

Sistemi e Applicazioni Cloud - Esercizio

Simulatore reti sbilanciate [Tempo consegna: 2h 30m]

Parte 1: rete base

Si simuli una rete a code che implementa modelli M/M/1 come quella in figura.

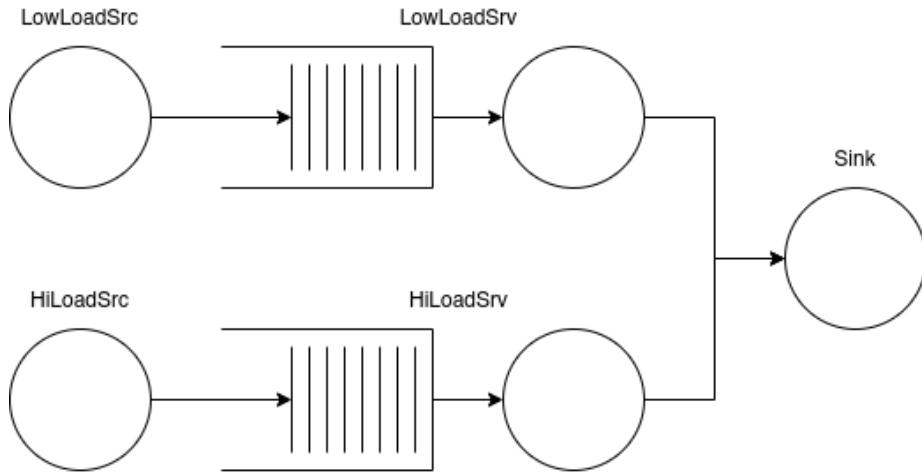


Figure 1: Modello di rete

La rete ha due generatori di carico:

- LowLoadSrc che invia richieste con un tasso $\lambda - \delta$
- HiLoadSrc che invia richieste con un tasso $\lambda + \delta$

Con $\lambda = 6$ richieste al secondo

Entrambi i server hanno un processing rate $\mu = 10$ richieste al secondo. Si chiede di mostrare il tempo di risposta T_r e l'utilizzazione dei server ρ per diversi valori di δ (espresso in richieste al secondo):

δ	Avg(T_r)	StdDev(T_r)	Max(T_r)	ρ (LowLoadSrv)	ρ (HiLoadSrv)
0	0.254	0.004634	2.145	0.600	0.590
1	0.2767	0.006955	3.0368	0.199	0.700
2	0.3967	0.009026	4.6430	0.399	0.8022
3	0.7954	0.03457	7.20628	0.299	0.90457

Parte 2: problema duale

Si consideri la stessa rete di prima. Questa volta però le sorgenti producono lo stesso volume di richieste λ , mentre la capacità di processing è diversi per i due

server, rispettivamente:

- LowLoadSrv che processa dati a un tasso $\mu + \delta$
 - HiLoadSrv che processa dati a un tasso $\mu - \delta$

δ	Avg(T_r)	StdDev(T_r)	Max(T_r)	$\rho(\text{LowLoadSrv})$	$\rho(\text{HiLoadSrv})$
0	0.254	0.00463	2.8245	0.6001	0.5999
1	0.26112	0.007265	3.2989	0.58163	0.66465
2	0.335	0.00584	6.2600	0.5202	0.74017
3	0.5872	0.0268	3.0736	0.46017	0.8582

Parte 3: configurazione avanzata

Valutare la possibilità di prendere i dati relativi alle due sorgenti dati. Per farlo:

- Studiare l'uso della proprietà `jobType` nel componente `Source`
 - Studiare il componente `Classifier` con particolare riferimento alla proprietà `dispatcherField`
 - Modificare la rete per aver due `Sink` e un `Classifier` che smista il traffico tra i due
 - Ri-valutare i tempi di risposta medi dei due server nei due problemi precedenti

es.ned	esame.init.mako	E Classifier.ned	esame.json	E fse1.tdata							
E fse1.tdata											
#signLifeTime_mean_low	signal(LifeTime_mean_low)	LifeTime_mean_high	signal(LifeTime_mean_high)	LifeTime_max_low	signal(LifeTime_max_low)	LifeTime_max_high	signal(LifeTime_max_high)	BusyTime_low	signal(BusyTime_low)	BusyTime_high	signal(BusyTime_high)
# 0.2504987998937434587	0.08388032324734587	0.2509716823537765	0.0802423150217895	0.2815079764455989	0.35697759149877895	0.6048655264629668	0.08475714930148483	0.59996135348438	0.082998085527367816	0.2504987998937434587	0.08388032324734587
1.0996851902183798	0.026726762038212012	1.091635444680535648	0.1016983505835648	2.1366830721870803	0.38098240988779462	4.1368592016354802	0.4017931375597	0.499368614595981	0.10262964134595981	0.082696380237987	0.0855179427289255
2.1674371187177408	0.0223091652818436	5.87589428424439	0.18031351494363558	4.1792650534688802	0.184018830592974	4.613895368878999	0.62852235787534885	0.399797867687685	0.0825235787534885	0.082517620315301	0.082517620315301
1.4148306994692356	0.1021633026518534	0.104093698508	0.84404374482187	1.409664613534	0.1292437444008106	7.28625031841919	0.7184464182184	0.29592917105161	0.0811506761613667	0.90517261448665	0.0838747023431472