Performance Modeling of Computer Systems and Networks

Prof. Vittoria de Nitto Personè

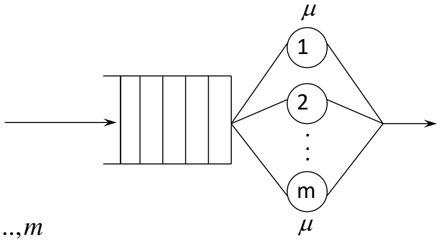
The multi-server queue

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

1

Erlang, 1917
M/M/m abstract scheduling

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



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

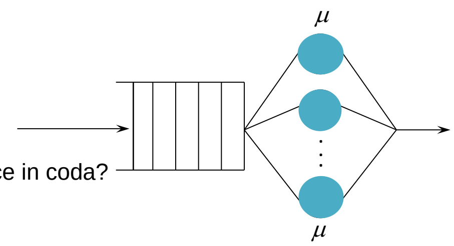
ci sono 'n' job

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

probabilità che il sistema sia vuoto

2

The *Erlang*-C formula



probabilità che quando un job arriva finisce in coda?

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

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

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

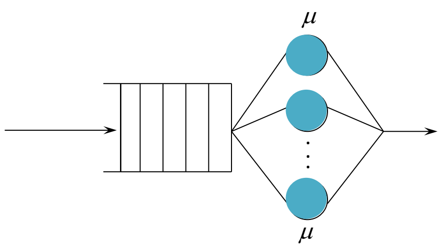
$$\frac{1}{1-\rho}$$

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

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

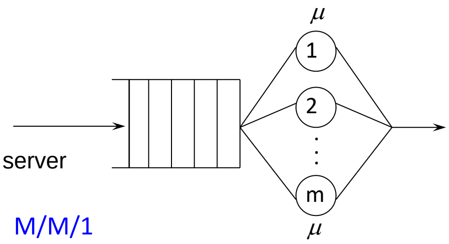
sommo quelli serviti mediamente

Little's law

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

NB: vale solo per distribuzioni esponenziali, mentre la KP è più generica

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!

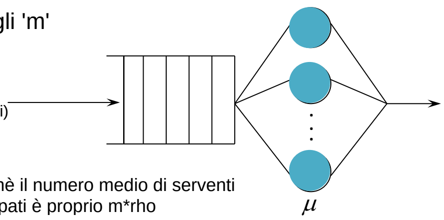
$$E(S) = \frac{E(S_i)}{m}$$

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)

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

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

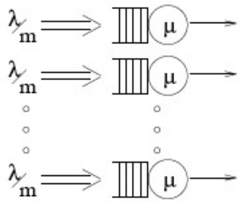
dato un certo carico, lambda e mu, la probabilità che siano tutti pieni è più piccola della probabilità che sia pieno solo uno.

6

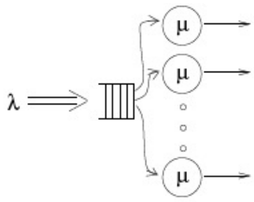
Nel multiserver ho più "sedie" su cui far sedere i job, se devo ottimizzare l'attesa, conviene distribuire la capacità, avere ad esempio 10 server meno potenti che uno 10 volte più potente, perchè dal punto di vista dell'attesa rho>Pq
Se devo minimizzare tempi di attesa è meglio la soluzione distribuita!

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($1/\mu$)

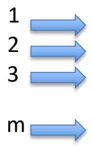
Frequency-division Multiplexing

Statistical Multiplexing

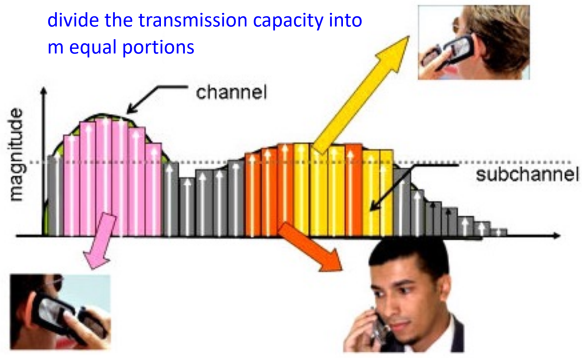
Communication systems

Frequency-division Multiplexing

divide the transmission capacity into
m equal portions



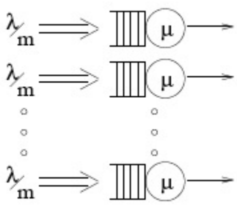
separated streams



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

Communication systems

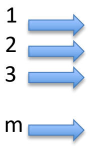
Frequency-division Multiplexing



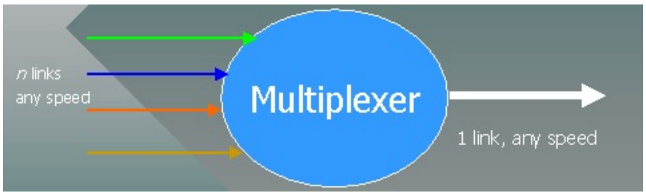
Communication systems

Statistical Multiplexing

keep the transmission capacity as a whole



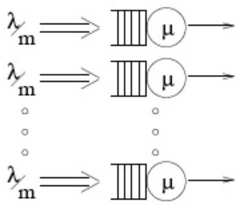
merge into a single stream



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

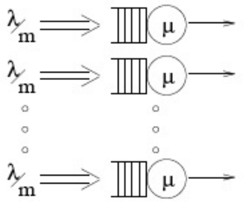


Statistical multiplexing

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

Communication systems

Frequency-division Multiplexing



Statistical multiplexing

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

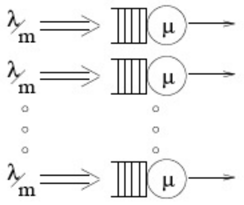
$$E(T_S) = \frac{1}{\mu\left(1-\frac{\lambda}{\mu}\right)} = \frac{1}{\mu-\lambda}$$

M/M/1

$\lambda \quad \mu$

Communication systems

Frequency-division Multiplexing



Statistical multiplexing

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

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

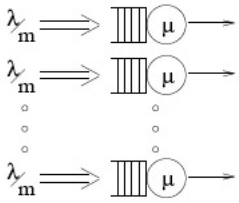
FDM shows a response time *m* times greater then for *SM* !

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

Sia 'C' la capacità operativa fisica, essa è fissa, non probabilistica Si esprime in [oper/sec].
C'è anche 'Z': la domanda di servizio media del job, espressa in [oper/job].
Quando parlavamo di E[S] 'univamo' questi due concetti, infatti E[S] = Z/C, espressa in [sec/job]
allora mu, che è l'inverso, è espresso in [job/sec]
Questo vuol dire che, se dovessero essere forniti C e Z, allora devo ricavarmi E[S]

Communication systems

Frequency-division Multiplexing



Statistical multiplexing

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

No QoS guarantee

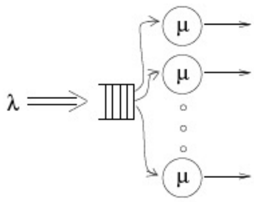
se avessi Poisson, riotterrei Poisson

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

11/04/2023

Server Organizations



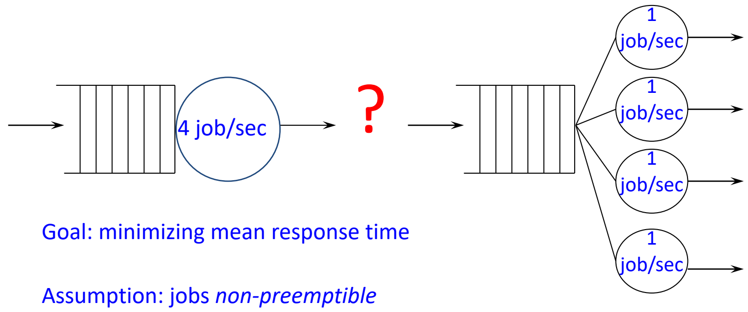
rho = rapporto frequenza arrivo / massima frequenza di servizio

devo sempre ragionare a parità di condizione!

16

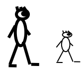
Modelling power: Design tool

Design Example 4: One machine or many?



Goal: minimizing mean response time

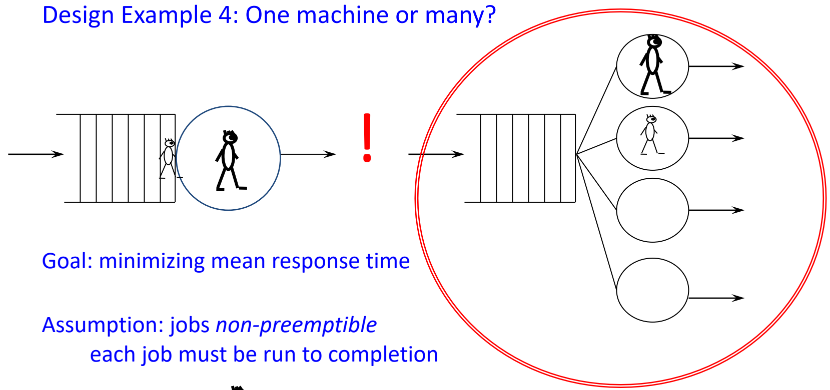
Assumption: jobs *non-preemptible*
each job must be run to completion

high variability 

17

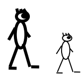
Modelling power: Design tool

Design Example 4: One machine or many?



Goal: minimizing mean response time

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

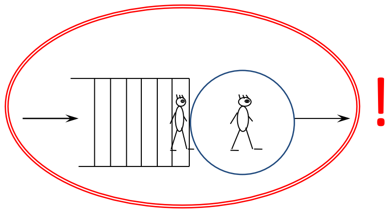
high variability 

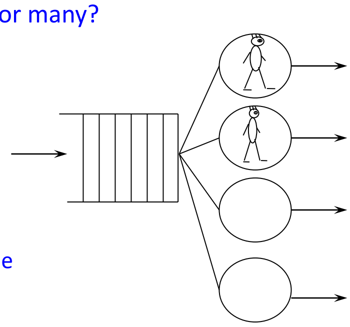
meglio dividere!

18

Modelling power: Design tool

Design Example 4: One machine or many?

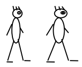




Goal: minimizing mean response time

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

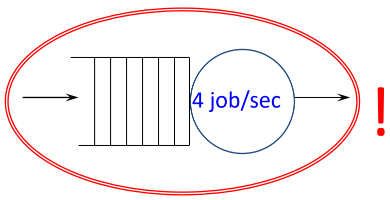
low variability

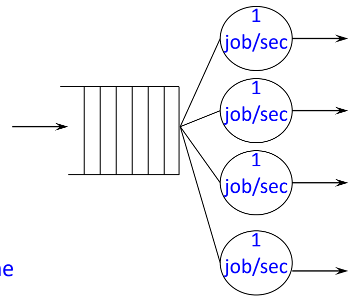


variabilità bassa, attesa proporzionale per tutti.
Meglio uno più potente.

Modelling power: Design tool

Design Example 4: One machine or many?





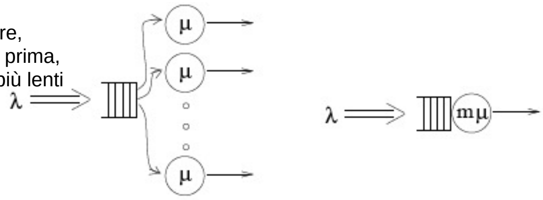
Goal: minimizing mean response time

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

Server Organizations

Stiamo parlando di esponenziale, perchè uso Erlang. La variabilità dell'esponenziale è medio/bassa.

Qui attesa minore,
i servizi iniziano prima,
anche se sono più lenti



Per l'organizzazione a sinistra, essendo un multiserver, devo usare Erlang, il quale modella tale organizzazione.
A destra, avendo un servente singolo, non serve erlang ma uso KP.
In entrambi i casi ho esponenziali però!

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

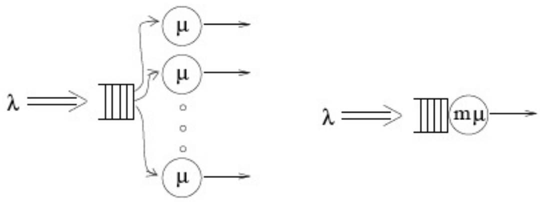
$\rho \geq P_Q$

rho, cioè prob che UNO SIA PIENO >= prob. che siano tutti pieni!

from the waiting time perspective the distributed capacity solution produces an improvement in the user perceived QoS

Server Organizations

a sinistra i serventi sono 'm' volte più lenti

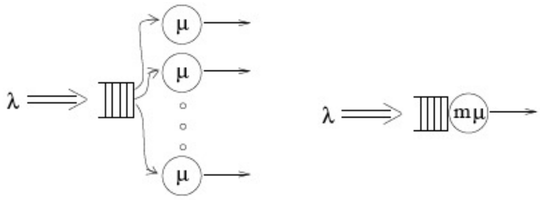


What about the response time perspective??

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

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

Server Organizations



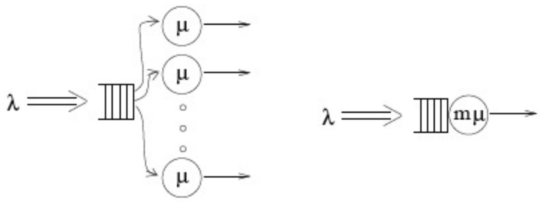
What about the response time perspective??

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

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

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

Server Organizations



Performance goal:

Waiting time perspective — Distributed capacity

Response time perspective

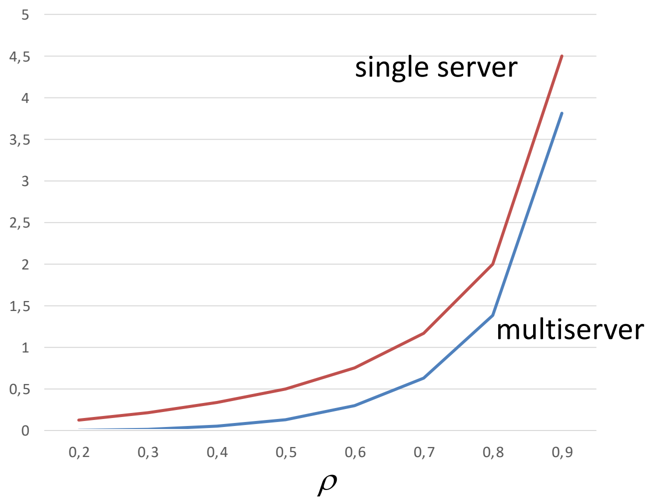
$\rho \to 0$ distrib. capac. gives an *m* times slower organization

$\rho \to 1$ approximately the same response time

I casi qui esaminati valgono solo per l'ESPONENZIALE, se così non fosse?

Waiting time perspective

E(S)=0,5 s



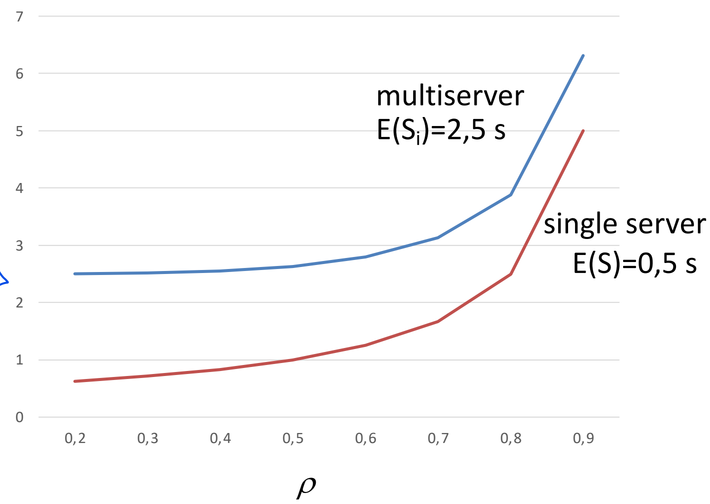
Notiamo come single server sia sempre sopra il multiserver, a conferma del fatto che il single server presenti tempi di attesa in coda maggiori

Prof. Vittoria de Nitto Personè 25

25

Response time perspective

Qui abbiamo che E[Si] = m*E[S] all'inizio, con rho che è vicino a 0, il single server è meglio, infatti il distribuito è "m" volte più lento



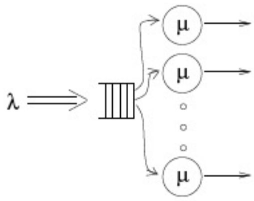
Con rho che si avvicina ad 1, i denominatori dei waiting time tendono a infinito, e quindi la differenza tra i tempi di servizio diventa insignificante, poichè E[Ts] = E[Tq] + E[s], con E[Tq] che cresce a dismisura.

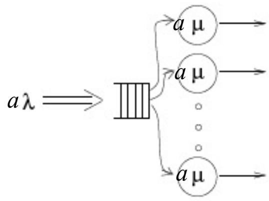
Prof. Vittoria de Nitto Personè 26

26

Scaling factor

Problema del fattore di scala:
Se faccio crescere dello stesso fattore sia tasso di arrivo sia tasso di servizio, cosa succede alle prestazioni?





Se lambda aumenta? Vorrei che i clienti non se ne rendessero conto!

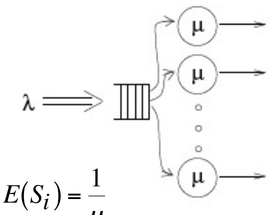
What about waiting and response time? a = fattore di scala

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

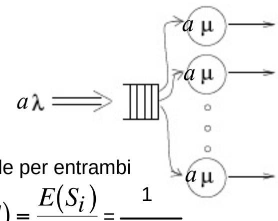
$$E(S_i) = \frac{1}{\mu} \qquad E(S_i) = \frac{1}{a\mu} \qquad E(S) = \frac{E(S_i)}{m}$$

27

Scaling factor



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



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

vale per entrambi

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

Mean waiting time

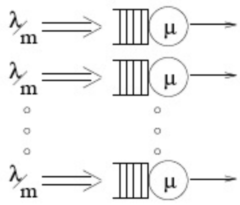
tempo servizio medio per liberarne uno qualsiasi

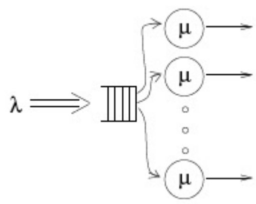
$$E(T_Q)_{m,a} = \frac{P_Q E(S)_{m,a}}{1-\rho} = \frac{P_Q}{ma\mu(1-\rho)} = \frac{1}{a}\frac{P_Q E(S)m,1}{(1-\rho)} = \frac{1}{a}E(T_Q)_{m,1}$$

Pq è in funzione di 'rho' ed 'm', non viene variata da lambda.

28

Quando faccio il tempo di servizio devo sommarvi E[Si], perchè è ciò che sperimenta un singolo job.
Qui invece, il job sta in coda per un tempo E[Tq] per poi andare nel primo servente libero, e il tempo in cui un servente si libera è dato da E[S].
Quindi, nel multiserver:
Uso E[Si] se devo sapere quanto tempo un job sta nel centro, il tempo che sperimenta.
Uso E[S] quando calcolo E[Tq] e voglio sapere il tempo che aspetta un job per entrare in un servente libero qualsiasi.





u = "mu"

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

$\rho = 0.666667$ (lambda/m)*1/u

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

qui il mio lambda è lambda/4 = 1

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

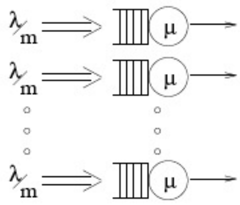
potevo anche fare E[Ts] - E[S]

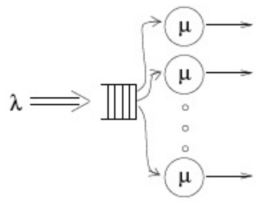
$$E(S) = \frac{E(S_i)}{4} = 0.1667$$

$\rho = 0.666667$ lambda*1/(m*u)

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

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





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

$\rho = 0.666667$

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

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

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

$E(S_i)$=0.666667 s

$\rho = 0.666667$

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

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

$$E(T_Q) = 0.189292$$

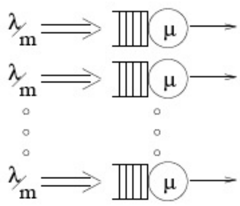
$\lambda = 4$ j/s, $m\mu = 4\text{x}1.5 = 6$ j/s E(S)=0.166667 s

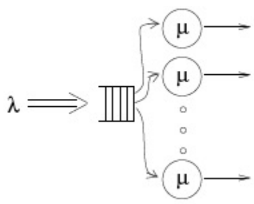
$\rho = 0.666667$

$$E(T_S) = \frac{1}{m\mu - \lambda} = 0.5$$ 1/4 di quello prima

$$E(T_Q) = 0.3334$$

31





$\rho = 0.666667$

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

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

cosa sarebbe successo
con variabilità non
esponenziale?
con un altro 'm'?

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

32

Le popolazioni del sistema si mantengono costanti, anche se sembra controintuitivo, ma alla fine con Little:
N = lamda*T = a*lambda*T/a, non cambia nulla, alla fine si semplifica e resta un qualcosa di costante.
Concettualmente è proprio il fatto che la popolazione rimanga costante che ci dà l'idea che il tempo sia minore.
Con Little abbiamo popolazione in media nel sistema, che è diverso dal traffico in entrata che è cresciuto da 'a'.