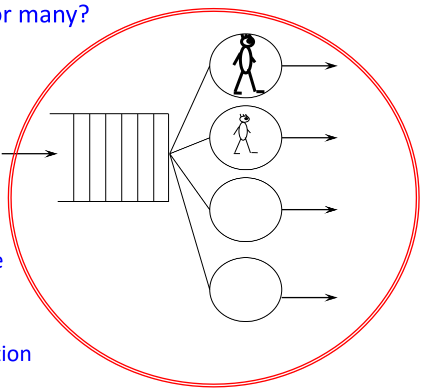
Assumption: jobs *non-preemptible*
each job must be run to completion

high variability 

Modelling power: Design tool

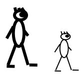
Design Example 4: One machine or many?





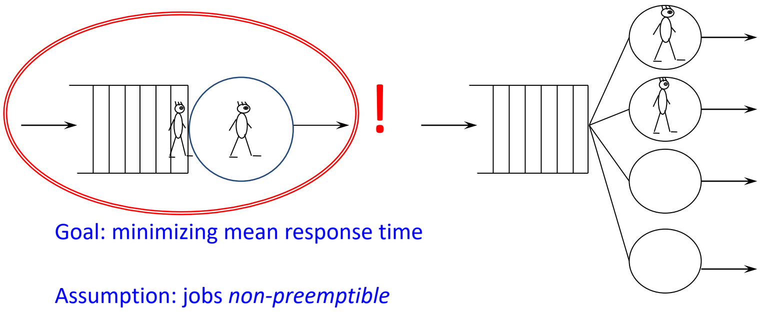
Goal: minimizing mean response time

Assumption: jobs *non-preemptible*
each job must be run to completion

high variability 

Modelling power: Design tool

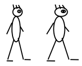
Design Example 4: One machine or many?



Goal: minimizing mean response time

Assumption: jobs *non-preemptible*
each job must be run to completion

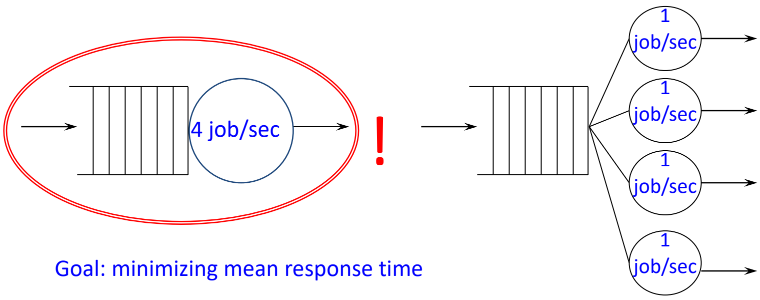
low variability



19

Modelling power: Design tool

Design Example 4: One machine or many?

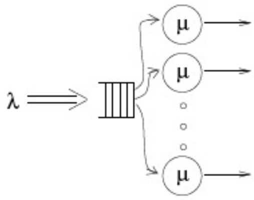


Goal: minimizing mean response time

Assumption: jobs *preemptible*
each job can be stopped and restarted where they left off

20

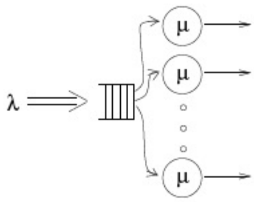
Server Organizations



$$E(T_Q)_{Erlang} = \frac{P_Q E(S)}{1-\rho} \qquad E(T_Q)_{KP} = \frac{\rho E(S)}{1-\rho}$$

$\rho \geq P_Q$ → from the waiting time perspective the distributed capacity solution produces an improvement in the user perceived QoS

Server Organizations

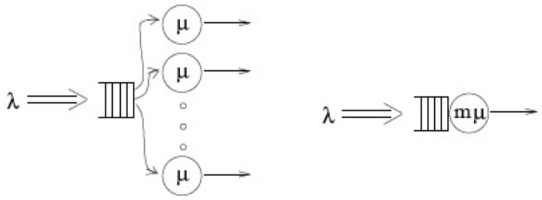


What about the response time perspective??

$$E(T_S)_{Erlang} = \frac{P_Q E(S)}{1-\rho} + E(S_i) \qquad E(T_S)_{KP} = \frac{\rho E(S)}{1-\rho} + E(S)$$

$$E(S_i) = \frac{1}{\mu} = m\frac{1}{m\mu} = mE(S)$$

Server Organizations



What about the response time perspective??

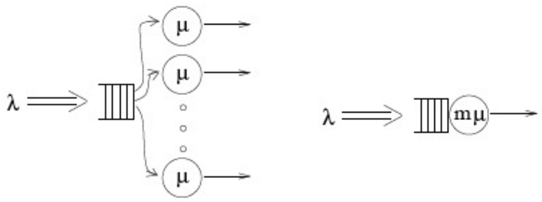
$$E(T_S)_{Erlang} = \frac{P_Q E(S)}{1-\rho} + mE(S)$$

Decreasing less than linear $\wedge$ $\vee$ linear growth

$$E(T_S)_{KP} = \frac{\rho E(S)}{1-\rho} + E(S)$$

23

Server Organizations



Performance goal:

Waiting time perspective — Distributed capacity

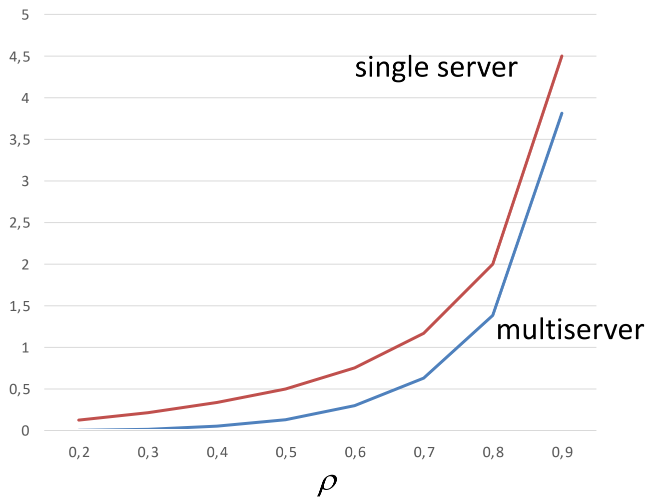
Response time perspective

- $\rho \to 0$: distrib. capac. gives an *m* times slower organization
- $\rho \to 1$: approximately the same response time

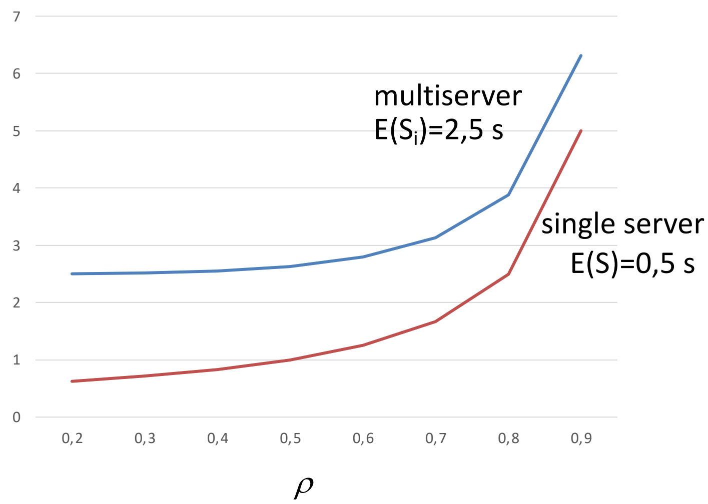
24

Waiting time perspective

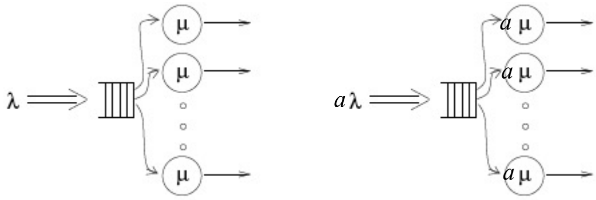
E(S)=0,5 s



Response time perspective



Scaling factor



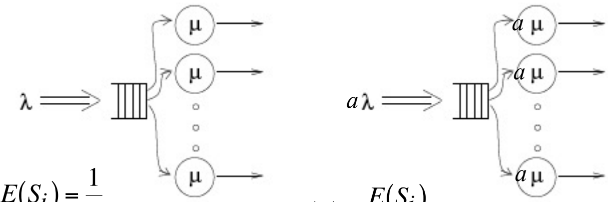
What about waiting and response time?

$$\rho = \frac{\lambda}{m\mu} \qquad \rho = \frac{a\lambda}{ma\mu} = \frac{\lambda}{m\mu}$$

$$E(S_i) = \frac{1}{\mu} \qquad E(S_i) = \frac{1}{a\mu} \qquad E(S) = \frac{E(S_i)}{m}$$

27

Scaling factor

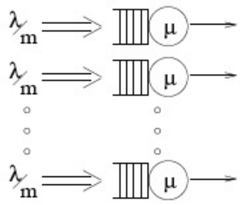


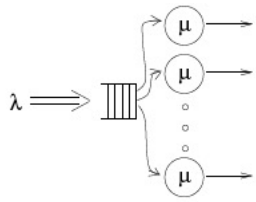
$$E(S_i) = \frac{1}{\mu} \qquad E(S) = \frac{E(S_i)}{m} \qquad E(S_i) = \frac{1}{a\mu}$$

Mean waiting time

$$E(T_Q)_{m,a} = \frac{P_Q E(S)_{m,a}}{1-\rho} = \frac{P_Q}{ma\mu(1-\rho)} = \frac{1}{a}\frac{P_Q E(S)_{m,1}}{(1-\rho)} = \frac{1}{a}E(T_Q)_{m,1}$$

28





$\lambda = 4$ j/s,   m=4, μ=1.5 j/s E(S)=0.666667 s

$$E(S) = \frac{E(S_i)}{4} = 0.1667$$

$$\rho = 0.666667$$

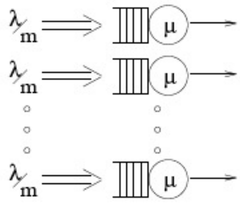
$$E(T_S) = \frac{1}{\mu - \lambda} = 2\ s$$

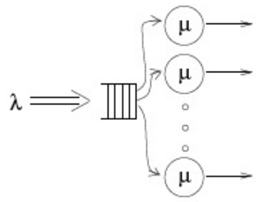
$$E(T_Q) = \frac{\rho E(S)}{1-\rho} = 1.3336$$

$$\rho = 0.666667$$

$$p(0) = \left[\sum_{i=0}^{3} \frac{(4\rho)^i}{i!} + \frac{(4\rho)^4}{4!(1-\rho)}\right]^{-1}$$

$$= \left[1 + 4\rho + \frac{(4\rho)^2}{2} + \frac{(4\rho)^3}{6} + \frac{(4\rho)^4}{24(1-\rho)}\right]^{-1} = 0.059857$$





$\lambda = 4$ j/s,   m=4, μ=1.5 j/s E(S)=0.666667 s

$$E(S) = \frac{E(S_i)}{4} = 0.1667$$

$E(S_i)$=0.666667 s

$$\rho = 0.666667$$

$$E(T_S) = \frac{1}{\mu - \lambda} = 2\ s$$

$$E(T_Q) = \frac{\rho E(S)}{1-\rho} = 1.3336$$

$$\rho = 0.666667$$

$$P_Q = \frac{(4\rho)^4}{4!(1-\rho)} p(0) = 0.37847$$

$$E(T_S) = \frac{P_Q E(S)}{1-\rho} + E(S_i) = 0.855992$$

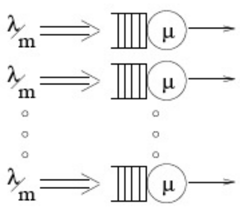
$$E(T_Q) = 0.189292$$

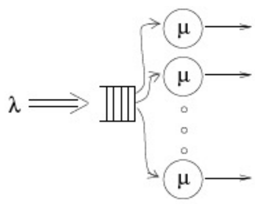
$\lambda = 4$ j/s, $m\mu = 4\text{x}1.5 = 6$ j/s E(S)=0.166667 s

$\rho = 0.666667$

$$E(T_S) = \frac{1}{m\mu - \lambda} = 0.5$$

$$E(T_Q) = 0.3334$$





$\rho = 0.666667$

$$E(T_S) = \frac{1}{\mu - \lambda} = 2 \qquad E(T_S) = 0.855992 \qquad \boxed{E(T_S) = \frac{1}{\mu - \lambda} = 0.5\ s}$$

$$E(T_Q) = \frac{\rho E(S)}{1-\rho} = 1.3336 \qquad \boxed{E(T_Q) = 0.189292} \qquad E(T_Q) = 0.3334$$

Esercizio proposto: $\rho = 0.533334 \quad \rho = 0.8$