Progetto di Simulazione di Sistemi

Samuele Evangelisti

a.a. 2019/2020

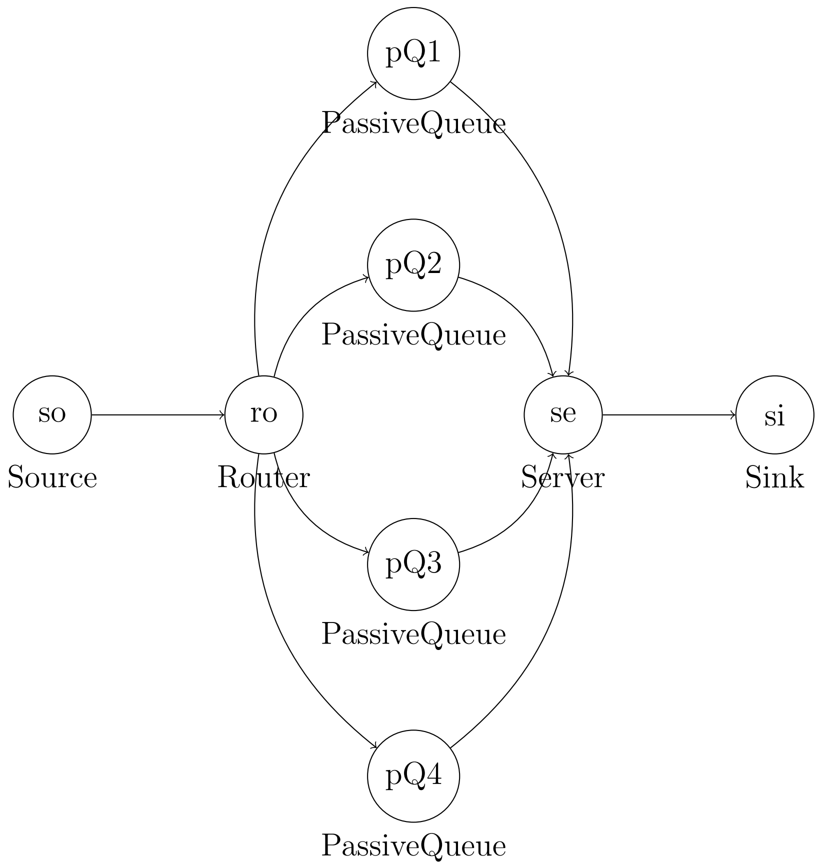
Contents

1 Introduzione

Il modello di simulazione studiato è una variante del modello riportato in [1] nella sezione 2. A tale scopo è stata utilizzata la piattaforma *OMNeT++ 5.6.1* modificando opportunamente alcuni moduli presenti nella libreria *queueinglib*. La nuova libreria, utilizzata nella fase di simulazione, è stata chiamata *pssqueueinglib*.
Di seguito vengono elencate le specifiche, i dettagli implementativi e i risultati sperimentali.

2 Modello

2.1 Rete



- La *Source* **so** genera *Job* seguendo una distribuzione di probabilità esponenziale di media $1/\lambda$ con $\lambda = 2.0, 1.4, 1.2, 1.0$
- Il *Router* **ro** inoltra i *Job* alle prime $K = 1, 2, 4$ *PassiveQueue* seguendo una distribuzione uniforme
- Le *PassiveQueue* **pQ1, ..., pQ4** hanno capacità illimitata
- Il *Server* **se** fornisce servizio alle code secondo la politica *exhaustive service* (il *Server* svuota completamente la *PassiveQueue* prima di spostarsi sulla successiva) facendo polling sulle code in maniera circolare da **pQ1** a **pQ4**, il tempo di servizio è caratterizzato da una distribuzione di probabilità esponenziale negativa di media $1/\mu$ con $\mu = 3.0, 4.0$
- A ogni *Job* viene assegnato un valore di deadline secondo una distrbuzione uniforme su $[a, b] = [4.0, 6.0], [3.0, 7.0]$, se il *Job* supera la sua deadline viene scartato dal *Server* altrimenti inizia il servizio

Il codice sorgente del file *network.ned* è riportato in appendice nella sezione A.1 a pagina 25.

2.2 Configurazione

I possibili valori dei parametri sono:

- λ: 4 valori (2.0, 1.4, 1.2, 1.0)
- K: 3 valori (1, 2, 3)
- μ: 2 valori (3.0, 4.0)
- $[a, b]$: 2 valori ([4.0, 6.0], [3.0, 7.0])

In totale per il sistema sono presenti 48 possibili configurazioni. In seguito verranno forniti i dati sperimentali per ogni possibile configurazione.
Il codice sorgente del file *omnetpp.ini* è riportato in appendice nella sezione A.2 a pagina 27.

3 Misure di Prestazione

Le misure di prestazioni sulla quale ci si concentra sono le seguenti:

1. Mediana della distribuzione del tempo di risposta del sistema (risultati nella sezione 4.4 a pagina 5)
2. Tempo minimo di permanenza dei *Job* nel sistema (risultati nella sezione 4.5.1 a pagina 9)
3. Tempo massimo di permanenza dei *Job* nel sistema (risultati nella sezione 4.5.2 a pagina 13)
4. Tempo medio di permanenza dei *Job* nel sistema (risultati nella sezione 4.5.3 a pagina 17)
5. Numero medio di *Job* non serviti (risultati nella sezione 4.6 a pagina 21)

Per ogni misura di prestazione vengono calcolati la stima puntuale e l'intervallo di confidenza al 90% e al 95%.
Per ottenere le misure di prestazione richieste sono state valutati i seguenti valori:

- *Network.sink.lifeTime:vector* (1, 2, 3, 4)
- *Network.server.droppedForDeadline:count* (5)

4 Risultati Sperimentali

4.1 Numero di Rilevazioni

Le rilevazioni di interesse vengono fornite dai *Job* che circolano nel sistema. Per ottenere un numero ragionevole di osservazioni è necessario che un numero ragionevole di *Job* vengano generati e terminati. Un valore ragionevole per la simulazione è di circa 1000 *Job*. Per ottenere questi valori per ogni run sono stati analizzati:

- *Network.source.created:last* (numero di *Job* creati)
- *Network.sink.lifeTime:vector* (numero di *Job* terminati)

Analizzando i risultati ottenuti si ottiene

	$R_g(J)$	$R_t(J)$	
$J \geq 1000$	828	828	86.25%
$900 \leq J < 1000$	132	132	13.75%
$J < 900$	0	0	0.00%
	960		

Table 1: Analisi del numero di rilevazioni

con:

- $R_g(J)$: numero di run che hanno generato J *Job*
- $R_t(J)$: numero di run che hanno terminato J *Job*

4.2 Transiente Iniziale

Come si nota dai grafici (riportati in appendice nella sezione B a pagina 59) il transiente iniziale termina nell'intervallo che va da 200s a 400s. Analizzando nuovamente il numero di rilevazioni escludendo il transiente iniziale si ottiene:

	$R_{t,s\geq200}(J)$		$R_{t,s\geq300}(J)$		$R_{t,s\geq400}(J)$	
$J \geq 1000$	492	51.250%	262	27.291%	240	25.000%
$900 \leq J < 1000$	216	22.500%	218	22.708%	0	0.000%
$800 \leq J < 900$	114	11.875%	216	22.500%	216	22.500%
$700 \leq J < 800$	138	14.375%	150	15.625%	187	19.479%
$600 \leq J < 700$	0	0.000%	114	11.875%	197	20.520%
$500 \leq J < 600$	0	0.000%	0	0.000%	120	12.500%
$J < 500$	0	0.000%	0	0.000%	0	0.000%
	960					

Table 2: Analisi del numero di rilevazioni escludendo il transiente iniziale

con:

- $R_{t,s\geq200}(J)$: numero di run che hanno terminato J *Job* prendendo la parte steady-state nell'intervallo [200s, 1000s]

- $R_{t,s\geq 300}(J)$: numero di run che hanno terminato J *Job* prendendo la parte steady-state nell'intervallo [300s, 1000s]
- $R_{t,s\geq 400}(J)$: numero di run che hanno terminato J *Job* prendendo la parte steady-state nell'intervallo [300s, 1000s]

Dai risultati si può notare che prendendo la parte steady-state nell'intervallo [300s, 1000s] per ogni run di simulazione sono garantiti almeno 600 osservazioni. Essendo un numero ragionevole di osservazioni nelle sezioni successive analizzeremo le rilevazioni sull'intervallo [300s, 1000s].

4.3 Analisi dei Valori

Nelle sezioni successive sono riportati i risultati dell'analisi delle misure di prestazione per ogni differente configurazione del sistema. Per ogni misura di prestazione, analizzando i 20 run della configurazione, vengono calcolati i seguenti valori:

Media di un run :

$$\mu_m = \frac{1}{N - \hat{n}_0} \sum_{n=\hat{n}_0+1}^{N} V_{mn}$$

con:

- V_{mn}: n-esimo valore del run m
- N: numero di rilevazioni
- $[0, \hat{n}_0]$: transiente iniziale
- $[\hat{n}_0 + 1, N]$: steady state

Media di una configurazione :

$$\hat{\mu} = \frac{1}{M} \sum_{m=1}^{M} \hat{mu}_m$$

con:

- M: numero di run

Varianza :

$$\sigma^2(\hat{\mu}_m) = \frac{1}{M-1} \sum_{m=1}^{M} (\hat{\mu}_m - \hat{\mu})^2$$

Deviazione Standard :

$$\sigma(\hat{\mu}_m) = \sqrt{\sigma^2(\hat{\mu}_m)}$$

Intervallo di Confidenza :

$$\hat{\mu} - t_{M-1}(1 - \frac{\alpha}{2}) \frac{\sigma(\hat{\mu}_m)}{M^{1/2}} \leq \hat{\mu} \leq \hat{\mu} + t_{M-1}(1 - \frac{\alpha}{2}) \frac{\sigma(\hat{\mu}_m)}{M^{1/2}}$$

Intervallo di confidenza al 90%:

$$IC_{90} = [\hat{\mu} - 1.64\sigma(\hat{\mu}_m)\sqrt{M}, \hat{\mu} + 1.64\sigma(\hat{\mu}_m)\sqrt{M}]$$

Intervallo di confidenza al 95%:

$$IC_{90} = [\hat{\mu} - 1.96\sigma(\hat{\mu}_m)\sqrt{M}, \hat{\mu} + 1.96Am\sigma(\hat{\mu}_m)\sqrt{M}]$$

4.4 Tempo di Risposta

	Configurazione 1
μ	0.674136134358
σ^2	0.004254766769
σ	0.065228573253
IC_{90}	[0.650215828443, 0.698056440273]
IC_{95}	[0.645548451680, 0.702723817036]

	Configurazione 2
μ	0.344984628733
σ^2	0.000409650329
σ	0.020239820373
IC_{90}	[0.337562380003, 0.352406877463]
IC_{95}	[0.336114136349, 0.353855121117]

	Configurazione 3
μ	0.613641051176
σ^2	0.002839978095
σ	0.053291444853
IC_{90}	[0.594098271073, 0.633183831279]
IC_{95}	[0.590285045687, 0.636997056665]

	Configurazione 4
μ	0.332640744781
σ^2	0.000339349283
σ	0.018421435434
IC_{90}	[0.325885325354, 0.339396164209]
IC_{95}	[0.324567194734, 0.340714294829]

	Configurazione 5
μ	0.602845390211
σ^2	0.002280896668
σ	0.047758733945
IC_{90}	[0.585331539009, 0.620359241412]
IC_{95}	[0.581914202189, 0.623776578232]

	Configurazione 6
μ	0.331571806101
σ^2	0.000347185934
σ	0.018632926074
IC_{90}	[0.324738829853, 0.338404782350]
IC_{95}	[0.323405566194, 0.339738046008]

	Configurazione 7
μ	0.670027016797
σ^2	0.004577878760
σ	0.067660023355
IC_{90}	[0.645215061298, 0.694838972296]
IC_{95}	[0.640373704128, 0.699680329467]

	Configurazione 8
μ	0.344902605458
σ^2	0.000410074648
σ	0.020250299940
IC_{90}	[0.337476513712, 0.352328697203]
IC_{95}	[0.336027520200, 0.353777690715]

	Configurazione 9
μ	0.608686525370
σ^2	0.003275404992
σ	0.057231154030
IC_{90}	[0.587698994233, 0.629674056508]
IC_{95}	[0.583603866206, 0.633769184535]

	Configurazione 10
μ	0.332701552505
σ^2	0.000341006017
σ	0.018466348242
IC_{90}	[0.325929662870, 0.339473442139]
IC_{95}	[0.324608318551, 0.340794786458]

	Configurazione 11
μ	0.599814359171
σ^2	0.002963692608
σ	0.054439807203
IC_{90}	[0.579850457198, 0.619778261143]
IC_{95}	[0.575955061691, 0.623673656650]

	Configurazione 12
μ	0.330859379279
σ^2	0.000381427379
σ	0.019530165869
IC_{90}	[0.323697371606, 0.338021386952]
IC_{95}	[0.322299906694, 0.339418851864]

	Configurazione 13
μ	0.426744110910
σ^2	0.000971466795
σ	0.031168362082
IC_{90}	[0.415314200386, 0.438174021433]
IC_{95}	[0.413083973943, 0.440404247877]

	Configurazione 14
μ	0.267446338550
σ^2	0.000343013866
σ	0.018520633518
IC_{90}	[0.260654541683, 0.274238135417]
IC_{95}	[0.259329313026, 0.275563364074]

	Configurazione 15
μ	0.416230281984
σ^2	0.000784422782
σ	0.028007548667
IC_{90}	[0.405959489621, 0.426501074347]
IC_{95}	[0.403955432575, 0.428505131394]

	Configurazione 16
μ	0.263658232812
σ^2	0.000331626611
σ	0.018210618077
IC_{90}	[0.256980123303, 0.270336342321]
IC_{95}	[0.255677077546, 0.271639388079]

	Configurazione 17
μ	0.411535515666
σ^2	0.000777128809
σ	0.027877030135
IC_{90}	[0.401312586426, 0.421758444906]
IC_{95}	[0.399317868525, 0.423753162807]

	Configurazione 18
μ	0.262291383599
σ^2	0.000308682331
σ	0.017569357727
IC_{90}	[0.255848433974, 0.268734333224]
IC_{95}	[0.254591273072, 0.269991494126]

	Configurazione 19
μ	0.426613132062
σ^2	0.000966421134
σ	0.031087314686
IC_{90}	[0.415212942846, 0.438013321277]
IC_{95}	[0.412988515682, 0.440237748441]

	Configurazione 20
μ	0.267461187300
σ^2	0.000343871635
σ	0.018543776170
IC_{90}	[0.260660903671, 0.274261470928]
IC_{95}	[0.259334019060, 0.275588355539]

	Configurazione 21
μ	0.415481808205
σ^2	0.000782865575
σ	0.027979735084
IC_{90}	[0.405221215504, 0.425742400906]
IC_{95}	[0.403219148635, 0.427744467774]

	Configurazione 22
μ	0.263550588376
σ^2	0.000328361703
σ	0.018120753370
IC_{90}	[0.256905433617, 0.270195743136]
IC_{95}	[0.255608818054, 0.271492358698]

	Configurazione 23
μ	0.410917241919
σ^2	0.000807611459
σ	0.028418505566
IC_{90}	[0.400495745435, 0.421338738402]
IC_{95}	[0.398462282707, 0.423372201131]

	Configurazione 24
μ	0.262272012570
σ^2	0.000309043408
σ	0.017579630496
IC_{90}	[0.255825295766, 0.268718729375]
IC_{95}	[0.254567399804, 0.269976625336]

	Configurazione 25
μ	0.392315120257
σ^2	0.000779110512
σ	0.027912551159
IC_{90}	[0.382079164919, 0.402551075595]
IC_{95}	[0.380081905341, 0.404548335173]

	Configurazione 26
μ	0.251816025955
σ^2	0.000261738733
σ	0.016178341480
IC_{90}	[0.245883183060, 0.257748868850]
IC_{95}	[0.244725555178, 0.258906496732]

	Configurazione 27
μ	0.383388196366
σ^2	0.000667277248
σ	0.025831710120
IC_{90}	[0.373915316959, 0.392861075774]
IC_{95}	[0.372066950245, 0.394709442488]

	Configurazione 28
μ	0.248896192422
σ^2	0.000209593624
σ	0.014477348658
IC_{90}	[0.243587129362, 0.254205255482]
IC_{95}	[0.242551214618, 0.255241170226]

	Configurazione 29
μ	0.381603655185
σ^2	0.000634573773
σ	0.025190747773
IC_{90}	[0.372365826379, 0.390841483991]
IC_{95}	[0.370563323198, 0.392643987173]

	Configurazione 30
μ	0.249073657821
σ^2	0.000226295658
σ	0.015043126606
IC_{90}	[0.243557115416, 0.254590200225]
IC_{95}	[0.242480716899, 0.255666598743]

	Configurazione 31
μ	0.392240227264
σ^2	0.000764026584
σ	0.027641030814
IC_{90}	[0.382103842549, 0.402376611978]
IC_{95}	[0.380126011385, 0.404354443142]

	Configurazione 32
μ	0.251816025955
σ^2	0.000261738733
σ	0.016178341480
IC_{90}	[0.245883183060, 0.257748868850]
IC_{95}	[0.244725555178, 0.258906496732]

	Configurazione 33
μ	0.382949844069
σ^2	0.000645740015
σ	0.025411415061
IC_{90}	[0.373631093226, 0.392268594911]
IC_{95}	[0.371812800378, 0.394086887759]

	Configurazione 34
μ	0.248896192422
σ^2	0.000209593624
σ	0.014477348658
IC_{90}	[0.243587129362, 0.254205255482]
IC_{95}	[0.242551214618, 0.255241170226]

	Configurazione 35
μ	0.380169440899
σ^2	0.000600928143
σ	0.024513835739
IC_{90}	[0.371179845991, 0.389159035808]
IC_{95}	[0.369425778691, 0.390913103107]

	Configurazione 36
μ	0.249073657821
σ^2	0.000226295658
σ	0.015043126606
IC_{90}	[0.243557115416, 0.254590200225]
IC_{95}	[0.242480716899, 0.255666598743]

	Configurazione 37
μ	0.349793690777
σ^2	0.000446784324
σ	0.021137273345
IC_{90}	[0.342042332448, 0.357545049107]
IC_{95}	[0.340529872286, 0.359057509269]

	Configurazione 38
μ	0.235418879570
σ^2	0.000288896833
σ	0.016996965405
IC_{90}	[0.229185834881, 0.241651924259]
IC_{95}	[0.227969631039, 0.242868128101]

	Configurazione 39
μ	0.344362301644
σ^2	0.000363384239
σ	0.019062639885
IC_{90}	[0.337371742831, 0.351352860457]
IC_{95}	[0.336007731356, 0.352716871932]

	Configurazione 40
μ	0.234257752799
σ^2	0.000266421043
σ	0.016322409220
IC_{90}	[0.228272078081, 0.240243427517]
IC_{95}	[0.227104141551, 0.241411364047]

	Configurazione 41
μ	0.343700761468
σ^2	0.000358059135
σ	0.018922450550
IC_{90}	[0.336761612208, 0.350639910728]
IC_{95}	[0.335407631865, 0.351993891072]

	Configurazione 42
μ	0.233888792021
σ^2	0.000266009424
σ	0.016309795326
IC_{90}	[0.227907743009, 0.239869841033]
IC_{95}	[0.226740709056, 0.241036874986]

	Configurazione 43
μ	0.349787308000
σ^2	0.000446669074
σ	0.021134546927
IC_{90}	[0.342036949490, 0.357537666511]
IC_{95}	[0.340524684414, 0.359049931586]

	Configurazione 44
μ	0.235418879570
σ^2	0.000288896833
σ	0.016996965405
IC_{90}	[0.229185834881, 0.241651924259]
IC_{95}	[0.227969631039, 0.242868128101]

	Configurazione 45
μ	0.344337302602
σ^2	0.000363315150
σ	0.019060827622
IC_{90}	[0.337347408374, 0.351327196830]
IC_{95}	[0.335983526573, 0.352691078631]

	Configurazione 46
μ	0.234257752799
σ^2	0.000266421043
σ	0.016322409220
IC_{90}	[0.228272078081, 0.240243427517]
IC_{95}	[0.227104141551, 0.241411364047]

	Configurazione 47
μ	0.343068879199
σ^2	0.000338753030
σ	0.018405244642
IC_{90}	[0.336319397180, 0.349818361218]
IC_{95}	[0.335002425079, 0.351135333319]

	Configurazione 48
μ	0.233888792021
σ^2	0.000266009424
σ	0.016309795326
IC_{90}	[0.227907743009, 0.239869841033]
IC_{95}	[0.226740709056, 0.241036874986]

4.5 Tempo di Permanenza

4.5.1 Minimo

	Configurazione 1
μ	0.000805906653
σ^2	0.000000220213
σ	0.000469268561
IC_{90}	[0.000633818763, 0.000977994543]
IC_{95}	[0.000600240638, 0.001011572668]

	Configurazione 2
μ	0.000475572328
σ^2	0.000000131940
σ	0.000363235937
IC_{90}	[0.000342368207, 0.000608776448]
IC_{95}	[0.000316377159, 0.000634767496]

	Configurazione 3
μ	0.000785977175
σ^2	0.000000221788
σ	0.000470943767
IC_{90}	[0.000613274962, 0.000958679388]
IC_{95}	[0.000579576969, 0.000992377381]

	Configurazione 4
μ	0.000475572328
σ^2	0.000000131940
σ	0.000363235937
IC_{90}	[0.000342368207, 0.000608776448]
IC_{95}	[0.000316377159, 0.000634767496]

	Configurazione 5
μ	0.000785977175
σ^2	0.000000221788
σ	0.000470943767
IC_{90}	[0.000613274962, 0.000958679388]
IC_{95}	[0.000579576969, 0.000992377381]

	Configurazione 6
μ	0.000475572328
σ^2	0.000000131940
σ	0.000363235937
IC_{90}	[0.000342368207, 0.000608776448]
IC_{95}	[0.000316377159, 0.000634767496]

	Configurazione 7
μ	0.000785977175
σ^2	0.000000221788
σ	0.000470943767
IC_{90}	[0.000613274962, 0.000958679388]
IC_{95}	[0.000579576969, 0.000992377381]

	Configurazione 8
μ	0.000475572328
σ^2	0.000000131940
σ	0.000363235937
IC_{90}	[0.000342368207, 0.000608776448]
IC_{95}	[0.000316377159, 0.000634767496]

	Configurazione 9
μ	0.000785977175
σ^2	0.000000221788
σ	0.000470943767
IC_{90}	[0.000613274962, 0.000958679388]
IC_{95}	[0.000579576969, 0.000992377381]

	Configurazione 10
μ	0.000475572328
σ^2	0.000000131940
σ	0.000363235937
IC_{90}	[0.000342368207, 0.000608776448]
IC_{95}	[0.000316377159, 0.000634767496]

	Configurazione 11
μ	0.000778310754
σ^2	0.000000230536
σ	0.000480141815
IC_{90}	[0.000602235477, 0.000954386031]
IC_{95}	[0.000567879325, 0.000988742182]

	Configurazione 12
μ	0.000475572328
σ^2	0.000000131940
σ	0.000363235937
IC_{90}	[0.000342368207, 0.000608776448]
IC_{95}	[0.000316377159, 0.000634767496]

	Configurazione 13
μ	0.000871409584
σ^2	0.000002249759
σ	0.001499919685
IC_{90}	[0.000321366315, 0.001421452854]
IC_{95}	[0.000214040799, 0.001528778370]

	Configurazione 14
μ	0.000367294460
σ^2	0.000000072162
σ	0.000268629035
IC_{90}	[0.000268784123, 0.000465804796]
IC_{95}	[0.000249562594, 0.000485026325]

	Configurazione 15
μ	0.000871409584
σ^2	0.000002249759
σ	0.001499919685
IC_{90}	[0.000321366315, 0.001421452854]
IC_{95}	[0.000214040799, 0.001528778370]

	Configurazione 16
μ	0.000367294460
σ^2	0.000000072162
σ	0.000268629035
IC_{90}	[0.000268784123, 0.000465804796]
IC_{95}	[0.000249562594, 0.000485026325]

	Configurazione 17
μ	0.000871409584
σ^2	0.000002249759
σ	0.001499919685
IC_{90}	[0.000321366315, 0.001421452854]
IC_{95}	[0.000214040799, 0.001528778370]

	Configurazione 18
μ	0.000367294460
σ^2	0.000000072162
σ	0.000268629035
IC_{90}	[0.000268784123, 0.000465804796]
IC_{95}	[0.000249562594, 0.000485026325]

	Configurazione 19
μ	0.000871409584
σ^2	0.000002249759
σ	0.001499919685
IC_{90}	[0.000321366315, 0.001421452854]
IC_{95}	[0.000214040799, 0.001528778370]

	Configurazione 20
μ	0.000367294460
σ^2	0.000000072162
σ	0.000268629035
IC_{90}	[0.000268784123, 0.000465804796]
IC_{95}	[0.000249562594, 0.000485026325]

	Configurazione 21
μ	0.000871409584
σ^2	0.000002249759
σ	0.001499919685
IC_{90}	[0.000321366315, 0.001421452854]
IC_{95}	[0.000214040799, 0.001528778370]

	Configurazione 22
μ	0.000367294460
σ^2	0.000000072162
σ	0.000268629035
IC_{90}	[0.000268784123, 0.000465804796]
IC_{95}	[0.000249562594, 0.000485026325]

	Configurazione 23
μ	0.000871409584
σ^2	0.000002249759
σ	0.001499919685
IC_{90}	[0.000321366315, 0.001421452854]
IC_{95}	[0.000214040799, 0.001528778370]

	Configurazione 24
μ	0.000367294460
σ^2	0.000000072162
σ	0.000268629035
IC_{90}	[0.000268784123, 0.000465804796]
IC_{95}	[0.000249562594, 0.000485026325]

	Configurazione 25
μ	0.000843380181
σ^2	0.000001830705
σ	0.001353035425
IC_{90}	[0.000347201594, 0.001339558767]
IC_{95}	[0.000250386260, 0.001436374101]

	Configurazione 26
μ	0.000366367407
σ^2	0.000000068487
σ	0.000261699398
IC_{90}	[0.000270398274, 0.000462336541]
IC_{95}	[0.000251672589, 0.000481062226]

	Configurazione 27
μ	0.000843380181
σ^2	0.000001830705
σ	0.001353035425
IC_{90}	[0.000347201594, 0.001339558767]
IC_{95}	[0.000250386260, 0.001436374101]

	Configurazione 28
μ	0.000366367407
σ^2	0.000000068487
σ	0.000261699398
IC_{90}	[0.000270398274, 0.000462336541]
IC_{95}	[0.000251672589, 0.000481062226]

	Configurazione 29
μ	0.000843380181
σ^2	0.000001830705
σ	0.001353035425
IC_{90}	[0.000347201594, 0.001339558767]
IC_{95}	[0.000250386260, 0.001436374101]

	Configurazione 30
μ	0.000366367407
σ^2	0.000000068487
σ	0.000261699398
IC_{90}	[0.000270398274, 0.000462336541]
IC_{95}	[0.000251672589, 0.000481062226]

	Configurazione 31
μ	0.000843380181
σ^2	0.000001830705
σ	0.001353035425
IC_{90}	[0.000347201594, 0.001339558767]
IC_{95}	[0.000250386260, 0.001436374101]

	Configurazione 32
μ	0.000366367407
σ^2	0.000000068487
σ	0.000261699398
IC_{90}	[0.000270398274, 0.000462336541]
IC_{95}	[0.000251672589, 0.000481062226]

	Configurazione 33
μ	0.000843380181
σ^2	0.000001830705
σ	0.001353035425
IC_{90}	[0.000347201594, 0.001339558767]
IC_{95}	[0.000250386260, 0.001436374101]

	Configurazione 34
μ	0.000366367407
σ^2	0.000000068487
σ	0.000261699398
IC_{90}	[0.000270398274, 0.000462336541]
IC_{95}	[0.000251672589, 0.000481062226]

	Configurazione 35
μ	0.000843380181
σ^2	0.000001830705
σ	0.001353035425
IC_{90}	[0.000347201594, 0.001339558767]
IC_{95}	[0.000250386260, 0.001436374101]

	Configurazione 36
μ	0.000366367407
σ^2	0.000000068487
σ	0.000261699398
IC_{90}	[0.000270398274, 0.000462336541]
IC_{95}	[0.000251672589, 0.000481062226]

	Configurazione 37
μ	0.000585655633
σ^2	0.000000221249
σ	0.000470370908
IC_{90}	[0.000413163496, 0.000758147771]
IC_{95}	[0.000379506494, 0.000791804773]

	Configurazione 38
μ	0.000427059422
σ^2	0.000000096839
σ	0.000311189582
IC_{90}	[0.000312941488, 0.000541177356]
IC_{95}	[0.000290674574, 0.000563444270]

	Configurazione 39
μ	0.000585655633
σ^2	0.000000221249
σ	0.000470370908
IC_{90}	[0.000413163496, 0.000758147771]
IC_{95}	[0.000379506494, 0.000791804773]

	Configurazione 40
μ	0.000427059422
σ^2	0.000000096839
σ	0.000311189582
IC_{90}	[0.000312941488, 0.000541177356]
IC_{95}	[0.000290674574, 0.000563444270]

	Configurazione 41
μ	0.000585655633
σ^2	0.000000221249
σ	0.000470370908
IC_{90}	[0.000413163496, 0.000758147771]
IC_{95}	[0.000379506494, 0.000791804773]

	Configurazione 42
μ	0.000427059422
σ^2	0.000000096839
σ	0.000311189582
IC_{90}	[0.000312941488, 0.000541177356]
IC_{95}	[0.000290674574, 0.000563444270]

	Configurazione 43
μ	0.000585655633
σ^2	0.000000221249
σ	0.000470370908
IC_{90}	[0.000413163496, 0.000758147771]
IC_{95}	[0.000379506494, 0.000791804773]

	Configurazione 44
μ	0.000427059422
σ^2	0.000000096839
σ	0.000311189582
IC_{90}	[0.000312941488, 0.000541177356]
IC_{95}	[0.000290674574, 0.000563444270]

	Configurazione 45
μ	0.000585655633
σ^2	0.000000221249
σ	0.000470370908
IC_{90}	[0.000413163496, 0.000758147771]
IC_{95}	[0.000379506494, 0.000791804773]

	Configurazione 46
μ	0.000427059422
σ^2	0.000000096839
σ	0.000311189582
IC_{90}	[0.000312941488, 0.000541177356]
IC_{95}	[0.000290674574, 0.000563444270]

	Configurazione 47
μ	0.000585655633
σ^2	0.000000221249
σ	0.000470370908
IC_{90}	[0.000413163496, 0.000758147771]
IC_{95}	[0.000379506494, 0.000791804773]

	Configurazione 48
μ	0.000427059422
σ^2	0.000000096839
σ	0.000311189582
IC_{90}	[0.000312941488, 0.000541177356]
IC_{95}	[0.000290674574, 0.000563444270]

4.5.2 Massimo

	Configurazione 1
μ	5.190485467576
σ^2	0.581853479207
σ	0.762793208679
IC_{90}	[4.910757642926, 5.470213292227]
IC_{95}	[4.856176603969, 5.524794331183]

	Configurazione 2
μ	2.722497523093
σ^2	2.322912801989
σ	1.524110495335
IC_{90}	[2.163583116756, 3.281411929430]
IC_{95}	[2.054526647226, 3.390468398960]

	Configurazione 3
μ	5.385114178223
σ^2	0.513698157519
σ	0.716727394146
IC_{90}	[5.122279385581, 5.647948970865]
IC_{95}	[5.070994547992, 5.699233808454]

	Configurazione 4
μ	3.608593426970
σ^2	2.276525667147
σ	1.508815981870
IC_{90}	[3.055287750406, 4.161899103534]
IC_{95}	[2.947325667174, 4.269861186766]

	Configurazione 5
μ	5.604101358375
σ^2	0.632446959280
σ	0.795265338915
IC_{90}	[5.312465511669, 5.895737205081]
IC_{95}	[5.255560956214, 5.952641760536]

	Configurazione 6
μ	3.785319090357
σ^2	2.166292039257
σ	1.471832884283
IC_{90}	[3.245575675909, 4.325062504804]
IC_{95}	[3.140259887724, 4.430378292989]

	Configurazione 7
μ	5.089979697687
σ^2	0.869770660289
σ	0.932614958216
IC_{90}	[4.747975664969, 5.431983730406]
IC_{95}	[4.681243170780, 5.498716224595]

	Configurazione 8
μ	2.702571294083
σ^2	2.259029684041
σ	1.503006880903
IC_{90}	[2.151395902842, 3.253746685324]
IC_{95}	[2.043849485039, 3.361293103128]

	Configurazione 9
μ	5.344287568689
σ^2	1.296009535512
σ	1.138424145699
IC_{90}	[4.926810189259, 5.761764948119]
IC_{95}	[4.845351188395, 5.843223948983]

	Configurazione 10
μ	3.552338806103
σ^2	2.220821998589
σ	1.490242261711
IC_{90}	[3.005844394082, 4.098833218124]
IC_{95}	[2.899211338078, 4.205466274128]

	Configurazione 11
μ	5.487085340201
σ^2	1.331218379261
σ	1.153784372949
IC_{90}	[5.063975132758, 5.910195547645]
IC_{95}	[4.981417043500, 5.992753636902]

	Configurazione 12
μ	3.417524344051
σ^2	2.789780198791
σ	1.670263511782
IC_{90}	[2.805013412611, 4.030035275491]
IC_{95}	[2.685499084525, 4.149549603576]

	Configurazione 13
μ	3.636529832853
σ^2	1.185140412723
σ	1.088641544643
IC_{90}	[3.237308487353, 4.035751178353]
IC_{95}	[3.159411639450, 4.113648026256]

	Configurazione 14
μ	2.439608942309
σ^2	0.511758150733
σ	0.715372735525
IC_{90}	[2.177270923504, 2.701946961114]
IC_{95}	[2.126083017395, 2.753134867222]

	Configurazione 15
μ	4.548943916544
σ^2	0.519316337280
σ	0.720636064376
IC_{90}	[4.284675755319, 4.813212077769]
IC_{95}	[4.233111236055, 4.864776597033]

	Configurazione 16
μ	2.908266038591
σ^2	1.013122205095
σ	1.006539718588
IC_{90}	[2.539152676409, 3.277379400773]
IC_{95}	[2.467130556959, 3.349401520223]

	Configurazione 17
μ	4.382060630680
σ^2	0.464673708333
σ	0.681669794206
IC_{90}	[4.132081991000, 4.632039270361]
IC_{95}	[4.083305671062, 4.680815590299]

	Configurazione 18
μ	2.834002189681
σ^2	0.842014931360
σ	0.917613715765
IC_{90}	[2.497499339813, 3.170505039549]
IC_{95}	[2.431840247156, 3.236164132206]

	Configurazione 19
μ	3.629023075570
σ^2	1.172455444139
σ	1.082799817205
IC_{90}	[3.231943980013, 4.026102171126]
IC_{95}	[3.154465132100, 4.103581019040]

	Configurazione 20
μ	2.439608942309
σ^2	0.511758150733
σ	0.715372735525
IC_{90}	[2.177270923504, 2.701946961114]
IC_{95}	[2.126083017395, 2.753134867222]

	Configurazione 21
μ	4.270961443640
σ^2	0.517939674914
σ	0.719680258805
IC_{90}	[4.007043790796, 4.534879096483]
IC_{95}	[3.955547663412, 4.586375223867]

	Configurazione 22
μ	2.775790751680
σ^2	0.850000354274
σ	0.921954637861
IC_{90}	[2.437696019922, 3.113885483438]
IC_{95}	[2.371726316164, 3.179855187196]

	Configurazione 23
μ	4.279939863816
σ^2	0.525361206987
σ	0.724818050953
IC_{90}	[4.014138104763, 4.545741622869]
IC_{95}	[3.962274346899, 4.597605380733]

	Configurazione 24
μ	2.773124533760
σ^2	0.722452662282
σ	0.849972153828
IC_{90}	[2.461426869310, 3.084822198210]
IC_{95}	[2.400607812832, 3.145641254689]

	Configurazione 25
μ	3.477467348692
σ^2	1.148842708301
σ	1.071840803618
IC_{90}	[3.084407089429, 3.870527607956]
IC_{95}	[3.007712404694, 3.947222292690]

	Configurazione 26
μ	2.035876721379
σ^2	0.615292123040
σ	0.784405585804
IC_{90}	[1.748223310646, 2.323530132112]
IC_{95}	[1.692095815868, 2.379657626889]

	Configurazione 27
μ	3.752652279005
σ^2	1.065237039852
σ	1.032103211822
IC_{90}	[3.374164396610, 4.131140161399]
IC_{95}	[3.300313102485, 4.204991455525]

	Configurazione 28
μ	2.452539688700
σ^2	0.598593990200
σ	0.773688561502
IC_{90}	[2.168816373123, 2.736263004277]
IC_{95}	[2.113455726181, 2.791623651219]

	Configurazione 29
μ	4.219112170998
σ^2	0.645528116194
σ	0.803447643717
IC_{90}	[3.924475749173, 4.513748592823]
IC_{95}	[3.866985715646, 4.571238626350]

	Configurazione 30
μ	2.463374616019
σ^2	0.638816750074
σ	0.799260126663
IC_{90}	[2.170273820132, 2.756475411907]
IC_{95}	[2.113083420934, 2.813665811105]

	Configurazione 31
μ	3.460001805048
σ^2	1.102761529348
σ	1.050124530400
IC_{90}	[3.074905232139, 3.845098377958]
IC_{95}	[2.999764437425, 3.920239172672]

	Configurazione 32
μ	2.035876721379
σ^2	0.615292123040
σ	0.784405585804
IC_{90}	[1.748223310646, 2.323530132112]
IC_{95}	[1.692095815868, 2.379657626889]

	Configurazione 33
μ	3.688717766618
σ^2	1.030675673711
σ	1.015221982480
IC_{90}	[3.316420486745, 4.061015046490]
IC_{95}	[3.243777115062, 4.133658418173]

	Configurazione 34
μ	2.452539688700
σ^2	0.598593990200
σ	0.773688561502
IC_{90}	[2.168816373123, 2.736263004277]
IC_{95}	[2.113455726181, 2.791623651219]

	Configurazione 35
μ	4.190465588786
σ^2	0.897416053152
σ	0.947320459587
IC_{90}	[3.843068825952, 4.537862351621]
IC_{95}	[3.775284091740, 4.605647085832]

	Configurazione 36
μ	2.463374616019
σ^2	0.638816750074
σ	0.799260126663
IC_{90}	[2.170273820132, 2.756475411907]
IC_{95}	[2.113083420934, 2.813665811105]

	Configurazione 37
μ	2.962626177928
σ^2	0.911789686685
σ	0.954876791364
IC_{90}	[2.612458393766, 3.312793962091]
IC_{95}	[2.544132972466, 3.381119383391]

	Configurazione 38
μ	1.958763628028
σ^2	0.575252976504
σ	0.758454333829
IC_{90}	[1.680626934512, 2.236900321544]
IC_{95}	[1.626356360167, 2.291170895889]

	Configurazione 39
μ	3.615090832862
σ^2	0.724765406502
σ	0.851331549105
IC_{90}	[3.302894657571, 3.927287008153]
IC_{95}	[3.241978330685, 3.988203335039]

	Configurazione 40
μ	2.256751932015
σ^2	0.879967688486
σ	0.938065929712
IC_{90}	[1.912748945476, 2.600754918554]
IC_{95}	[1.845626411518, 2.667877452513]

	Configurazione 41
μ	3.605885502103
σ^2	0.742650369116
σ	0.861771645574
IC_{90}	[3.289860785287, 3.921910218919]
IC_{95}	[3.228197425908, 3.983573578298]

	Configurazione 42
μ	2.247703051783
σ^2	0.831963653285
σ	0.912120416000
IC_{90}	[1.913214678153, 2.582191425413]
IC_{95}	[1.847948654030, 2.647457449536]

	Configurazione 43
μ	2.770017066973
σ^2	1.293031728346
σ	1.137115529903
IC_{90}	[2.353019576779, 3.187014557166]
IC_{95}	[2.271654212839, 3.268379921107]

	Configurazione 44
μ	1.958763628028
σ^2	0.575252976504
σ	0.758454333829
IC_{90}	[1.680626934512, 2.236900321544]
IC_{95}	[1.626356360167, 2.291170895889]

	Configurazione 45
μ	3.590975641600
σ^2	0.724847325144
σ	0.851379659813
IC_{90}	[3.278761823383, 3.903189459816]
IC_{95}	[3.217842053975, 3.964109229224]

	Configurazione 46
μ	2.256751932015
σ^2	0.879967688486
σ	0.938065929712
IC_{90}	[1.912748945476, 2.600754918554]
IC_{95}	[1.845626411518, 2.667877452513]

	Configurazione 47
μ	3.515642695942
σ^2	0.655552033332
σ	0.809661678315
IC_{90}	[3.218727493498, 3.812557898387]
IC_{95}	[3.160792819850, 3.870492572034]

	Configurazione 48
μ	2.247703051783
σ^2	0.831963653285
σ	0.912120416000
IC_{90}	[1.913214678153, 2.582191425413]
IC_{95}	[1.847948654030, 2.647457449536]

4.5.3 Medio

	Configurazione 1
μ	0.957053139848
σ^2	0.011176154538
σ	0.105717333195
IC_{90}	[0.918284992327, 0.995821287370]
IC_{95}	[0.910720475738, 1.003385803959]

	Configurazione 2
μ	0.493015602972
σ^2	0.001162086069
σ	0.034089383516
IC_{90}	[0.480514509640, 0.505516696304]
IC_{95}	[0.478075271916, 0.507955934027]

	Configurazione 3
μ	0.895943320000
σ^2	0.008186049831
σ	0.090476791669
IC_{90}	[0.862764109925, 0.929122530076]
IC_{95}	[0.856290117715, 0.935596522285]

	Configurazione 4
μ	0.490183310019
σ^2	0.000999386494
σ	0.031613074735
IC_{90}	[0.478590316629, 0.501776303410]
IC_{95}	[0.476328269139, 0.504038350900]

	Configurazione 5
μ	0.896130330416
σ^2	0.007998957014
σ	0.089436888442
IC_{90}	[0.863332468607, 0.928928192225]
IC_{95}	[0.856932885815, 0.935327775018]

	Configurazione 6
μ	0.491042111389
σ^2	0.001119596847
σ	0.033460377268
IC_{90}	[0.478771684176, 0.503312538601]
IC_{95}	[0.476377454476, 0.505706768301]

	Configurazione 7
μ	0.949860683735
σ^2	0.014122486751
σ	0.118838069452
IC_{90}	[0.906280963471, 0.993440403999]
IC_{95}	[0.897777603420, 1.001943764050]

	Configurazione 8
μ	0.491974437781
σ^2	0.001128865032
σ	0.033598586757
IC_{90}	[0.479653327056, 0.504295548507]
IC_{95}	[0.477249207890, 0.506699667673]

	Configurazione 9
μ	0.885100840864
σ^2	0.010942720473
σ	0.104607458974
IC_{90}	[0.846739701032, 0.923461980696]
IC_{95}	[0.839254600577, 0.930947081151]

	Configurazione 10
μ	0.489393315837
σ^2	0.001040804890
σ	0.032261507869
IC_{90}	[0.477562532194, 0.501224099480]
IC_{95}	[0.475254086606, 0.503532545069]

	Configurazione 11
μ	0.885500717163
σ^2	0.010715385160
σ	0.103515144592
IC_{90}	[0.847540145562, 0.923461288764]
IC_{95}	[0.840133204761, 0.930868229565]

	Configurazione 12
μ	0.488970151777
σ^2	0.001182057287
σ	0.034381059999
IC_{90}	[0.476362096260, 0.501578207294]
IC_{95}	[0.473901987867, 0.504038315687]

	Configurazione 13
μ	0.613574183746
σ^2	0.001911376224
σ	0.043719288929
IC_{90}	[0.597541658223, 0.629606709270]
IC_{95}	[0.594413360559, 0.632735006933]

	Configurazione 14
μ	0.384334564898
σ^2	0.000683019623
σ	0.026134644116
IC_{90}	[0.374750595005, 0.393918534791]
IC_{95}	[0.372880552099, 0.395788577697]

	Configurazione 15
μ	0.609237465824
σ^2	0.001710306168
σ	0.041355848057
IC_{90}	[0.594071649871, 0.624403281778]
IC_{95}	[0.591112466270, 0.627362465378]

	Configurazione 16
μ	0.384308666292
σ^2	0.000681223019
σ	0.026100249402
IC_{90}	[0.374737309461, 0.393880023122]
IC_{95}	[0.372869727641, 0.395747604943]

	Configurazione 17
μ	0.606366987881
σ^2	0.001835053655
σ	0.042837526243
IC_{90}	[0.590657818091, 0.622076157670]
IC_{95}	[0.587592614230, 0.625141361532]

	Configurazione 18
μ	0.384308666292
σ^2	0.000681223019
σ	0.026100249402
IC_{90}	[0.374737309461, 0.393880023122]
IC_{95}	[0.372869727641, 0.395747604943]

	Configurazione 19
μ	0.613121345426
σ^2	0.001897445174
σ	0.043559673713
IC_{90}	[0.597147353220, 0.629095337632]
IC_{95}	[0.594030476692, 0.632212214160]

	Configurazione 20
μ	0.384257716179
σ^2	0.000680382716
σ	0.026084146834
IC_{90}	[0.374692264404, 0.393823167954]
IC_{95}	[0.372825834789, 0.395689597568]

	Configurazione 21
μ	0.606346747407
σ^2	0.001683804757
σ	0.041034190100
IC_{90}	[0.591298888299, 0.621394606516]
IC_{95}	[0.588362720668, 0.624330774147]

	Configurazione 22
μ	0.383307630372
σ^2	0.000674378356
σ	0.025968795804
IC_{90}	[0.373784479567, 0.392830781177]
IC_{95}	[0.371926303801, 0.394688956944]

	Configurazione 23
μ	0.603268214099
σ^2	0.001980211950
σ	0.044499572467
IC_{90}	[0.586949546782, 0.619586881416]
IC_{95}	[0.583765416574, 0.622771011624]

	Configurazione 24
μ	0.383879402002
σ^2	0.000676595448
σ	0.026011448411
IC_{90}	[0.374340609840, 0.393418194164]
IC_{95}	[0.372479382101, 0.395279421903]

	Configurazione 25
μ	0.568631680854
σ^2	0.001781685095
σ	0.042210011790
IC_{90}	[0.553152630120, 0.584110731587]
IC_{95}	[0.550132327538, 0.587131034170]

	Configurazione 26
μ	0.358020594047
σ^2	0.000502806747
σ	0.022423352713
IC_{90}	[0.349797610931, 0.366243577163]
IC_{95}	[0.348193126421, 0.367848061673]

	Configurazione 27
μ	0.562330934286
σ^2	0.001208883852
σ	0.034769007066
IC_{90}	[0.549580612703, 0.575081255868]
IC_{95}	[0.547092745077, 0.577569123494]

	Configurazione 28
μ	0.357733084522
σ^2	0.000494505283
σ	0.022237474749
IC_{90}	[0.349578265671, 0.365887903372]
IC_{95}	[0.347987081505, 0.367479087538]

	Configurazione 29
μ	0.563461446046
σ^2	0.001275275949
σ	0.035711005992
IC_{90}	[0.550365679188, 0.576557212905]
IC_{95}	[0.547810407605, 0.579112484487]

	Configurazione 30
μ	0.357742599282
σ^2	0.000494897861
σ	0.022246299948
IC_{90}	[0.349584544098, 0.365900654467]
IC_{95}	[0.347992728452, 0.367492470113]

	Configurazione 31
μ	0.566730965032
σ^2	0.001564897377
σ	0.039558783821
IC_{90}	[0.552224159756, 0.581237770308]
IC_{95}	[0.549393563605, 0.584068366460]

	Configurazione 32
μ	0.358020594047
σ^2	0.000502806747
σ	0.022423352713
IC_{90}	[0.349797610931, 0.366243577163]
IC_{95}	[0.348193126421, 0.367848061673]

	Configurazione 33
μ	0.560559248464
σ^2	0.001136373258
σ	0.033710135830
IC_{90}	[0.548197231004, 0.572921265925]
IC_{95}	[0.545785130036, 0.575333366893]

	Configurazione 34
μ	0.357733084522
σ^2	0.000494505283
σ	0.022237474749
IC_{90}	[0.349578265671, 0.365887903372]
IC_{95}	[0.347987081505, 0.367479087538]

	Configurazione 35
μ	0.559435077919
σ^2	0.001078252353
σ	0.032836753085
IC_{90}	[0.547393343141, 0.571476812697]
IC_{95}	[0.545043736355, 0.573826419483]

	Configurazione 36
μ	0.357742599282
σ^2	0.000494897861
σ	0.022246299948
IC_{90}	[0.349584544098, 0.365900654467]
IC_{95}	[0.347992728452, 0.367492470113]

	Configurazione 37
μ	0.500362621284
σ^2	0.001091193063
σ	0.033033211522
IC_{90}	[0.488248842222, 0.512476400347]
IC_{95}	[0.485885178014, 0.514840064554]

	Configurazione 38
μ	0.336394630280
σ^2	0.000263306922
σ	0.016226734788
IC_{90}	[0.330444040826, 0.342345219735]
IC_{95}	[0.329282950201, 0.343506310360]

	Configurazione 39
μ	0.499378900724
σ^2	0.000998987651
σ	0.031606765898
IC_{90}	[0.487788220880, 0.510969580568]
IC_{95}	[0.485526624813, 0.513231176635]

	Configurazione 40
μ	0.336394630280
σ^2	0.000263306922
σ	0.016226734788
IC_{90}	[0.330444040826, 0.342345219735]
IC_{95}	[0.329282950201, 0.343506310360]

	Configurazione 41
μ	0.498510810696
σ^2	0.000912335098
σ	0.030204885329
IC_{90}	[0.487434221693, 0.509587399699]
IC_{95}	[0.485272936034, 0.511748685358]

	Configurazione 42
μ	0.336394630280
σ^2	0.000263306922
σ	0.016226734788
IC_{90}	[0.330444040826, 0.342345219735]
IC_{95}	[0.329282950201, 0.343506310360]

	Configurazione 43
μ	0.500151947678
σ^2	0.001076563581
σ	0.032811028340
IC_{90}	[0.488119646554, 0.512184248801]
IC_{95}	[0.485771880481, 0.514532014874]

	Configurazione 44
μ	0.336394630280
σ^2	0.000263306922
σ	0.016226734788
IC_{90}	[0.330444040826, 0.342345219735]
IC_{95}	[0.329282950201, 0.343506310360]

	Configurazione 45
μ	0.498265868555
σ^2	0.000916106655
σ	0.030267253834
IC_{90}	[0.487166408077, 0.509365329034]
IC_{95}	[0.485000659691, 0.511531077420]

	Configurazione 46
μ	0.336394630280
σ^2	0.000263306922
σ	0.016226734788
IC_{90}	[0.330444040826, 0.342345219735]
IC_{95}	[0.329282950201, 0.343506310360]

	Configurazione 47
μ	0.496889534884
σ^2	0.000814322015
σ	0.028536327989
IC_{90}	[0.486424831133, 0.507354238635]
IC_{95}	[0.484382937719, 0.509396132050]

	Configurazione 48
μ	0.336394630280
σ^2	0.000263306922
σ	0.016226734788
IC_{90}	[0.330444040826, 0.342345219735]
IC_{95}	[0.329282950201, 0.343506310360]

4.6 Job non Serviti

	Configurazione 1
μ	3.750000000000
σ^2	11.881578947368
σ	3.446966629860
IC_{90}	[2.485945121111, 5.014054878889]
IC_{95}	[2.239300266694, 5.260699733306]

	Configurazione 2
μ	0.050000000000
σ^2	0.050000000000
σ	0.223606797750
IC_{90}	[-0.032000000000, 0.132000000000]
IC_{95}	[-0.048000000000, 0.148000000000]

	Configurazione 3
μ	13.950000000000
σ^2	32.155263157895
σ	5.670561097272
IC_{90}	[11.870519346213, 16.029480653787]
IC_{95}	[11.464767023523, 16.435232976477]

	Configurazione 4
μ	0.550000000000
σ^2	1.207894736842
σ	1.099042645598
IC_{90}	[0.146964413221, 0.953035586779]
IC_{95}	[0.068323323117, 1.031676676883]

	Configurazione 5
μ	15.450000000000
σ^2	53.102631578947
σ	7.287155245975
IC_{90}	[12.777689783214, 18.122310216786]
IC_{95}	[12.256263399451, 18.643736600549]

	Configurazione 6
μ	0.650000000000
σ^2	1.186842105263
σ	1.089422831257
IC_{90}	[0.250492144863, 1.049507855137]
IC_{95}	[0.172539392642, 1.127460607358]

	Configurazione 7
μ	7.600000000000
σ^2	77.305263157895
σ	8.792341164781
IC_{90}	[4.375715305766, 10.824284694234]
IC_{95}	[3.746586584939, 11.453413415061]

	Configurazione 8
μ	0.100000000000
σ^2	0.200000000000
σ	0.447213595500
IC_{90}	[-0.064000000000, 0.264000000000]
IC_{95}	[-0.096000000000, 0.296000000000]

	Configurazione 9
μ	18.900000000000
σ^2	169.042105263158
σ	13.001619332343
IC_{90}	[14.132109238270, 23.667890761730]
IC_{95}	[13.201789089640, 24.598210910360]

	Configurazione 10
μ	0.900000000000
σ^2	2.094736842105
σ	1.447320573372
IC_{90}	[0.369245621284, 1.430754378716]
IC_{95}	[0.265683791291, 1.534316208709]

	Configurazione 11
μ	21.600000000000
σ^2	216.778947368421
σ	14.723414935687
IC_{90}	[16.200700708230, 26.999299291770]
IC_{95}	[15.147178895202, 28.052821104798]

	Configurazione 12
μ	1.100000000000
σ^2	3.147368421053
σ	1.774082416646
IC_{90}	[0.449417103466, 1.750582896534]
IC_{95}	[0.322474099264, 1.877525900736]

	Configurazione 13
μ	0.300000000000
σ^2	0.852631578947
σ	0.923380516877
IC_{90}	[-0.038617623193, 0.638617623193]
IC_{95}	[-0.104689354548, 0.704689354548]

	Configurazione 14
μ	0.000000000000
σ^2	0.000000000000
σ	0.000000000000
IC_{90}	[0.000000000000, 0.000000000000]
IC_{95}	[0.000000000000, 0.000000000000]

	Configurazione 15
μ	1.300000000000
σ^2	2.221052631579
σ	1.490319640741
IC_{90}	[0.753477211916, 1.846522788084]
IC_{95}	[0.646838619120, 1.953161380880]

	Configurazione 16
μ	0.000000000000
σ^2	0.000000000000
σ	0.000000000000
IC_{90}	[0.000000000000, 0.000000000000]
IC_{95}	[0.000000000000, 0.000000000000]

	Configurazione 17
μ	1.900000000000
σ^2	3.357894736842
σ	1.832455930396
IC_{90}	[1.228010651713, 2.571989348287]
IC_{95}	[1.096890778877, 2.703109221123]

	Configurazione 18
μ	0.000000000000
σ^2	0.000000000000
σ	0.000000000000
IC_{90}	[0.000000000000, 0.000000000000]
IC_{95}	[0.000000000000, 0.000000000000]

	Configurazione 19
μ	0.400000000000
σ^2	0.568421052632
σ	0.753937034925
IC_{90}	[0.123519868421, 0.676480131579]
IC_{95}	[0.069572525674, 0.730427474326]

	Configurazione 20
μ	0.050000000000
σ^2	0.050000000000
σ	0.223606797750
IC_{90}	[-0.032000000000, 0.132000000000]
IC_{95}	[-0.048000000000, 0.148000000000]

	Configurazione 21
μ	2.200000000000
σ^2	6.063157894737
σ	2.462348045004
IC_{90}	[1.297019671485, 3.102980328515]
IC_{95}	[1.120828387873, 3.279171612127]

	Configurazione 22
μ	0.150000000000
σ^2	0.239473684211
σ	0.489360484930
IC_{90}	[-0.029455902808, 0.329455902808]
IC_{95}	[-0.064471688722, 0.364471688722]

	Configurazione 23
μ	3.200000000000
σ^2	11.957894736842
σ	3.458018903482
IC_{90}	[1.931892084951, 4.468107915049]
IC_{95}	[1.684456394209, 4.715543605791]

	Configurazione 24
μ	0.150000000000
σ^2	0.134210526316
σ	0.366347548533
IC_{90}	[0.015654804407, 0.284345195593]
IC_{95}	[-0.010558892294, 0.310558892294]

	Configurazione 25
μ	0.400000000000
σ^2	1.621052631579
σ	1.273205651723
IC_{90}	[-0.066903799401, 0.866903799401]
IC_{95}	[-0.158006979771, 0.958006979771]

	Configurazione 26
μ	0.000000000000
σ^2	0.000000000000
σ	0.000000000000
IC_{90}	[0.000000000000, 0.000000000000]
IC_{95}	[0.000000000000, 0.000000000000]

	Configurazione 27
μ	1.000000000000
σ^2	4.736842105263
σ	2.176428750330
IC_{90}	[0.201870608037, 1.798129391963]
IC_{95}	[0.046138043751, 1.953861956249]

	Configurazione 28
μ	0.050000000000
σ^2	0.050000000000
σ	0.223606797750
IC_{90}	[-0.032000000000, 0.132000000000]
IC_{95}	[-0.048000000000, 0.148000000000]

	Configurazione 29
μ	1.000000000000
σ^2	3.684210526316
σ	1.919429739875
IC_{90}	[0.296116038271, 1.703883961729]
IC_{95}	[0.158772826227, 1.841227173773]

	Configurazione 30
μ	0.050000000000
σ^2	0.050000000000
σ	0.223606797750
IC_{90}	[-0.032000000000, 0.132000000000]
IC_{95}	[-0.048000000000, 0.148000000000]

	Configurazione 31
μ	0.300000000000
σ^2	0.852631578947
σ	0.923380516877
IC_{90}	[-0.038617623193, 0.638617623193]
IC_{95}	[-0.104689354548, 0.704689354548]

	Configurazione 32
μ	0.000000000000
σ^2	0.000000000000
σ	0.000000000000
IC_{90}	[0.000000000000, 0.000000000000]
IC_{95}	[0.000000000000, 0.000000000000]

	Configurazione 33
μ	1.400000000000
σ^2	6.778947368421
σ	2.603641175051
IC_{90}	[0.445205340345, 2.354794659655]
IC_{95}	[0.258903943339, 2.541096056661]

	Configurazione 34
μ	0.050000000000
σ^2	0.050000000000
σ	0.223606797750
IC_{90}	[-0.032000000000, 0.132000000000]
IC_{95}	[-0.048000000000, 0.148000000000]

	Configurazione 35
μ	1.600000000000
σ^2	6.252631578947
σ	2.500526260399
IC_{90}	[0.683019141565, 2.516980858435]
IC_{95}	[0.504096047236, 2.695903952764]

	Configurazione 36
μ	0.100000000000
σ^2	0.094736842105
σ	0.307793505626
IC_{90}	[-0.012872541064, 0.212872541064]
IC_{95}	[-0.034896451516, 0.234896451516]

	Configurazione 37
μ	0.000000000000
σ^2	0.000000000000
σ	0.000000000000
IC_{90}	[0.000000000000, 0.000000000000]
IC_{95}	[0.000000000000, 0.000000000000]

	Configurazione 38
μ	0.000000000000
σ^2	0.000000000000
σ	0.000000000000
IC_{90}	[0.000000000000, 0.000000000000]
IC_{95}	[0.000000000000, 0.000000000000]

	Configurazione 39
μ	0.150000000000
σ^2	0.134210526316
σ	0.366347548533
IC_{90}	[0.015654804407, 0.284345195593]
IC_{95}	[-0.010558892294, 0.310558892294]

	Configurazione 40
μ	0.000000000000
σ^2	0.000000000000
σ	0.000000000000
IC_{90}	[0.000000000000, 0.000000000000]
IC_{95}	[0.000000000000, 0.000000000000]

	Configurazione 41
μ	0.250000000000
σ^2	0.513157894737
σ	0.716350399411
IC_{90}	[-0.012696542962, 0.512696542962]
IC_{95}	[-0.063954405003, 0.563954405003]

	Configurazione 42
μ	0.050000000000
σ^2	0.050000000000
σ	0.223606797750
IC_{90}	[-0.032000000000, 0.132000000000]
IC_{95}	[-0.048000000000, 0.148000000000]

	Configurazione 43
μ	0.100000000000
σ^2	0.094736842105
σ	0.307793505626
IC_{90}	[-0.012872541064, 0.212872541064]
IC_{95}	[-0.034896451516, 0.234896451516]

	Configurazione 44
μ	0.000000000000
σ^2	0.000000000000
σ	0.000000000000
IC_{90}	[0.000000000000, 0.000000000000]
IC_{95}	[0.000000000000, 0.000000000000]

	Configurazione 45
μ	0.550000000000
σ^2	1.418421052632
σ	1.190974832913
IC_{90}	[0.113251487515, 0.986748512485]
IC_{95}	[0.028032265567, 1.071967734433]

	Configurazione 46
μ	0.050000000000
σ^2	0.050000000000
σ	0.223606797750
IC_{90}	[-0.032000000000, 0.132000000000]
IC_{95}	[-0.048000000000, 0.148000000000]

	Configurazione 47
μ	0.600000000000
σ^2	1.305263157895
σ	1.142481141155
IC_{90}	[0.181034858880, 1.018965141120]
IC_{95}	[0.099285563052, 1.100714436948]

	Configurazione 48
μ	0.050000000000
σ^2	0.050000000000
σ	0.223606797750
IC_{90}	[-0.032000000000, 0.132000000000]
IC_{95}	[-0.048000000000, 0.148000000000]

A Codice Sorgente

Per l'implementazione delle funzionalità richieste dal modello di simulazione si è proceduto modificando alcune classi e alcuni moduli della libreria *queueinglib*. La nuova libreria è stata stata chiamata *pssqueueinglib* ed è stata utilizzata nel progetto al posto della libreria *queueinglib*.

Di seguito vengono riportate le modifiche apportate, nei moduli e nelle librerie il codice aggiunto è riportato sotto il commento

```
// progettoss
```

A.1 network.ned

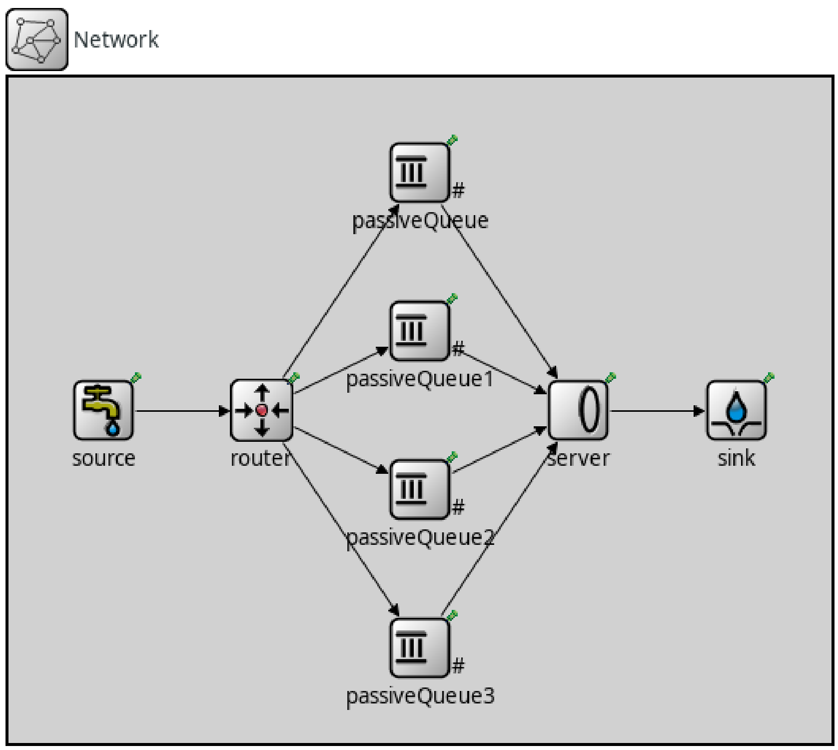


Figure 81: Visualizzazione grafica del file nework.ned

```
//
// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU Lesser General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU Lesser General Public License for more details.
//
// You should have received a copy of the GNU Lesser General Public License
// along with this program.  If not, see http://www.gnu.org/licenses/.
//
import org.omnetpp.queueing.PassiveQueue;
```

```
import org.omnetpp.queueing.Router;
import org.omnetpp.queueing.Server;
import org.omnetpp.queueing.Sink;
import org.omnetpp.queueing.Source;


//
// TODO documentation
//
network Network
{
    @display("bgb=520,420");
    submodules:
        source: Source {
            @display("p=60,210");
        }
        router: Router {
            @display("p=160,210");
        }
        passiveQueue: PassiveQueue {
            @display("p=260,60");
        }
        passiveQueue1: PassiveQueue {
            @display("p=260,160");
        }
        passiveQueue2: PassiveQueue {
            @display("p=260,260");
        }
        passiveQueue3: PassiveQueue {
            @display("p=260,360");
        }
        server: Server {
            @display("p=360,210");
        }
        sink: Sink {
            @display("p=460,210");
        }
    connections:
        source.out --> router.in++;
        router.out++ --> passiveQueue.in++;
        router.out++ --> passiveQueue1.in++;
        router.out++ --> passiveQueue2.in++;
        router.out++ --> passiveQueue3.in++;
        passiveQueue.out++ --> server.in++;
        passiveQueue1.out++ --> server.in++;
        passiveQueue2.out++ --> server.in++;
        passiveQueue3.out++ --> server.in++;
        server.out --> sink.in++;
}
```

Listing 1: "network.ned"

A.2 omnetpp.ini

```
[General]
network = Network

repeat = 20
sim-time-limit = 1000s

Network.router.routingAlgorithm = "pssRandom"
Network.server.fetchingAlgorithm = "exhaustiveService"
Network.server.checkJobDeadline = true

# [Example]
# Esponenziale di media 1/lambda = {0.5, 0.714285714286, 0.833333333333, 1}; lambda =
    {2.0, 1.4, 1.2, 1.0}
# Network.source.interArrivalTime = exponential(<1/lambda>s)

# H Uniforme su [a, b] = {[4.0, 6.0], [3.0, 7.0]}
# Network.source.jobRelativeDeadline = uniform(<a>s, <b>s)

# Code K = {1, 2, 4}
# Network.router.queueNumber = <K>

# S Esponenziale negativa di media 1/mu = {0.333333333333, 0.25}; mu = {3.0, 4.0}
# Network.server.serviceTime = exponential(<1/mu>s)

[Config n1lambda1H1K1mu1]

Network.source.interArrivalTime = exponential(0.5s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.333333333333s)

[Config n2lambda1H1K1mu2]

Network.source.interArrivalTime = exponential(0.5s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.25s)

[Config n3lambda1H1K2mu1]

Network.source.interArrivalTime = exponential(0.5s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.333333333333s)

[Config n4lambda1H1K2mu2]

Network.source.interArrivalTime = exponential(0.5s)
```

```
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.25s)
[Config n5lambda1H1K3mu1]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.333333333333s)
[Config n6lambda1H1K3mu2]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.25s)
[Config n7lambda1H2K1mu1]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.333333333333s)
[Config n8lambda1H2K1mu2]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.25s)
[Config n9lambda1H2K2mu1]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.333333333333s)
[Config n10lambda1H2K2mu2]
```

```
Network.source.interArrivalTime = exponential(0.5s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.25s)

[Config n11lambda1H2K3mu1]

Network.source.interArrivalTime = exponential(0.5s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.333333333333s)

[Config n12lambda1H2K3mu2]

Network.source.interArrivalTime = exponential(0.5s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.25s)

[Config n13lambda2H1K1mu1]

Network.source.interArrivalTime = exponential(0.714285714286s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.333333333333s)

[Config n14lambda2H1K1mu2]

Network.source.interArrivalTime = exponential(0.714285714286s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.25s)

[Config n15lambda2H1K2mu1]

Network.source.interArrivalTime = exponential(0.714285714286s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.333333333333s)

[Config n16lambda2H1K2mu2]
```

```
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.25s)
[Config n17lambda2H1K3mu1]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.333333333333s)
[Config n18lambda2H1K3mu2]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.25s)
[Config n19lambda2H2K1mu1]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.333333333333s)
[Config n20lambda2H2K1mu2]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.25s)
[Config n21lambda2H2K2mu1]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.333333333333s)
```

```
[Config n22lambda2H2K2mu2]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.25s)
[Config n23lambda2H2K3mu1]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.333333333333s)
[Config n24lambda2H2K3mu2]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.25s)
[Config n25lambda3H1K1mu1]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.333333333333s)
[Config n26lambda3H1K1mu2]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.25s)
[Config n27lambda3H1K2mu1]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.333333333333s)
```

```
[Config n28lambda3H1K2mu2]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.25s)
[Config n29lambda3H1K3mu1]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.333333333333s)
[Config n30lambda3H1K3mu2]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.25s)
[Config n31lambda3H2K1mu1]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.333333333333s)
[Config n32lambda3H2K1mu2]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.25s)
[Config n33lambda3H2K2mu1]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 2
```

```
Network.server.serviceTime = exponential(0.333333333333s)

[Config n34lambda3H2K2mu2]

Network.source.interArrivalTime = exponential(0.833333333333s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.25s)

[Config n35lambda3H2K3mu1]

Network.source.interArrivalTime = exponential(0.833333333333s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.333333333333s)

[Config n36lambda3H2K3mu2]

Network.source.interArrivalTime = exponential(0.833333333333s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.25s)

[Config n37lambda4H1K1mu1]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.333333333333s)

[Config n38lambda4H1K1mu2]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.25s)

[Config n39lambda4H1K2mu1]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 2
```

```
Network.server.serviceTime = exponential(0.333333333333s)

[Config n40lambda4H1K2mu2]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.25s)

[Config n41lambda4H1K3mu1]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.333333333333s)

[Config n42lambda4H1K3mu2]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.25s)

[Config n43lambda4H2K1mu1]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.333333333333s)

[Config n44lambda4H2K1mu2]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.25s)

[Config n45lambda4H2K2mu1]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
```

```
Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.333333333333s)

[Config n46lambda4H2K2mu2]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.25s)

[Config n47lambda4H2K3mu1]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.333333333333s)

[Config n48lambda4H2K3mu2]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.25s)
```

Listing 2: "omnetpp.ini"

A.3 Job (pssqueueinglib)

Aggiunte:

- *simtime_t absoluteDeadline*: attributo che contiene la deadline assoluta del *Job (Job.h)*
- *void setAbsoluteDeadline(simtime_t absoluteDeadline)*: metodo per impostare la deadline assoluta del *Job (Job.h, Job.cc)*
- *simtime_t getAbsoluteDeadline()*: metodo per ottenere la deadline assoluta del *Job (Job.h, Job.cc)*

A.3.1 Job.h

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// 'license' for details on this and other legal matters.
//

#ifndef __QUEUEING_JOB_H
#define __QUEUEING_JOB_H

#include <vector>
#include "Job_m.h"

namespace queueing {

class JobList;

/**
 * We extend the generated Job_Base class with support for split-join, as well
 * as the ability to enumerate all jobs in the system.
 *
 * To support split-join, Jobs manage parent-child relationships. A
 * relationship is created with the makeChildOf() or addChild() methods,
 * and lives until the parent or the child Job is destroyed.
 * It can be queried with the getParent() and getNumChildren()/getChild(k)
 * methods.
 *
 * To support enumerating all jobs in the system, each Job automatically
 * registers itself in a JobList module, if one exist in the model.
 * (If there's no JobList module, no registration takes place.) If there
 * are more than one JobList modules, the first one is chosen.
 * JobList can also be explicitly specified in the Job constructor.
 * The default JobList can be obtained with the JobList::getDefaultInstance()
 * method. Then one can query JobList for the set of Jobs currently present.
 */
class QUEUEING_API Job: public Job_Base
{
    friend class JobList;
    protected:
        Job *parent;
        std::vector<Job*> children;
        JobList *jobList;
        virtual void setParent(Job *parent); // only for addChild()
```

```
        virtual void parentDeleted();
        virtual void childDeleted(Job *child);
        // progettoss
        simtime_t absoluteDeadline;
    public:
        /**
         * Creates a job with the given name, message kind, and jobList. If
         * jobList==nullptr, the default one (or none if none exist) will be chosen.
         */
        Job(const char *name=nullptr, int kind=0, JobList *table=nullptr);

        /** Copy constructor */
        Job(const Job& job);

        /** Destructor */
        virtual ~Job();

        /** Duplicates this job */
        virtual Job *dup() const override {return new Job(*this);}

        /** Assignment operator. Does not affect parent, children and jobList. */
        Job& operator=(const Job& job);

        /** @name Parent-child relationships */
        //@{
        /** Returns the parent job. Returns nullptr if there's no parent or it no longer
             exists. */
        virtual Job *getParent();

        /** Returns the number of children. Deleted children are automatically removed
           from this list. */
        virtual int getNumChildren() const;

        /** Returns the kth child. Throws an error if index is out of range. */
        virtual Job *getChild(int k);

        /** Marks the given job as the child of this one. */
        void addChild(Job *child);

        /** Same as addChild(), but has to be invoked on the child job */
        virtual void makeChildOf(Job *parent);
        //@}

        /** Returns the JobList where this job has been registered. */
        JobList *getContainingJobList() {return jobList;}

        // progettoss
        void setAbsoluteDeadline(simtime_t absoluteDeadline);

        simtime_t getAbsoluteDeadline();

};

}; // namespace

#endif
```

A.3.2 Job.cc

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// 'license' for details on this and other legal matters.
//

#include <algorithm>
#include "Job.h"
#include "JobList.h"

namespace queueing {

Job::Job(const char *name, int kind, JobList *jobList) : Job_Base(name, kind)
{
    parent = nullptr;
    if (jobList == nullptr && JobList::getDefaultInstance() != nullptr)
        jobList = JobList::getDefaultInstance();
    this->jobList = jobList;
    if (jobList != nullptr)
        jobList->registerJob(this);
}

Job::Job(const Job& job)
{
    setName(job.getName());
    operator=(job);
    parent = nullptr;
    jobList = job.jobList;
    if (jobList != nullptr)
        jobList->registerJob(this);
}

Job::~Job()
{
    if (parent)
        parent->childDeleted(this);
    for (int i = 0; i < (int)children.size(); i++)
        children[i]->parentDeleted();
    if (jobList != nullptr)
        jobList->deregisterJob(this);
}

Job& Job::operator=(const Job& job)
{
    if (this == &job)
        return *this;
    Job_Base::operator=(job);
    // leave parent and jobList untouched
    return *this;
}

Job *Job::getParent()
{
    return parent;
```

```
}

void Job::setParent(Job *parent)
{
    this->parent = parent;
}

int Job::getNumChildren() const
{
    return children.size();
}

Job *Job::getChild(int k)
{
    if (k < 0 || k >= (int)children.size())
        throw cRuntimeError(this, "child index %d out of bounds", k);
    return children[k];
}

void Job::makeChildOf(Job *parent)
{
    parent->addChild(this);
}

void Job::addChild(Job *child)
{
    child->setParent(this);
    ASSERT(std::find(children.begin(), children.end(), child) == children.end());
    children.push_back(child);
}

void Job::parentDeleted()
{
    parent = nullptr;
}

void Job::childDeleted(Job *child)
{
    std::vector<Job *>::iterator it = std::find(children.begin(), children.end(), child)
        ;
    ASSERT(it != children.end());
    children.erase(it);
}

void Job::setAbsoluteDeadline(simtime_t absoluteDeadline)
{
    this->absoluteDeadline = absoluteDeadline;
}

simtime_t Job::getAbsoluteDeadline()
{
    return absoluteDeadline;
}

}; // namespace
```

A.4 Source (pssqueueinglib)

Aggiunte:

- *double jobRelativeDeadline @unit(s) = default(0s)*: parametro per impostare la deadline relativa dei *Job (Source.ned)*

Modifiche:

- *Job *SourceBase::createJob()*: il *Job* viene configurato con la sua deadline assoluta *(Source.cc)*

A.4.1 Source.ned

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

package org.omnetpp.queueing;

//
// A module that generates jobs. One can specify the number of jobs to be generated,
// the starting and ending time, and interval between generating jobs.
// Job generation stops when the number of jobs or the end time has been reached,
// whichever occurs first. The name, type and priority of jobs can be set as well.
// One can specify the job relative deadline.
//
simple Source
{
    parameters:
        @group(Queueing);
        @signal[created](type="long");
        @statistic[created](title="the number of jobs created";record=last;
            interpolationmode=none);
        @display("i=block/source");
        int numJobs = default(-1);                // number of jobs to be generated (-1
            means no limit)
        volatile double interArrivalTime @unit(s); // time between generated jobs
        string jobName = default("job");          // the base name of the generated job (
            will be the module name if left empty)
        volatile int jobType = default(0);        // the type attribute of the created
            job (used by classifers and other modules)
        volatile int jobPriority = default(0);    // priority of the job
        double startTime @unit(s) = default(interArrivalTime); // when the module sends
            out the first job
        double stopTime @unit(s) = default(-1s); // when the module stops the job
            generation (-1 means no limit)
        // progettoss
        double jobRelativeDeadline @unit(s) = default(0s);    // job relative deadline
    gates:
        output out;
}
```

Listing 3: "Source.ned"

A.4.2 Source.cc

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// 'license' for details on this and other legal matters.
//

#include "Source.h"
#include "Job.h"

namespace queueing {

void SourceBase::initialize()
{
    createdSignal = registerSignal("created");
    jobCounter = 0;
    WATCH(jobCounter);
    jobName = par("jobName").stringValue();
    if (jobName == "")
        jobName = getName();
}

Job *SourceBase::createJob()
{
    char buf[80];
    sprintf(buf, "%.60s-%d", jobName.c_str(), ++jobCounter);
    Job *job = new Job(buf);
    job->setKind(par("jobType"));
    job->setPriority(par("jobPriority"));
    job->setAbsoluteDeadline(simTime() + par("jobRelativeDeadline"));
    return job;
}

void SourceBase::finish()
{
    emit(createdSignal, jobCounter);
}

//----

Define_Module(Source);

void Source::initialize()
{
    SourceBase::initialize();
    startTime = par("startTime");
    stopTime = par("stopTime");
    numJobs = par("numJobs");

    // schedule the first message timer for start time
    scheduleAt(startTime, new cMessage("newJobTimer"));
}

void Source::handleMessage(cMessage *msg)
{
```

```
    ASSERT(msg->isSelfMessage());

    if ((numJobs < 0 || numJobs > jobCounter) && (stopTime < 0 || stopTime > simTime()))
        {
        // reschedule the timer for the next message
        scheduleAt(simTime() + par("interArrivalTime").doubleValue(), msg);

        Job *job = createJob();
        send(job, "out");
    }
    else {
        // finished
        delete msg;
    }
}

//----

Define_Module(SourceOnce);

void SourceOnce::initialize()
{
    SourceBase::initialize();
    simtime_t time = par("time");
    scheduleAt(time, new cMessage("newJobTimer"));
}

void SourceOnce::handleMessage(cMessage *msg)
{
    ASSERT(msg->isSelfMessage());
    delete msg;

    int n = par("numJobs");
    for (int i = 0; i < n; i++) {
        Job *job = createJob();
        send(job, "out");
    }
}

}; //namespace
```

Listing 4: "Source.cc"

A.5 Router (pssqueueinglib)

Aggiunte:

- *int queueNumber = default(sizeof(out)-1)*: numero di code da utilizzare *(Router.ned)*
- *ALG_PSSRANDOM*: algoritmo che consente di inoltrare i messaggi solo alle prime n code in maniera casuale *(Router.h)*
- *int queueNumber*: numero di code da utilizzare *(Router.h)*

Modifiche:

- *string routingAlgorithm @enum("random","roundRobin","shortestQueue","minDelay","pssRandom") = default("random")*: *"pssRandom"* permette di inoltrare i messaggi solo alle prime n code in maniera casuale *(Router.ned)*
- *void Router::initialize()*: inizializzazione dell'algoritmo di instradamento e del numero di code da utilizzare *(Router.cc)*
- *void Router::handleMessage(cMessage *msg)*: implementazione dell'algoritmo di instradamento *ALG_PSSRANDOM (Router.cc)*

A.5.1 Router.ned

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

package org.omnetpp.queueing;

//
// Sends the messages to different outputs depending on a set algorithm.
// Sends the messages to first queueNumber-th queues.
//
// @author rhornig, Samuele Evangelisti
// @todo minDelay not implemented
//
simple Router
{
    parameters:
        @group(Queueing);
        @display("i=block/routing");
        string routingAlgorithm @enum("random","roundRobin","shortestQueue","minDelay","
           pssRandom") = default("random");
        volatile int randomGateIndex = default(intuniform(0, sizeof(out)-1));    // the
           destination gate in case of random routing
        // progettoss
        int queueNumber = default(sizeof(out)-1);    // queue number limit
    gates:
        input in[];
        output out[];
```

```
}
```

Listing 5: "Router.ned"

A.5.2 Router.h

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// 'license' for details on this and other legal matters.
//

#ifndef __QUEUEING_ROUTER_H
#define __QUEUEING_ROUTER_H

#include "QueueingDefs.h"

namespace queueing {

// routing algorithms
enum {
    ALG_RANDOM,
    ALG_ROUND_ROBIN,
    ALG_MIN_QUEUE_LENGTH,
    ALG_MIN_DELAY,
    ALG_MIN_SERVICE_TIME,
    // progettoss
    ALG_PSSRANDOM
};

/**
 * Sends the messages to different outputs depending on a set algorithm.
 * Sends the messages to first queueNumber-th queues.
 */
class QUEUEING_API Router : public cSimpleModule
{
    private:
        int routingAlgorithm;  // the algorithm we are using for routing
        int rrCounter;         // msgCounter for round robin routing
        // progettoss
        int queueNumber;
    protected:
        virtual void initialize() override;
        virtual void handleMessage(cMessage *msg) override;
};

}; //namespace

#endif
```

Listing 6: "Router.h"

A.5.3 Router.cc

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#include "Router.h"

namespace queueing {

Define_Module(Router);

void Router::initialize()
{
    const char *algName = par("routingAlgorithm");
    if (strcmp(algName, "random") == 0) {
        routingAlgorithm = ALG_RANDOM;
    }
    else if (strcmp(algName, "roundRobin") == 0) {
        routingAlgorithm = ALG_ROUND_ROBIN;
    }
    else if (strcmp(algName, "minQueueLength") == 0) {
        routingAlgorithm = ALG_MIN_QUEUE_LENGTH;
    }
    else if (strcmp(algName, "minDelay") == 0) {
        routingAlgorithm = ALG_MIN_DELAY;
    }
    else if (strcmp(algName, "minServiceTime") == 0) {
        routingAlgorithm = ALG_MIN_SERVICE_TIME;
    }
    else if (strcmp(algName, "pssRandom") == 0) {
        routingAlgorithm = ALG_PSSRANDOM;
    }
    rrCounter = 0;
    int qn = par("queueNumber").intValue() - 1;
    if (qn < 0 || qn > gateSize("out") - 1)
        throw cRuntimeError("Invalid queue number");
    else
        queueNumber = qn;
}

void Router::handleMessage(cMessage *msg)
{
    int outGateIndex = -1;  // by default we drop the message

    switch (routingAlgorithm) {
        case ALG_RANDOM:
            outGateIndex = par("randomGateIndex");
            break;

        case ALG_ROUND_ROBIN:
            outGateIndex = rrCounter;
            rrCounter = (rrCounter + 1) % gateSize("out");
            break;
```

```
        case ALG_MIN_QUEUE_LENGTH:
            // TODO implementation missing
            outGateIndex = -1;
            break;

        case ALG_MIN_DELAY:
            // TODO implementation missing
            outGateIndex = -1;
            break;

        case ALG_MIN_SERVICE_TIME:
            // TODO implementation missing
            outGateIndex = -1;
            break;

        case ALG_PSSRANDOM:
            outGateIndex = intuniform(0, queueNumber);
            break;

        default:
            outGateIndex = -1;
            break;
    }

    // send out if the index is legal
    if (outGateIndex < 0 || outGateIndex >= gateSize("out"))
        throw cRuntimeError("Invalid output gate selected during routing");

    send(msg, "out", outGateIndex);
}

}; //namespace
```

Listing 7: "Router.cc"

A.6 SelectionStrategies (pssqueueinglib)

Aggiunte:

- *class QUEUEING_API ExhaustiveServiceSelectionStrategy : public SelectionStrategy*: implementa la *SelectionStrategy* per ottenere un exhaustive service *(SelectionStrategies.h, SelectionStrategies.cc)*

Modifiche:

- *SelectionStrategy *SelectionStrategy::create(const char *algName, cSimpleModule *module, bool selectOnInGate*: inizializzazione di *ExhaustiveServieSelectionStrategy (SelectionStrategies.cc)*

A.6.1 SelectionStrategies.h

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#ifndef __QUEUEING_SELECTIONSTRATEGIES_H
#define __QUEUEING_SELECTIONSTRATEGIES_H

#include "QueueingDefs.h"

namespace queueing {

/**
 * Selection strategies used in queue, server and router classes to decide
 * which module to choose for further interaction.
 */
class QUEUEING_API SelectionStrategy : public cObject
{
    protected:
        bool isInputGate;
        int gateSize;        // the size of the gate vector
        cModule *hostModule; // the module using the strategy
    public:
        // on which module's gates should be used for selection
        // if selectOnInGate is true, then we will use "in" gate otherwise "out" is used
        SelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual ~SelectionStrategy();

        static SelectionStrategy * create(const char *algName, cSimpleModule *module,
            bool selectOnInGate);

        // which gate index the selection strategy selected
        virtual int select() = 0;
        // returns the i-th module's gate which connects to our host module
        cGate *selectableGate(int i);
    protected:
        // is this module selectable according to the policy? (queue is selectable if
            not empty, server is selectable if idle)
        virtual bool isSelectable(cModule *module);
};
```

```
/**
 * Priority based selection. The first selectable index will be returned.
 */
class QUEUEING_API PrioritySelectionStrategy : public SelectionStrategy
{
    public:
        PrioritySelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

/**
 * Random selection from the selectable modules, with uniform distribution.
 */
class QUEUEING_API RandomSelectionStrategy : public SelectionStrategy
{
    public:
        RandomSelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

/**
 * Uses Round Robin selection, but skips any module that is not available currently.
 */
class QUEUEING_API RoundRobinSelectionStrategy : public SelectionStrategy
{
    protected:
        int lastIndex; // the index of the module last time used
    public:
        RoundRobinSelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

/**
 * Chooses the shortest queue. If there are more than one
 * with the same length, it chooses by priority among them.
 * This strategy is for output only (i.e. for router module).
 */
class QUEUEING_API ShortestQueueSelectionStrategy : public SelectionStrategy
{
    public:
        ShortestQueueSelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

/**
 * Chooses the longest queue (where length>0 of course).
 * Input strategy (for servers).
 */
class QUEUEING_API LongestQueueSelectionStrategy : public SelectionStrategy
{
    public:
        LongestQueueSelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

// progettoss
/**
 * End all the tasks in a queue, then chooses cyclically the next one.
```

```
 * Input strategy (for servers).
 */
class QUEUEING_API ExhaustiveServiceSelectionStrategy : public SelectionStrategy
{
    private:
        int actualInputGate;    // actual input gate
    public:
        ExhaustiveServiceSelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

}; //namespace

#endif
```

Listing 8: "SelectionStrategies.h"

A.6.2 SelectionStrategies.cc

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#include "SelectionStrategies.h"
#include "PassiveQueue.h"
#include "Server.h"

namespace queueing {

SelectionStrategy::SelectionStrategy(cSimpleModule *module, bool selectOnInGate)
{
    hostModule = module;
    isInputGate = selectOnInGate;
    gateSize = isInputGate ? hostModule->gateSize("in") : hostModule->gateSize("out");
}

SelectionStrategy::~SelectionStrategy()
{
}

SelectionStrategy *SelectionStrategy::create(const char *algName, cSimpleModule *module,
    bool selectOnInGate)
{
    SelectionStrategy *strategy = nullptr;

    if (strcmp(algName, "priority") == 0) {
        strategy = new PrioritySelectionStrategy(module, selectOnInGate);
    }
    else if (strcmp(algName, "random") == 0) {
        strategy = new RandomSelectionStrategy(module, selectOnInGate);
    }
    else if (strcmp(algName, "roundRobin") == 0) {
        strategy = new RoundRobinSelectionStrategy(module, selectOnInGate);
```

```
    }
    else if (strcmp(algName, "shortestQueue") == 0) {
        strategy = new ShortestQueueSelectionStrategy(module, selectOnInGate);
    }
    else if (strcmp(algName, "longestQueue") == 0) {
        strategy = new LongestQueueSelectionStrategy(module, selectOnInGate);
    }
    else if (strcmp(algName, "exhaustiveService") == 0) {
        strategy = new ExhaustiveServiceSelectionStrategy(module, selectOnInGate);
    }

    return strategy;
}

cGate *SelectionStrategy::selectableGate(int i)
{
    if (isInputGate)
        return hostModule->gate("in", i)->getPreviousGate();
    else
        return hostModule->gate("out", i)->getNextGate();
}

bool SelectionStrategy::isSelectable(cModule *module)
{
    if (isInputGate) {
        IPassiveQueue *pqueue = dynamic_cast<IPassiveQueue *>(module);
        if (pqueue != nullptr)
            return pqueue->length() > 0;
    }
    else {
        IServer *server = dynamic_cast<IServer *>(module);
        if (server != nullptr)
            return server->isIdle();
    }

    throw cRuntimeError("Only IPassiveQueue (as input) and IServer (as output) is
        supported by this Strategy");
}

//
    ------------------------------------------------------------------------------------------------


PrioritySelectionStrategy::PrioritySelectionStrategy(cSimpleModule *module, bool
    selectOnInGate) :
    SelectionStrategy(module, selectOnInGate)
{
}

int PrioritySelectionStrategy::select()
{
    // return the smallest selectable index
    for (int i = 0; i < gateSize; i++)
        if (isSelectable(selectableGate(i)->getOwnerModule()))
            return i;

    // if none of them is selectable return an invalid no.
    return -1;
}
```

```
//
    ---------------------------------------------------------------------------------------------------------------------------


RandomSelectionStrategy::RandomSelectionStrategy(cSimpleModule *module, bool
    selectOnInGate) :
     SelectionStrategy(module, selectOnInGate)
{
}

int RandomSelectionStrategy::select()
{
     // return the smallest selectable index
     int noOfSelectables = 0;
     for (int i = 0; i < gateSize; i++)
          if (isSelectable(selectableGate(i)->getOwnerModule()))
               noOfSelectables++;

     int rnd = hostModule->intuniform(1, noOfSelectables);

     for (int i = 0; i < gateSize; i++)
          if (isSelectable(selectableGate(i)->getOwnerModule()) && (--rnd == 0))
               return i;

     return -1;
}

//
    ---------------------------------------------------------------------------------------------------------------------------


RoundRobinSelectionStrategy::RoundRobinSelectionStrategy(cSimpleModule *module, bool
    selectOnInGate) :
     SelectionStrategy(module, selectOnInGate)
{
     lastIndex = -1;
}

int RoundRobinSelectionStrategy::select()
{
     // return the smallest selectable index
     for (int i = 0; i < gateSize; ++i) {
          lastIndex = (lastIndex+1) % gateSize;
          if (isSelectable(selectableGate(lastIndex)->getOwnerModule()))
               return lastIndex;
     }

     // if none of them is selectable return an invalid no.
     return -1;
}

//
    ---------------------------------------------------------------------------------------------------------------------------


ShortestQueueSelectionStrategy::ShortestQueueSelectionStrategy(cSimpleModule *module,
    bool selectOnInGate) :
     SelectionStrategy(module, selectOnInGate)
```

```
{
}

int ShortestQueueSelectionStrategy::select()
{
    // return the smallest selectable index
    int result = -1;  // by default none of them is selectable
    int sizeMin = INT_MAX;
    for (int i = 0; i < gateSize; ++i) {
        cModule *module = selectableGate(i)->getOwnerModule();
        int length = (check_and_cast<IPassiveQueue *>(module))->length();
        if (isSelectable(module) && (length < sizeMin)) {
            sizeMin = length;
            result = i;
        }
    }
    return result;
}

// ----------------------------------------------------------------------------------------------------------------------------

LongestQueueSelectionStrategy::LongestQueueSelectionStrategy(cSimpleModule *module, bool
     selectOnInGate) :
    SelectionStrategy(module, selectOnInGate)
{
}

int LongestQueueSelectionStrategy::select()
{
    // return the longest selectable queue
    int result = -1;  // by default none of them is selectable
    int sizeMax = -1;
    for (int i = 0; i < gateSize; ++i) {
        cModule *module = selectableGate(i)->getOwnerModule();
        int length = (check_and_cast<IPassiveQueue *>(module))->length();
        if (isSelectable(module) && length > sizeMax) {
            sizeMax = length;
            result = i;
        }
    }
    return result;
}

// ----------------------------------------------------------------------------------------------------------------------------

ExhaustiveServiceSelectionStrategy::ExhaustiveServiceSelectionStrategy(cSimpleModule *
    module, bool selectOnInGate) :
    SelectionStrategy(module, selectOnInGate)
{
    actualInputGate = 0;
}

int ExhaustiveServiceSelectionStrategy::select()
{
    // previously selected queue is not empty
```

```
    if (isSelectable(selectableGate(actualInputGate)->getOwnerModule()))
        return actualInputGate;
    // scan cyclically the next non empty queue
    else {
        for (int i = 1; i < gateSize; i++) {
            int gn = (actualInputGate + i) % gateSize;
            if (isSelectable(selectableGate(gn)->getOwnerModule())) {
                actualInputGate = gn;
                return gn;
            }
        }
    }

    // if none of them is selectable return an invalid no.
    return -1;
}

}; //namespace
```

Listing 9: "SelectionStrategies.cc"

A.7 Server (pssqueueinglib)

Aggiunte:

- *@signal[droppedForDeadline](type="long")*: *signal* per la registrazione dei *Job* scartati a causa di un inizio di servizio successivo alla loro *absoluteDeadline (Server.ned)*
- *@statistic[droppedForDeadline](title="drop event for deadline reached";record=vector?,count;interpolationm* statistica relativa ai *Job* scartati a causa di un inizio di servizio successivo alla loro *absoluteDeadline (Server.ned)*
- *bool checkJobDeadline = default(true)*: specifica se sia necessario controllare *Job.absoluteDeadline* prima dell'inizio del servizio *(Server.ned)*
- *simsignal_t droppedForDeadlineSignal*: *signal* per la registrazione dei *Job* scartati a causa di un inizio di servizio successivo alla loro *absoluteDeadline (Server.h)*
- *bool checkJobDeadline*: specifica se sia necessario controllare *Job.absoluteDeadline* prima dell'inizio del servizio *(Server.h)*

Modifiche:

- *void Server::initialize()*: configurazione dei nuovi valori aggiunti *(Server.cc)*
- *void Server::handleMessage(cMessage *msg)*: modifiche al metodo per implementare il controllo di *Job.absoluteDeadline* e richiedere un nuovo *Job* in caso quello attuale venga scartato

A.7.1 Server.ned

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// 'license' for details on this and other legal matters.
//

package org.omnetpp.queueing;

//
// Queue server. It serves multiple input queues (PassiveQueue), using a preset
// algorithm. Inputs must be connected to Passive Queues (PassiveQueue)
//
simple Server
{
    parameters:
        @group(Queueing);
        @display("i=block/server");
        @signal[busy](type="bool");
        @statistic[busy](title="server busy state";record=vector?,timeavg;
            interpolationmode=sample-hold);
        // progettoss
        @signal[droppedForDeadline](type="long");
        @statistic[droppedForDeadline](title="drop event for deadline reached";record=
            vector?,count;interpolationmode=none);
```

```
        string fetchingAlgorithm @enum("priority","random","roundRobin","longestQueue","
            exhaustiveService") = default("priority");
              // how the next job will be choosen from the attached queues
        volatile double serviceTime @unit(s); // service time of a job
        // progettoss
        bool checkJobDeadline = default(false);
    gates:
        input in[];
        output out;
}
```

Listing 10: "Server.ned"

A.7.2 Server.h

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// 'license' for details on this and other legal matters.
//

#ifndef __QUEUEING_SERVER_H
#define __QUEUEING_SERVER_H

#include "IServer.h"

namespace queueing {

class Job;
class SelectionStrategy;

/**
 * The queue server. It cooperates with several Queues that which queue up
 * the jobs, and send them to Server on request.
 *
 * @see PassiveQueue
 */
class QUEUEING_API Server : public cSimpleModule, public IServer
{
    private:
        simsignal_t busySignal;
        bool allocated;

        SelectionStrategy *selectionStrategy;

        Job *jobServiced;
        cMessage *endServiceMsg;

        // progettoss
        simsignal_t droppedForDeadlineSignal;
        bool checkJobDeadline;

    public:
        Server();
        virtual ~Server();
```

```
    protected:
        virtual void initialize() override;
        virtual int numInitStages() const override {return 2;}
        virtual void handleMessage(cMessage *msg) override;
        virtual void refreshDisplay() const override;
        virtual void finish() override;

    public:
        virtual bool isIdle() override;
        virtual void allocate() override;
};

}; //namespace

#endif
```

Listing 11: "Server.h"

A.7.3 Server.cc

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#include "Server.h"
#include "Job.h"
#include "SelectionStrategies.h"
#include "IPassiveQueue.h"

namespace queueing {

Define_Module(Server);

Server::Server()
{
    selectionStrategy = nullptr;
    jobServiced = nullptr;
    endServiceMsg = nullptr;
    allocated = false;
}

Server::~Server()
{
    delete selectionStrategy;
    delete jobServiced;
    cancelAndDelete(endServiceMsg);
}

void Server::initialize()
{
    busySignal = registerSignal("busy");
    emit(busySignal, false);
```

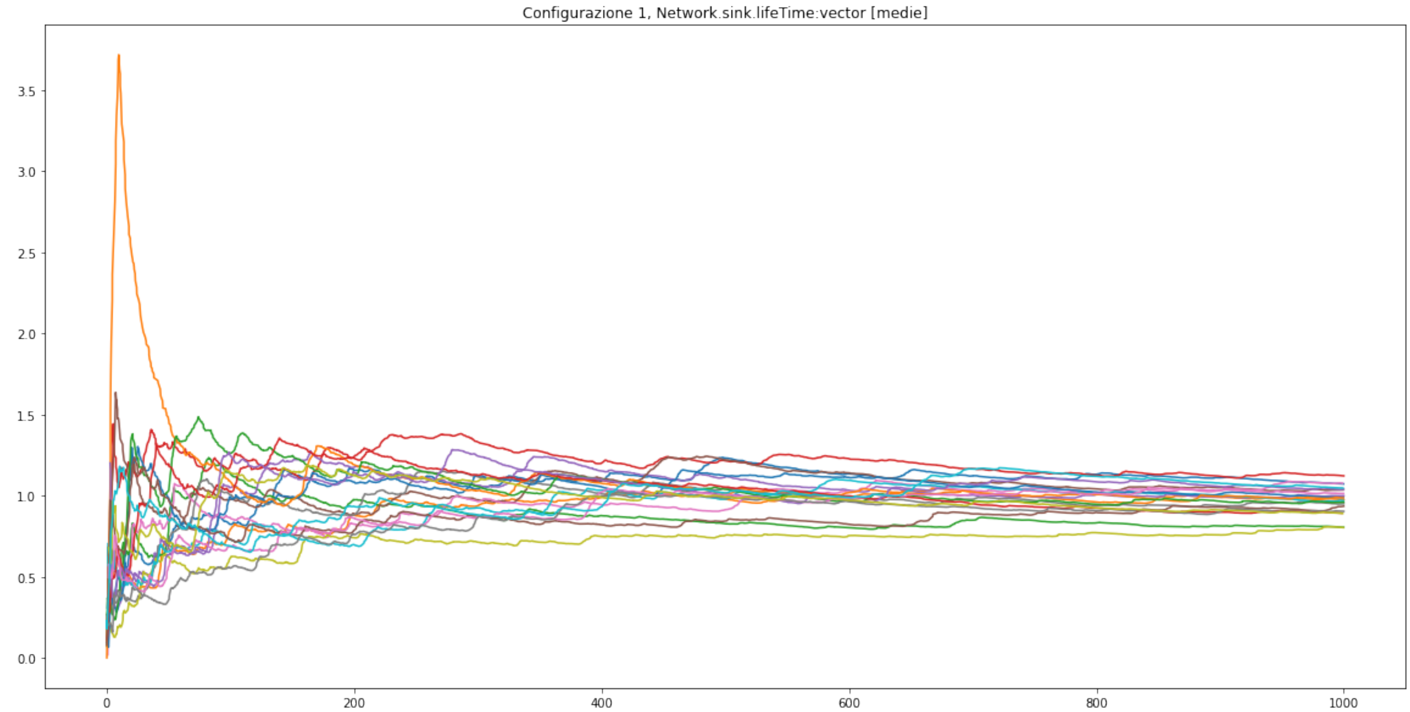
```
    endServiceMsg = new cMessage("end-service");
    jobServiced = nullptr;
    allocated = false;
    selectionStrategy = SelectionStrategy::create(par("fetchingAlgorithm"), this, true);
    if (!selectionStrategy)
        throw cRuntimeError("invalid selection strategy");
    droppedForDeadlineSignal = registerSignal("droppedForDeadline");
    checkJobDeadline = par("checkJobDeadline").boolValue();
}

void Server::handleMessage(cMessage *msg)
{
    if (msg == endServiceMsg) {
        ASSERT(jobServiced != nullptr);
        ASSERT(allocated);
        simtime_t d = simTime() - endServiceMsg->getSendingTime();
        jobServiced->setTotalServiceTime(jobServiced->getTotalServiceTime() + d);
        send(jobServiced, "out");
        jobServiced = nullptr;
        allocated = false;
        emit(busySignal, false);

        // examine all input queues, and request a new job from a non empty queue
        int k = selectionStrategy->select();
        if (k >= 0) {
            EV << "requesting job from queue " << k << endl;
            cGate *gate = selectionStrategy->selectableGate(k);
            check_and_cast<IPassiveQueue *>(gate->getOwnerModule())->request(gate->
                getIndex());
        }
    }
    else {
        if (!allocated)
            error("job arrived, but the sender did not call allocate() previously");
        if (jobServiced)
            throw cRuntimeError("a new job arrived while already servicing one");
        jobServiced = check_and_cast<Job *>(msg);
        simtime_t serviceTime = par("serviceTime");
        if (checkJobDeadline) {
            if (jobServiced->getAbsoluteDeadline() < simTime()) {
                EV << "Dropped!" << endl;
                if (hasGUI())
                    bubble("Dropped!");
                emit(droppedForDeadlineSignal, 1);
                delete msg;
                allocated = false;
                jobServiced = nullptr;
                int k = selectionStrategy->select();
                if (k >= 0) {
                    EV << "requesting job from queue " << k << endl;
                    cGate *gate = selectionStrategy->selectableGate(k);
                    check_and_cast<IPassiveQueue *>(gate->getOwnerModule())->request(
                        gate->getIndex());
                }
            }
            else {
                scheduleAt(simTime()+serviceTime, endServiceMsg);
                emit(busySignal, true);
```

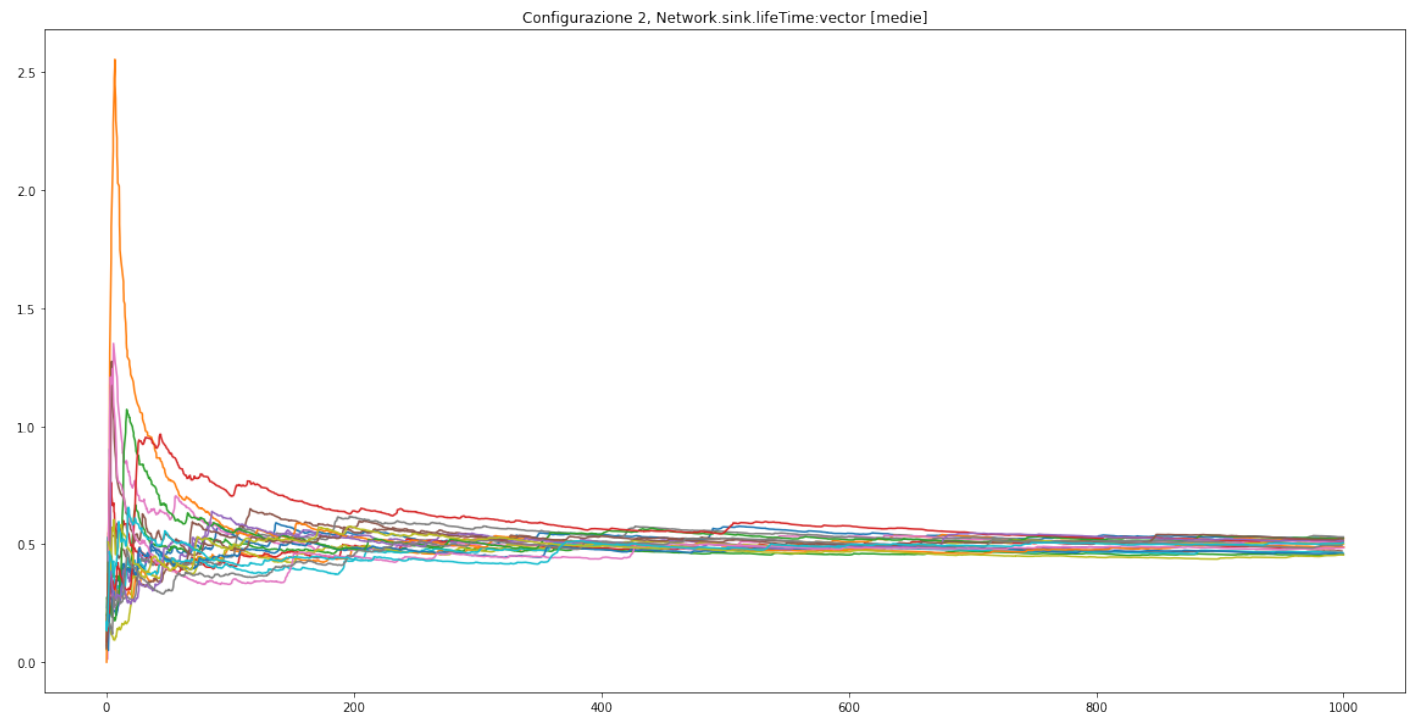
```
            }
        }
        else {
            scheduleAt(simTime()+serviceTime, endServiceMsg);
            emit(busySignal, true);
        }
    }
}

void Server::refreshDisplay() const
{
    getDisplayString().setTagArg("i2", 0, jobServiced ? "status/execute" : "");
}

void Server::finish()
{
}

bool Server::isIdle()
{
    return !allocated;  // we are idle if nobody has allocated us for processing
}

void Server::allocate()
{
    allocated = true;
}

}; //namespace
```

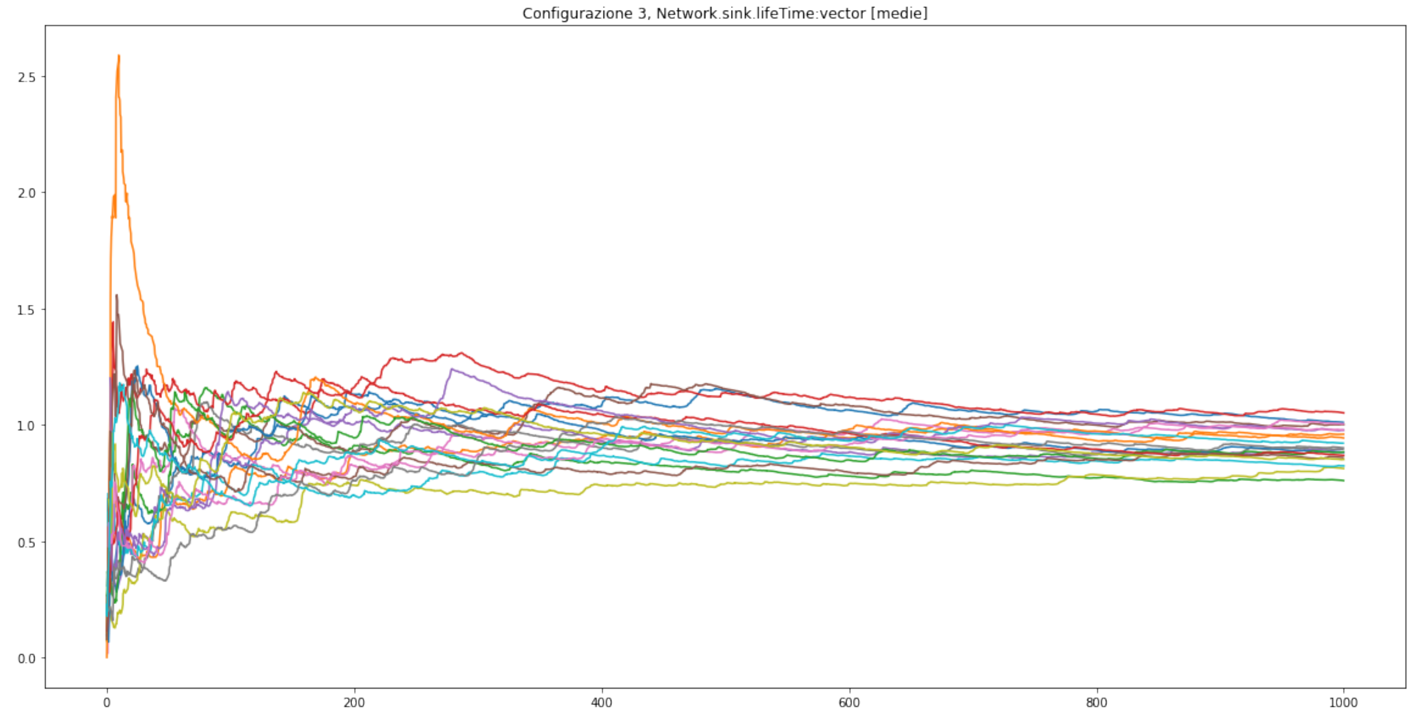
Listing 12: "Server.cc"

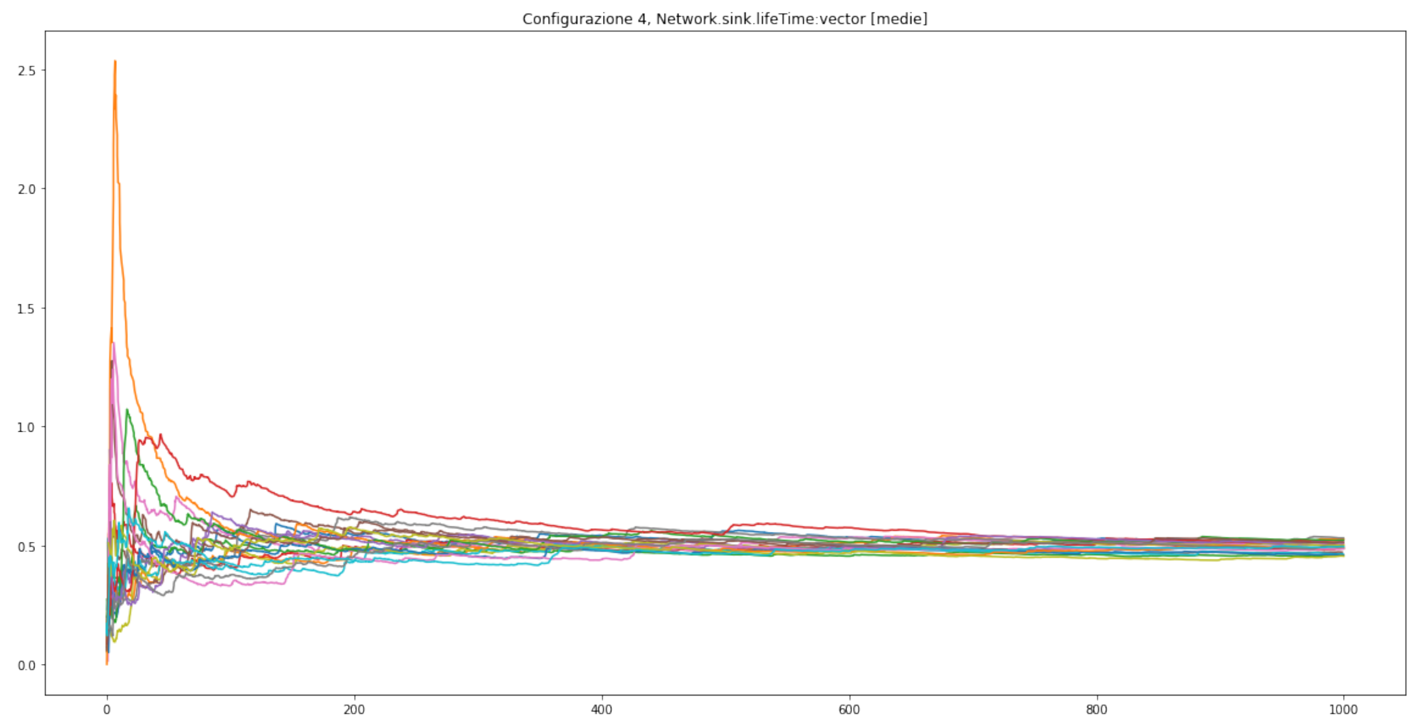
B Grafici

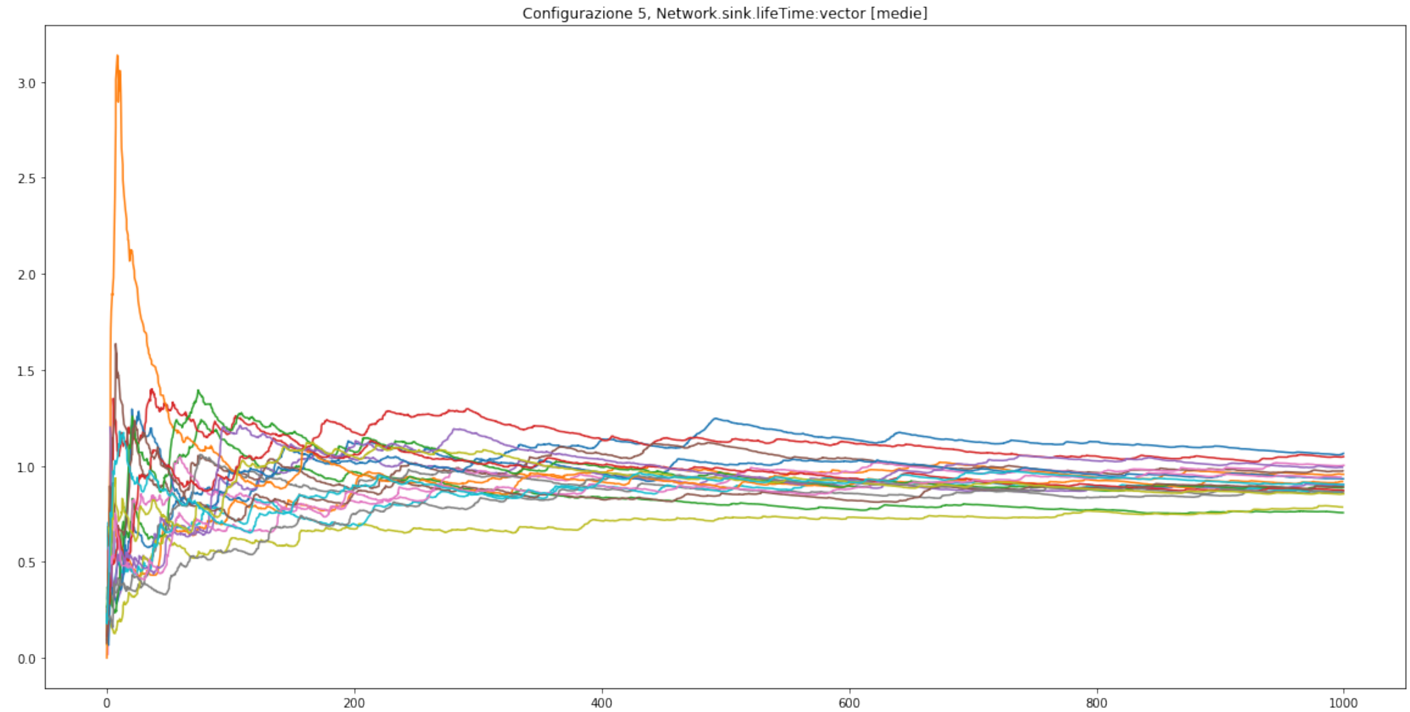
B.1 Network.sink.lifetime:vector [medie]

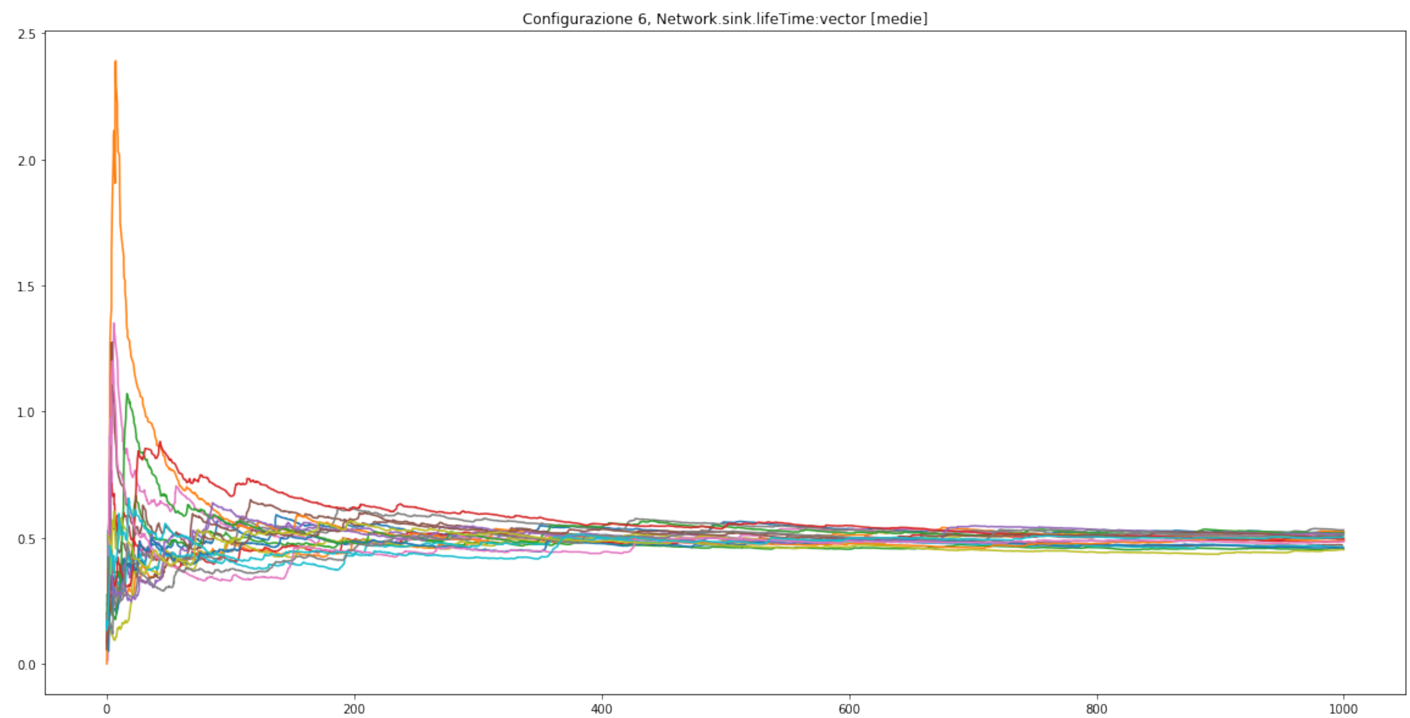


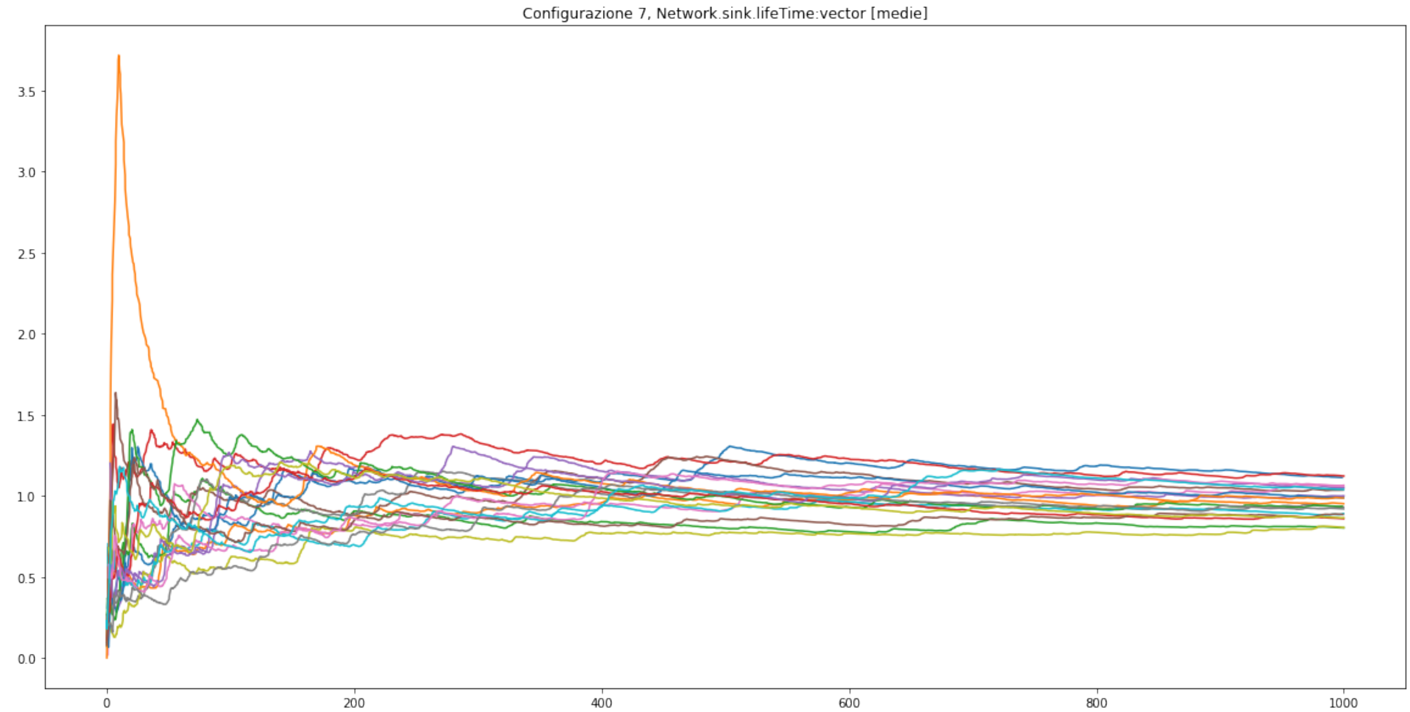


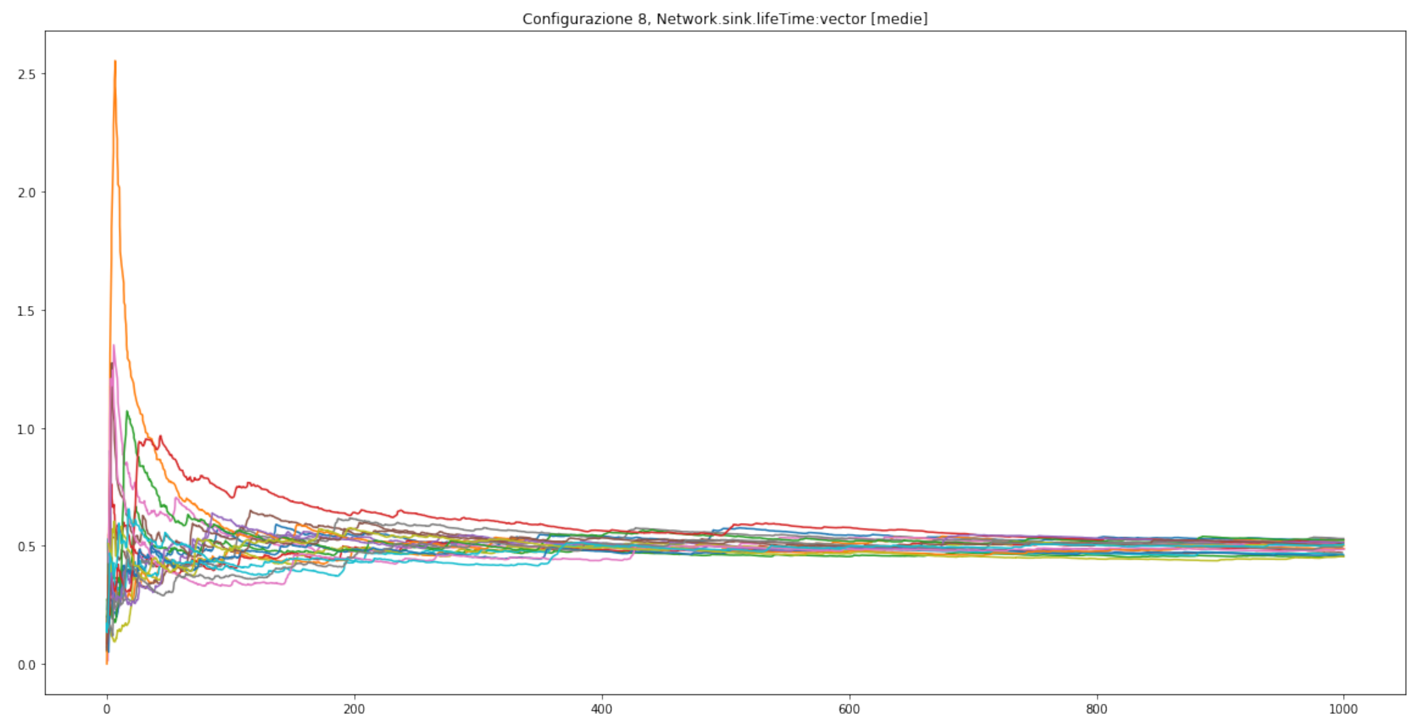

Configurazione 3, Network.sink.lifeTime:vector [medie]
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000


Configurazione 4, Network.sink.lifeTime:vector [medie]
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

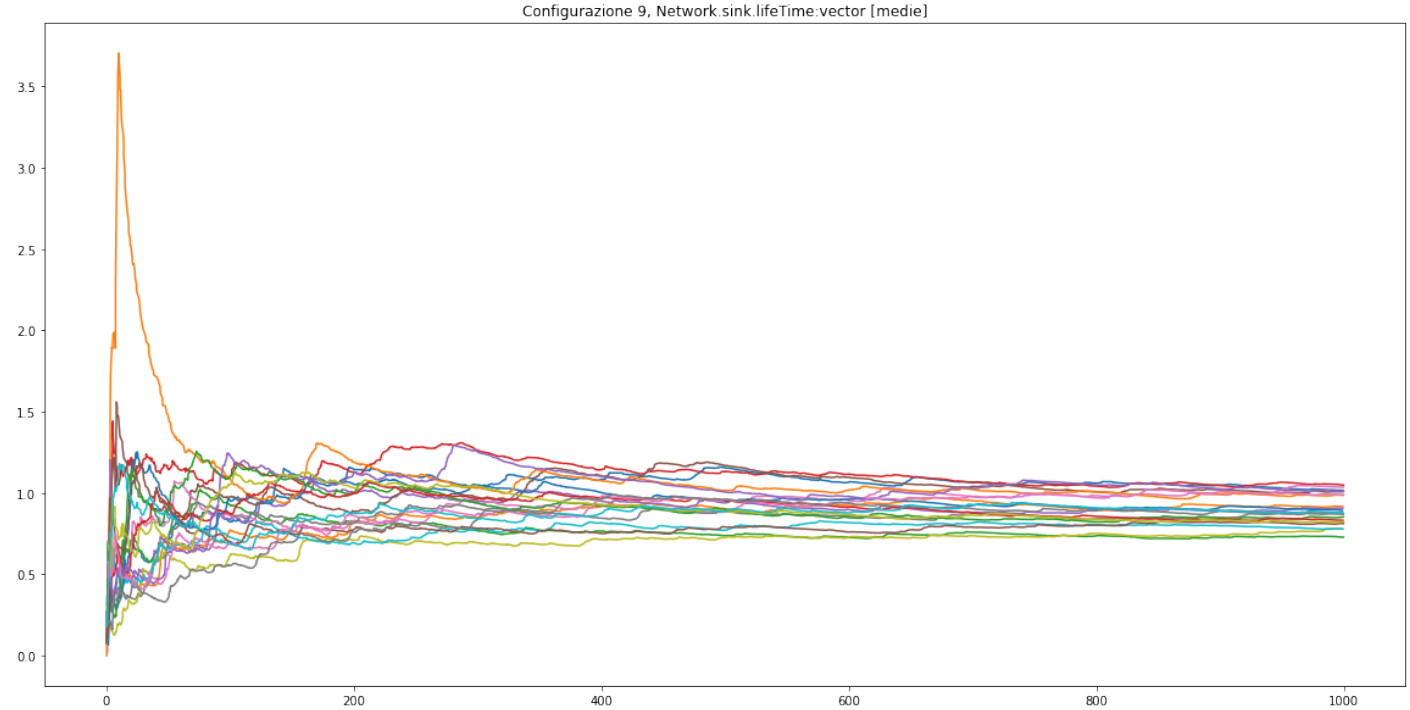

Configurazione 5, Network.sink.lifeTime:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000


Configurazione 6, Network.sink.lifeTime:vector [medie]
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

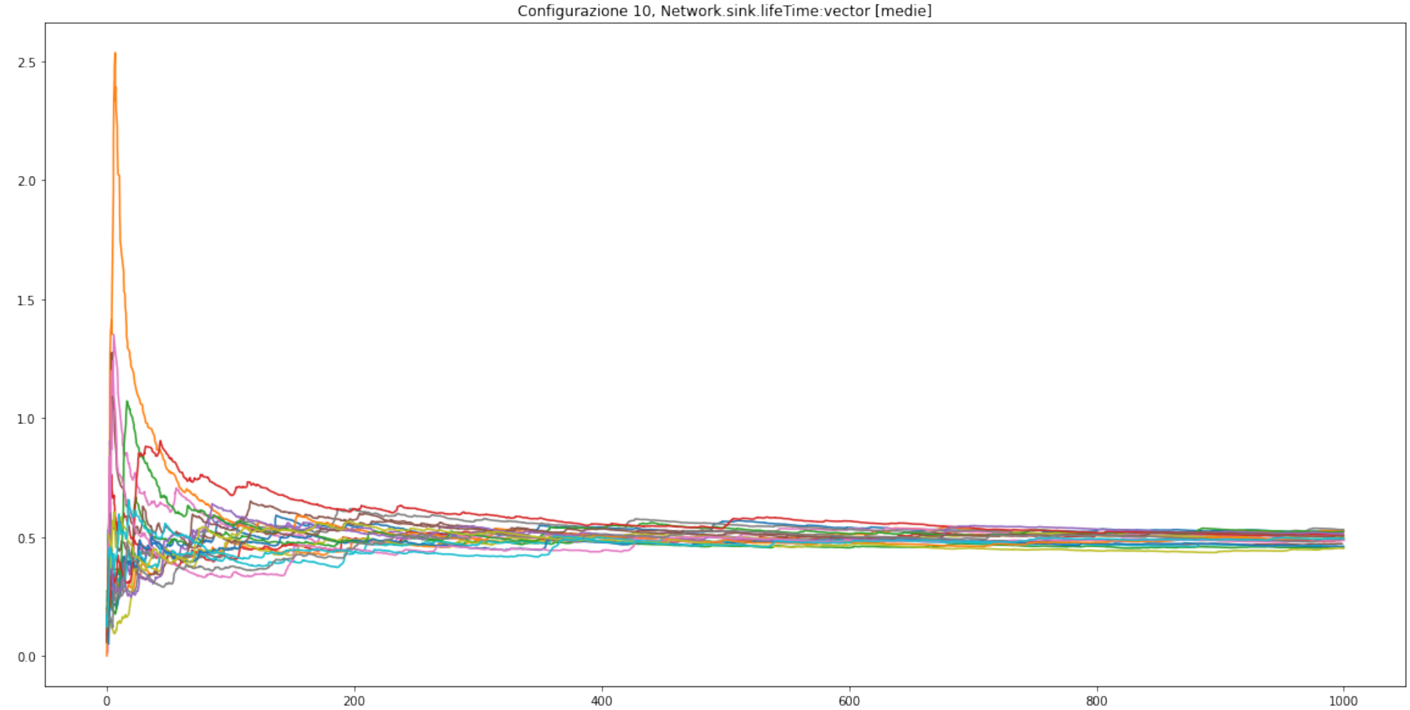

Configurazione 7, Network.sink.lifeTime:vector [medie]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
200
400
600
800
1000

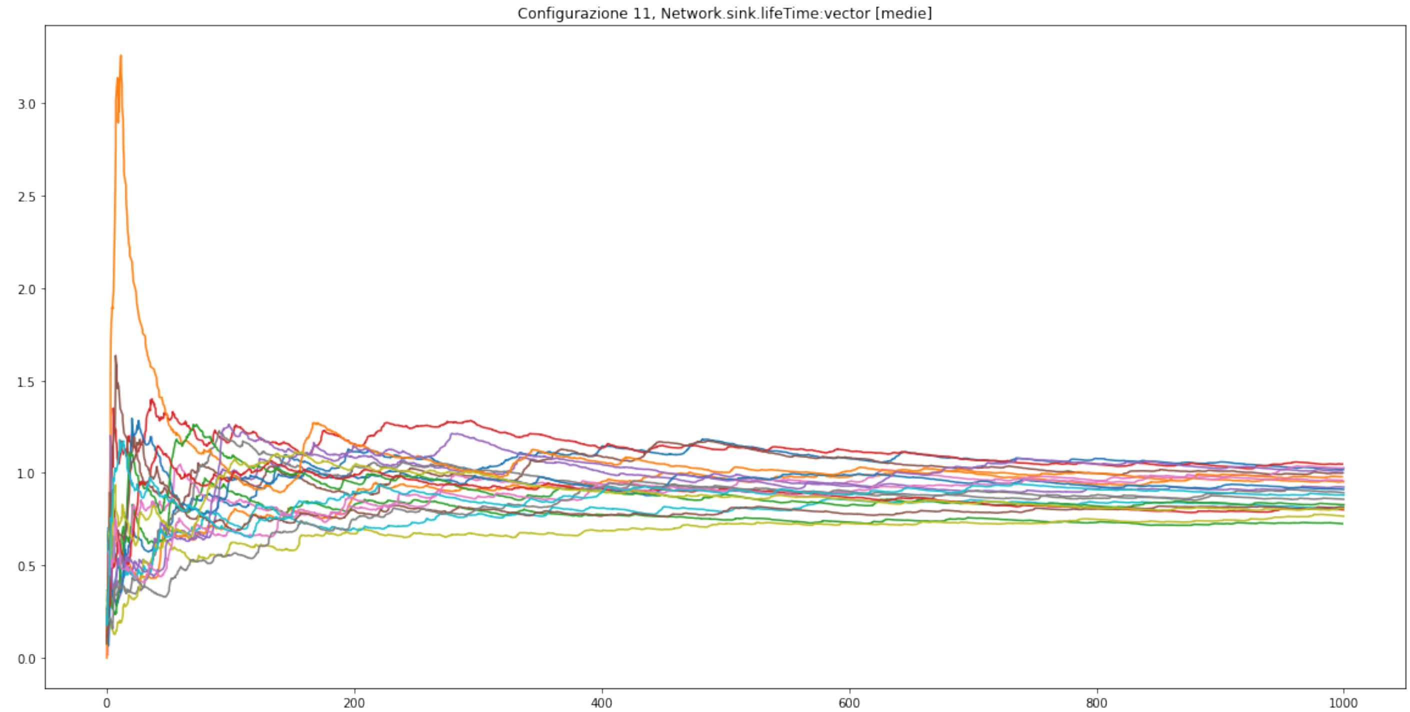

Configurazione 8, Network.sink.lifeTime:vector [medie]
0.0
0.5
1.0
1.5
2.0
2.5
0
200
400
600
800
1000

Configurazione 9, Network.sink.lifeTime:vector [medie]

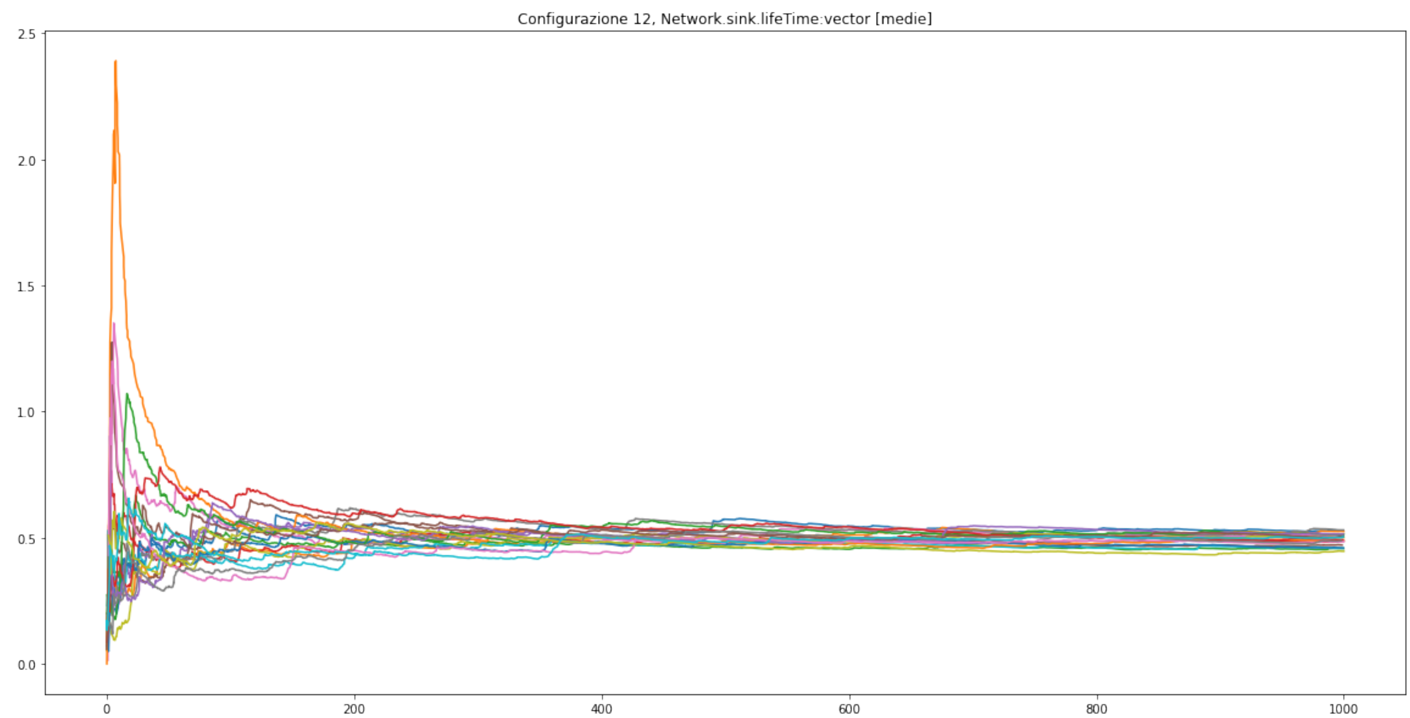


Configurazione 10, Network.sink.lifeTime:vector [medie]

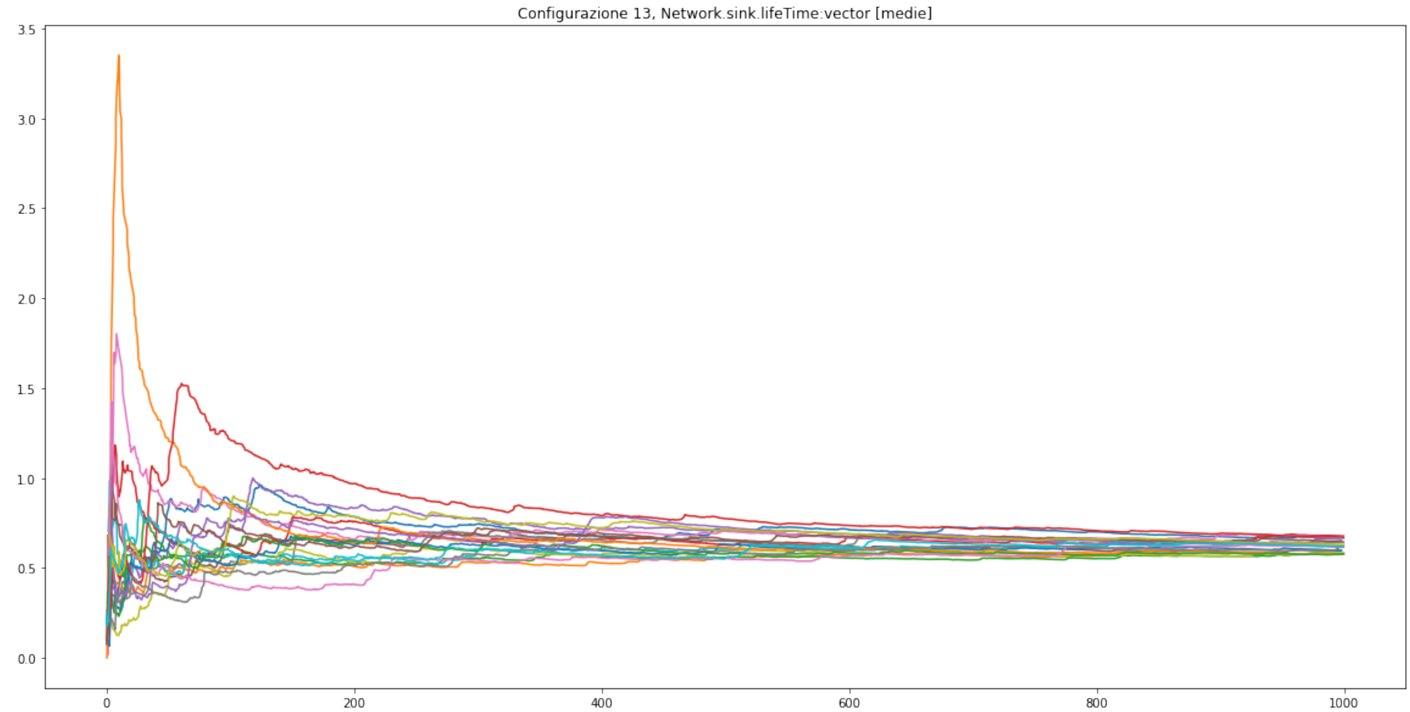


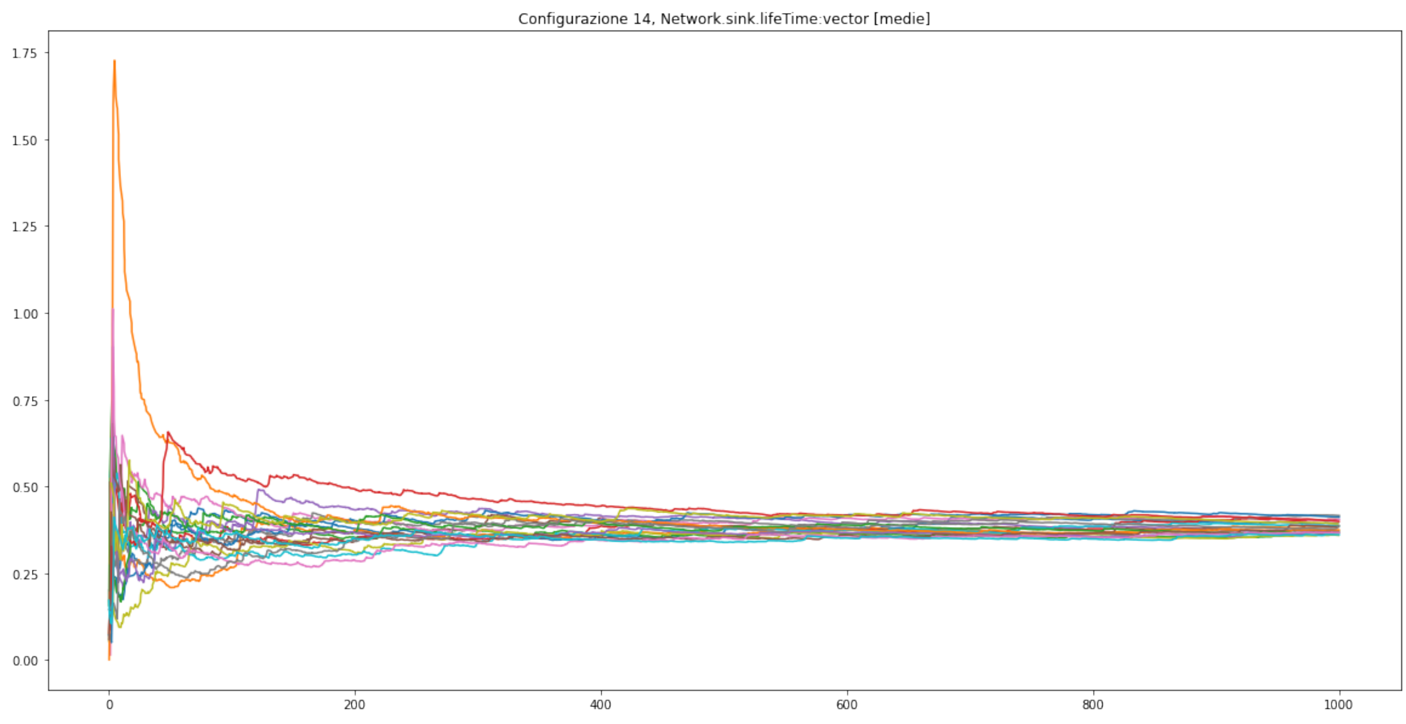


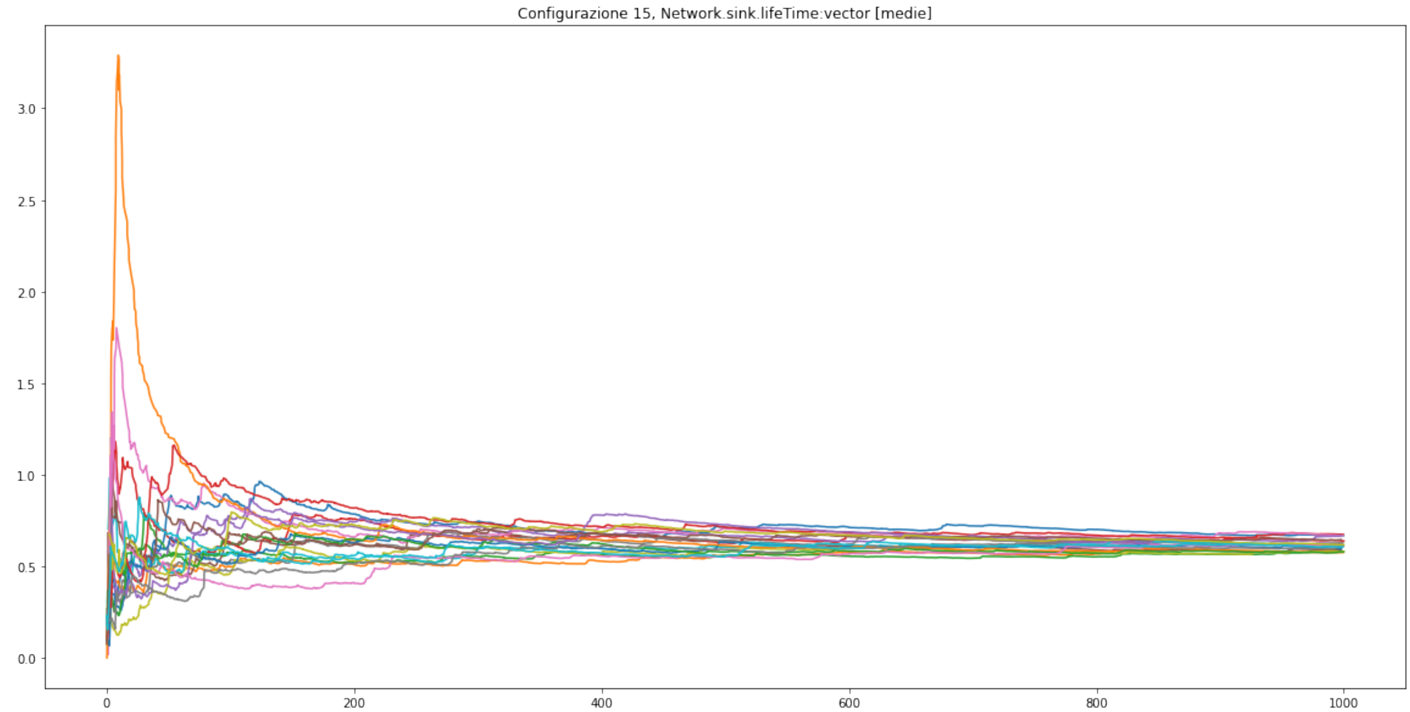
Configurazione 11, Network.sink.lifeTime:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

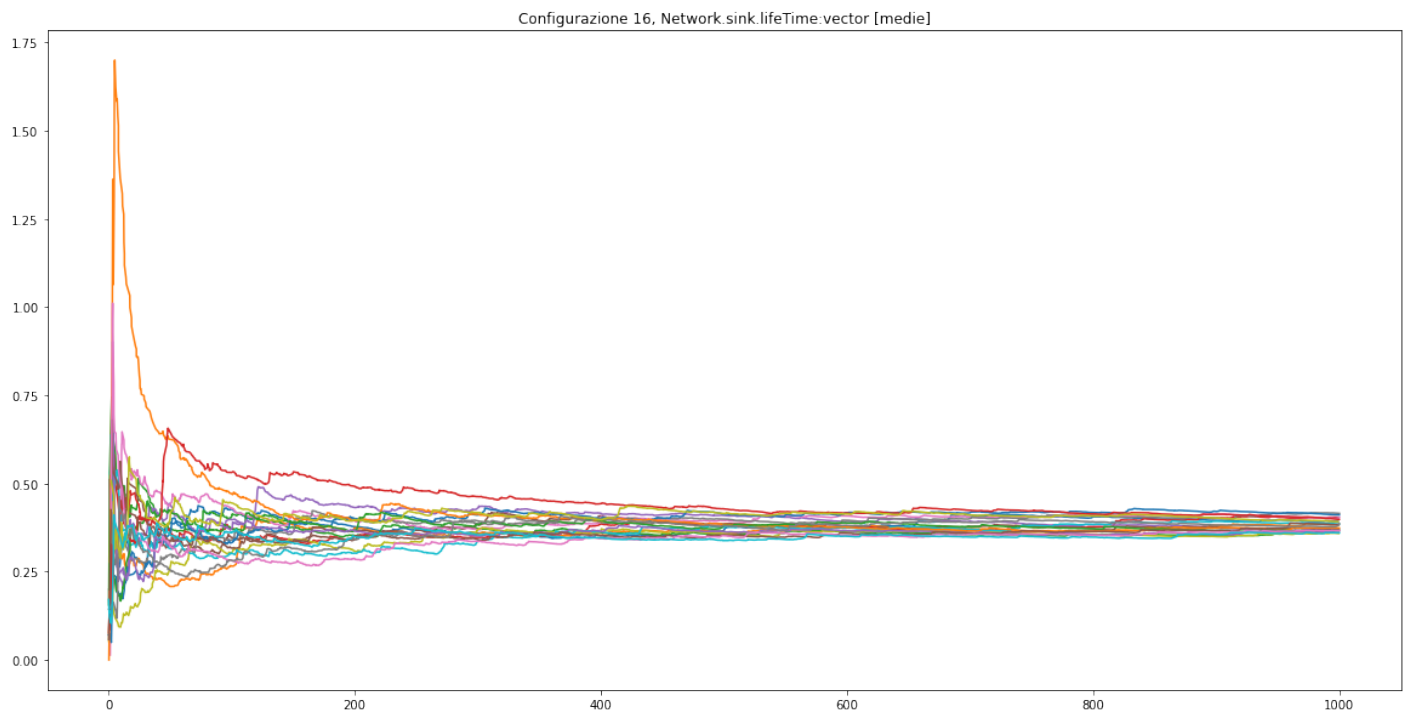


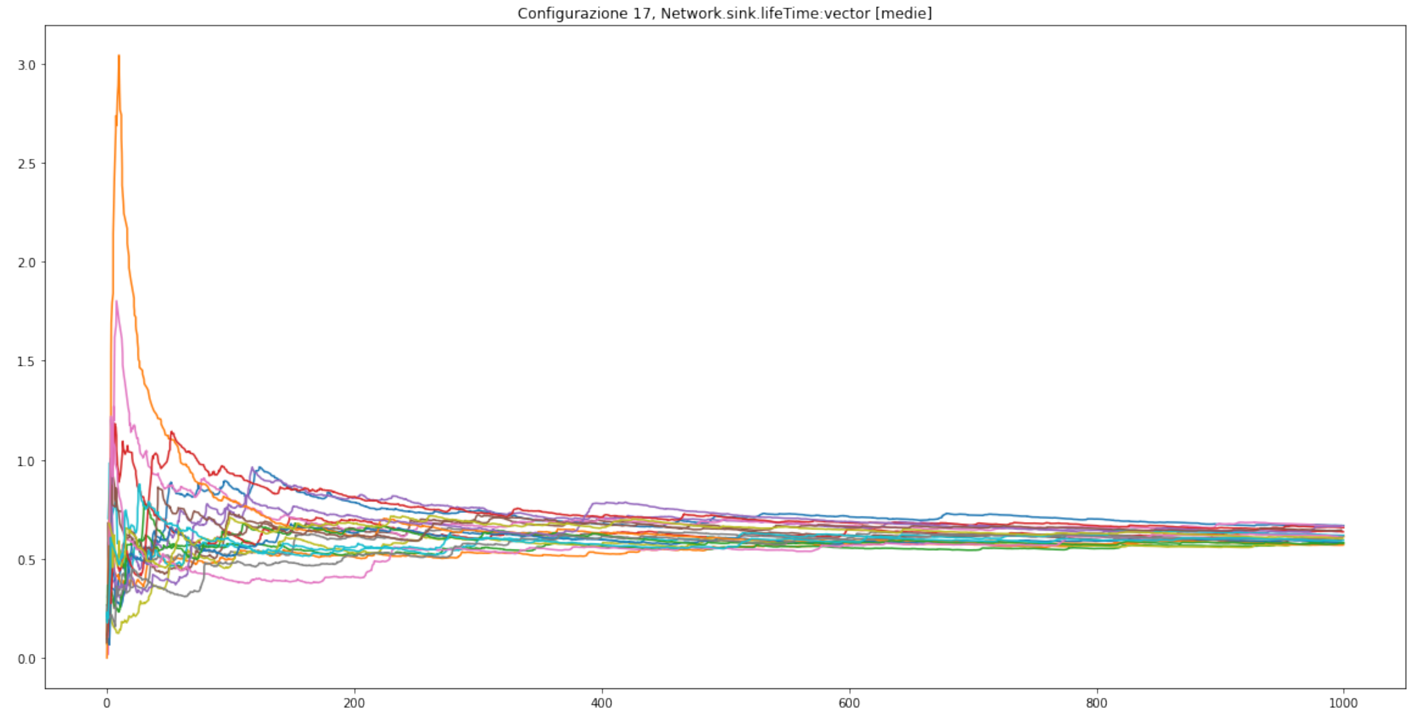
Configurazione 12, Network.sink.lifeTime:vector [medie]
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

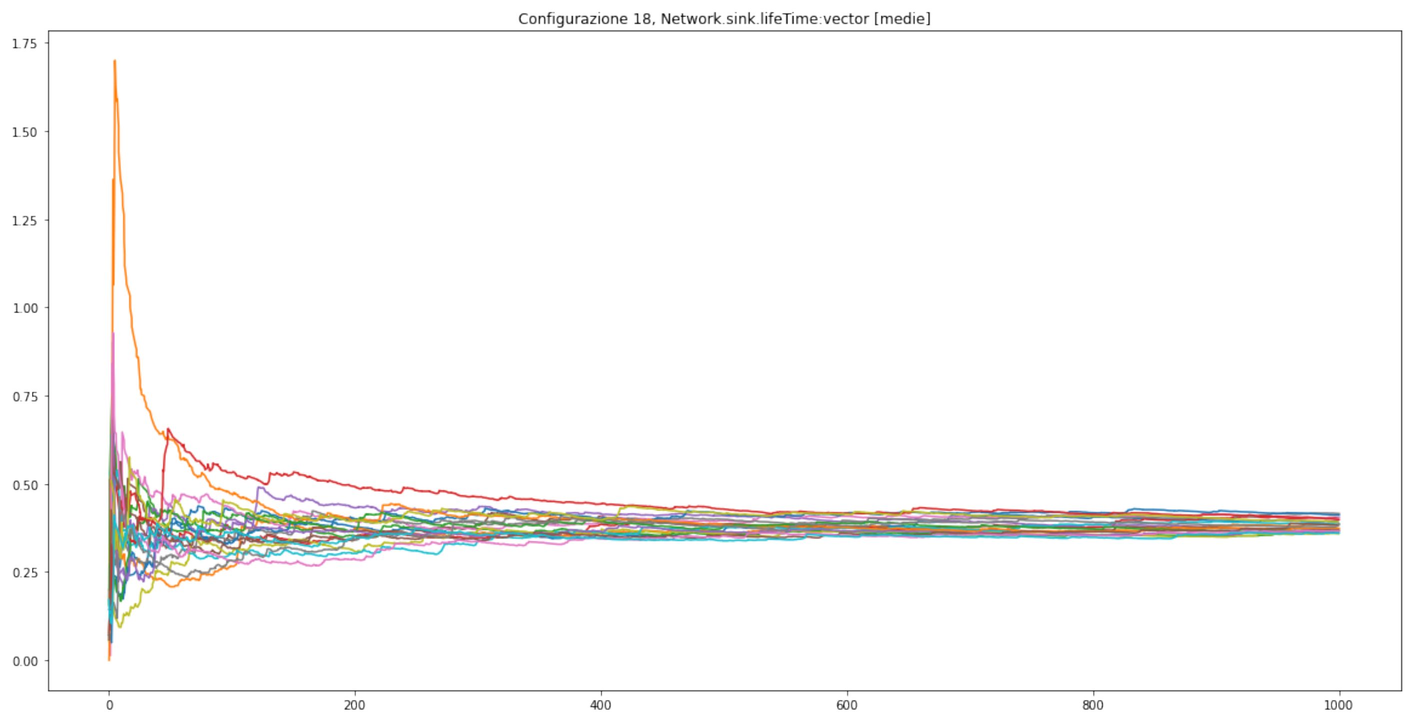

Configurazione 13, Network.sink.lifeTime:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

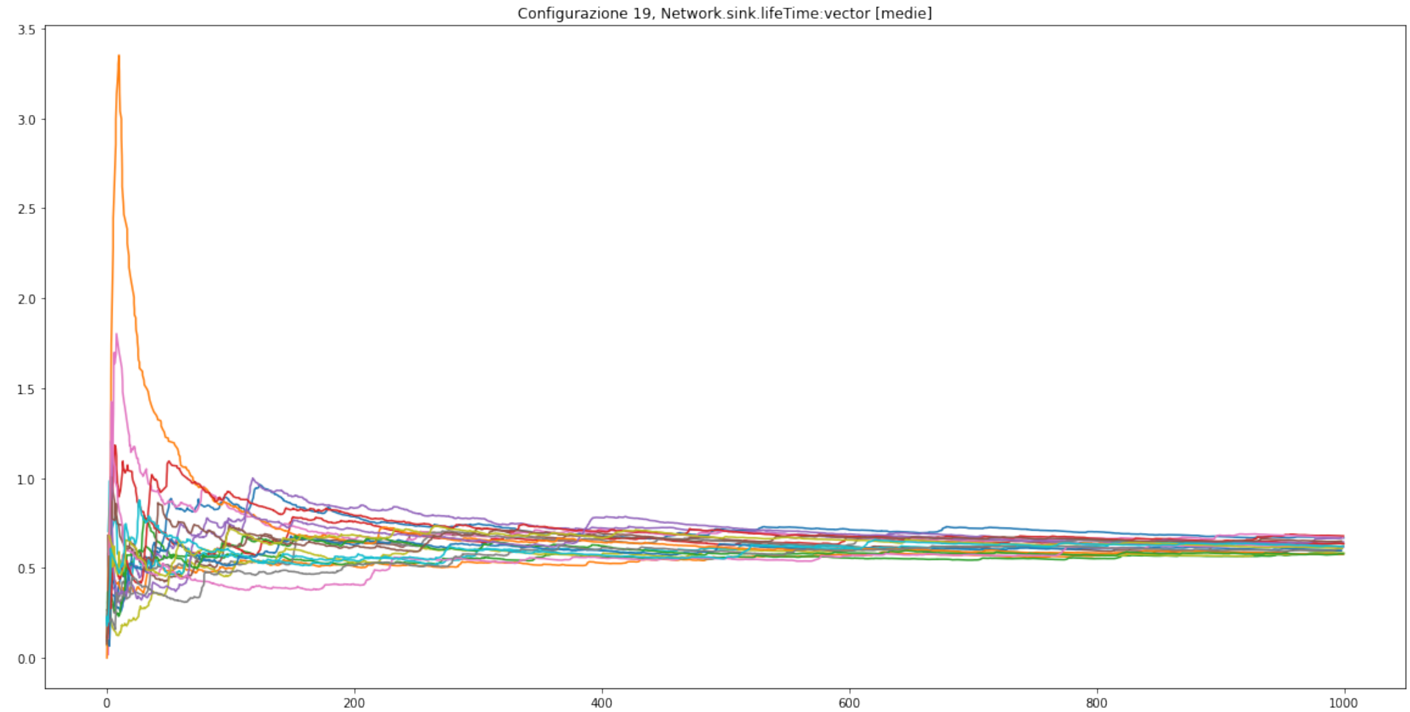

Configurazione 14, Network.sink.lifeTime:vector [medie]
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

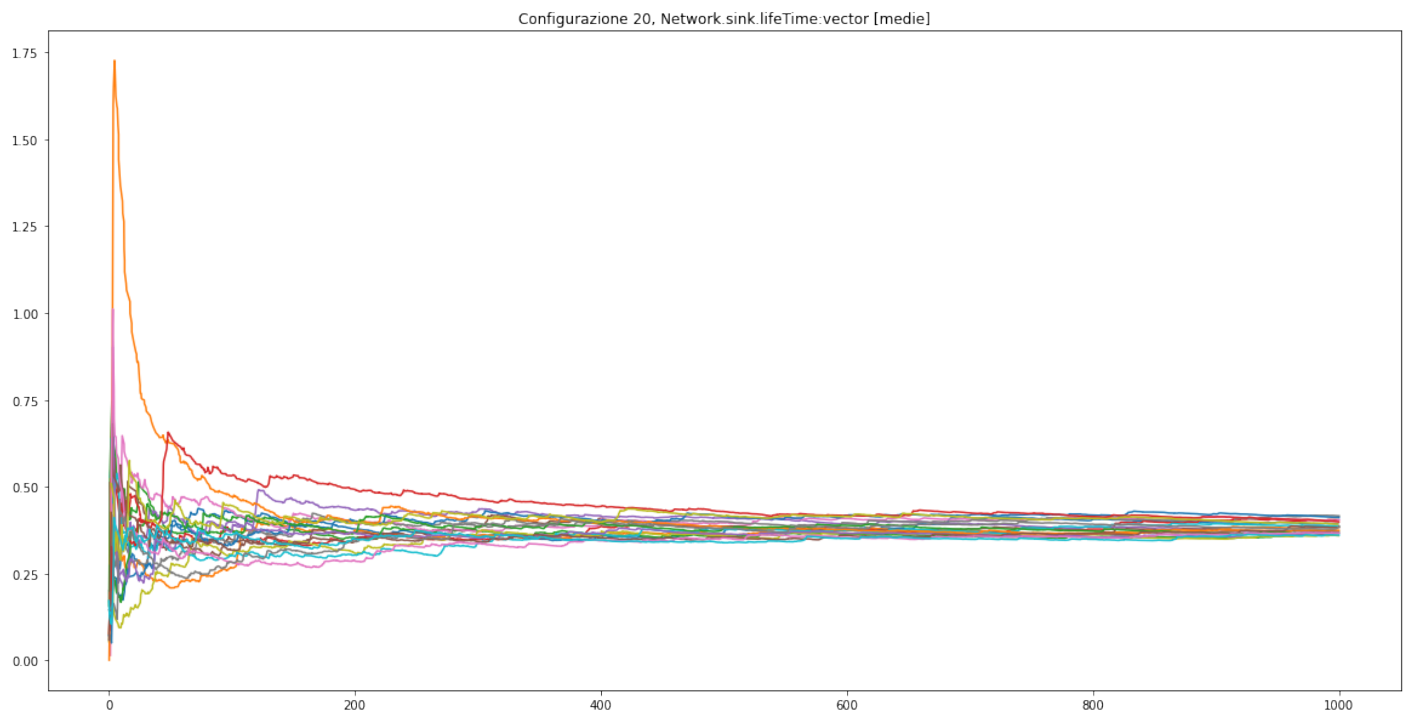

Configurazione 15, Network.sink.lifeTime:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

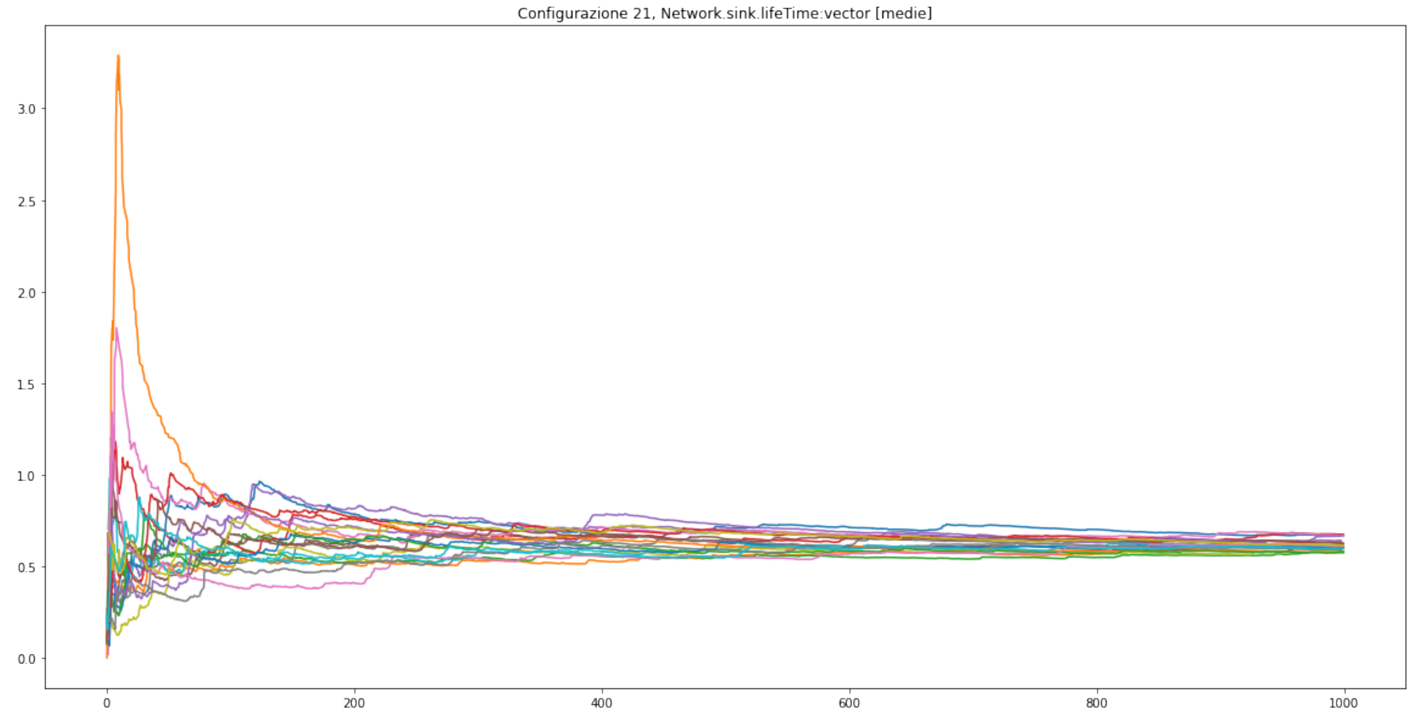

Configurazione 16, Network.sink.lifeTime:vector [medie]
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

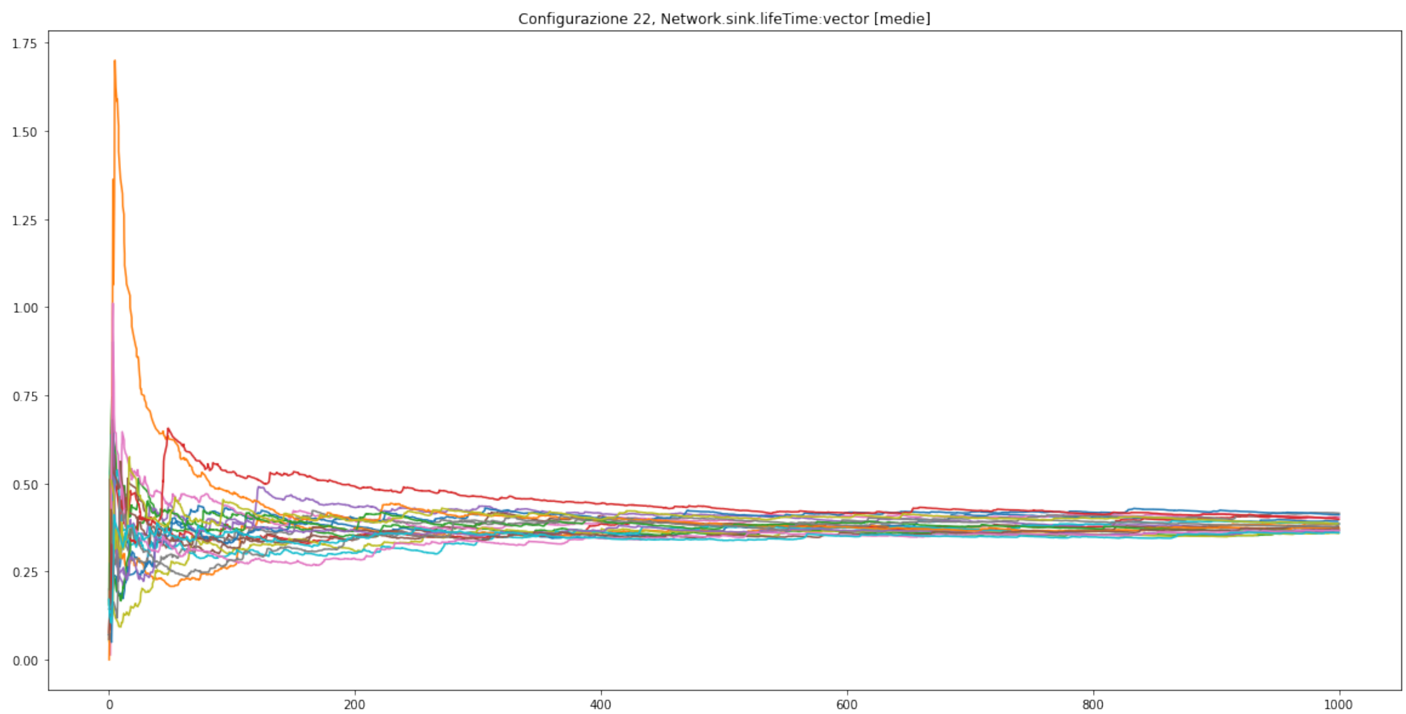

Configurazione 17, Network.sink.lifeTime:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

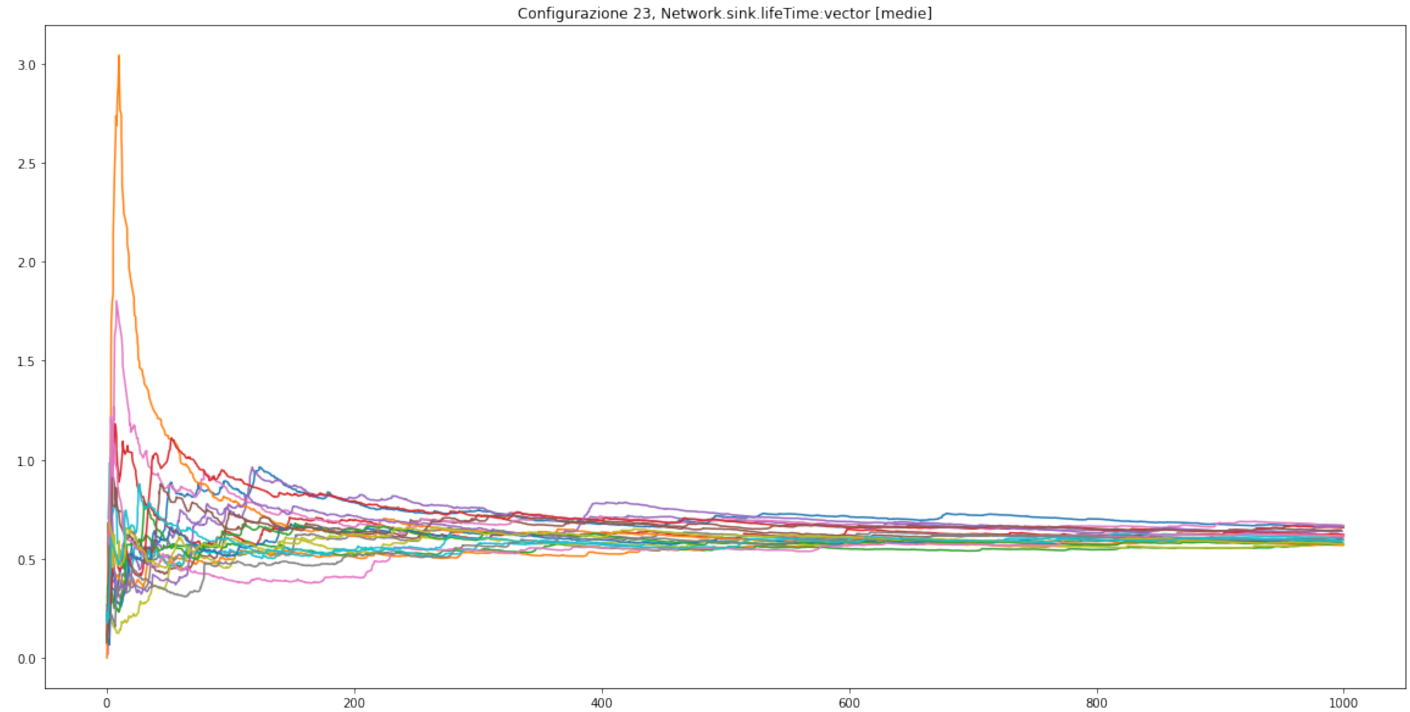

Configurazione 18, Network.sink.lifeTime:vector [medie]
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

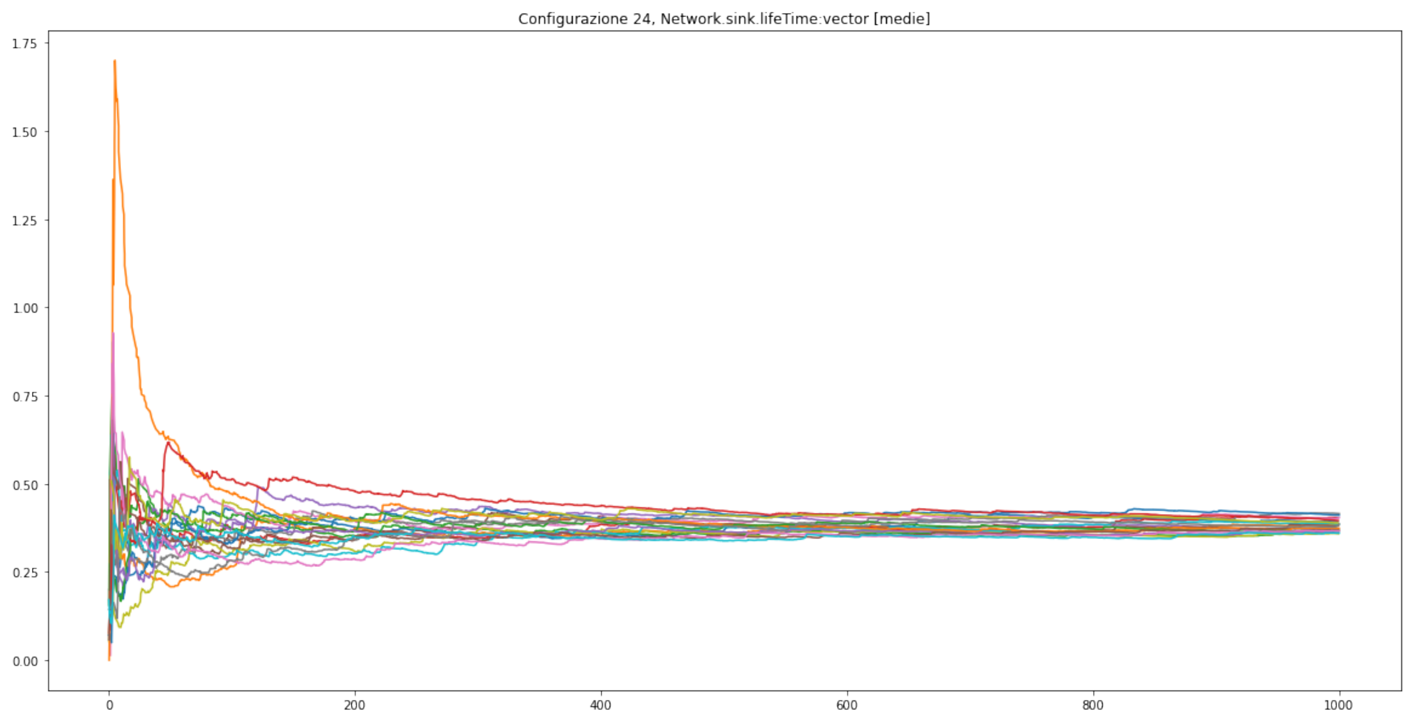

Configurazione 19, Network.sink.lifeTime:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

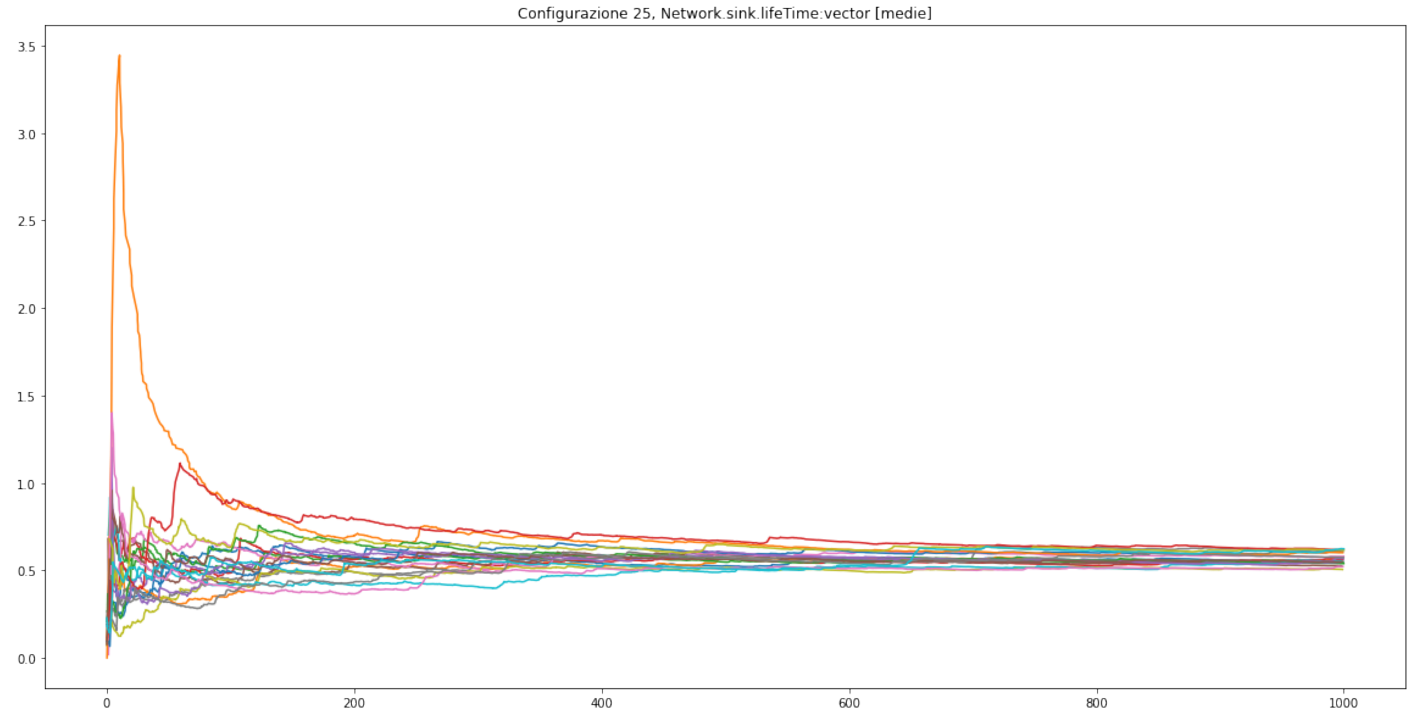

Configurazione 20, Network.sink.lifeTime:vector [medie]
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

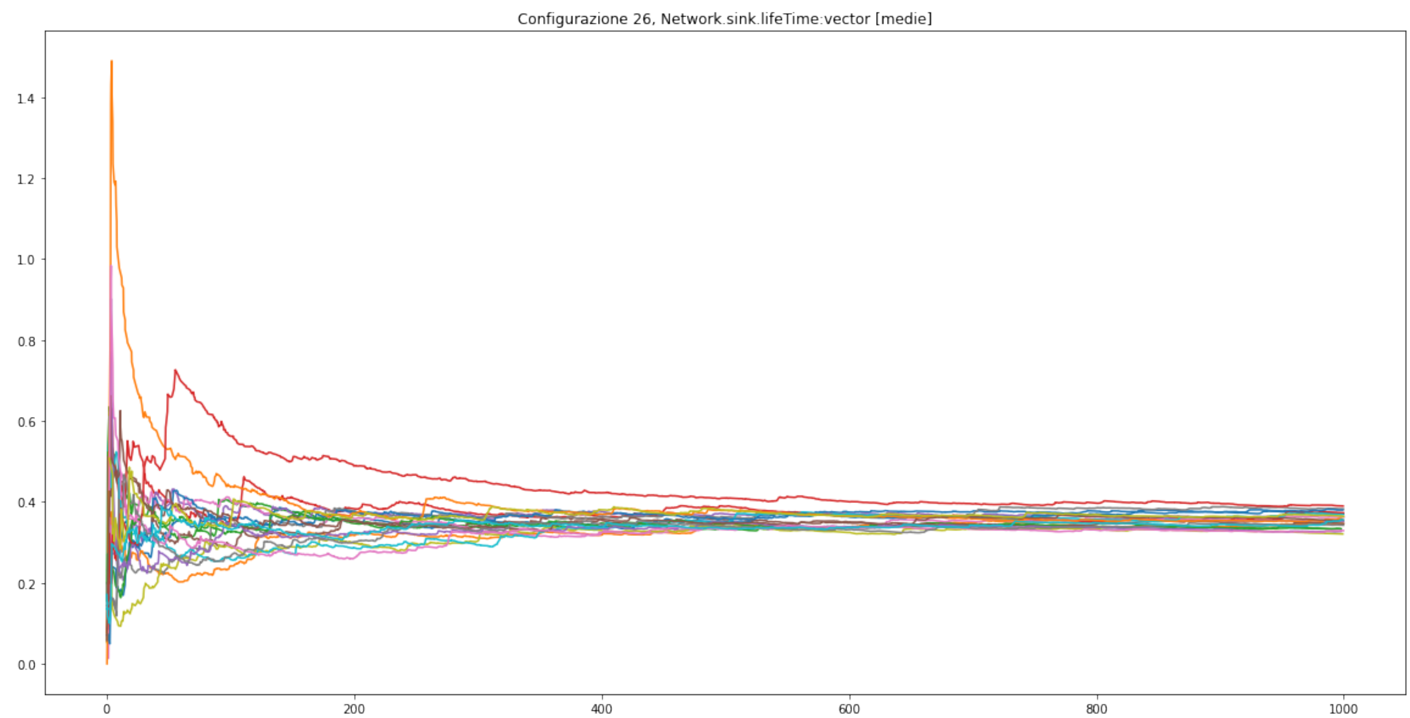

Configurazione 21, Network.sink.lifeTime:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

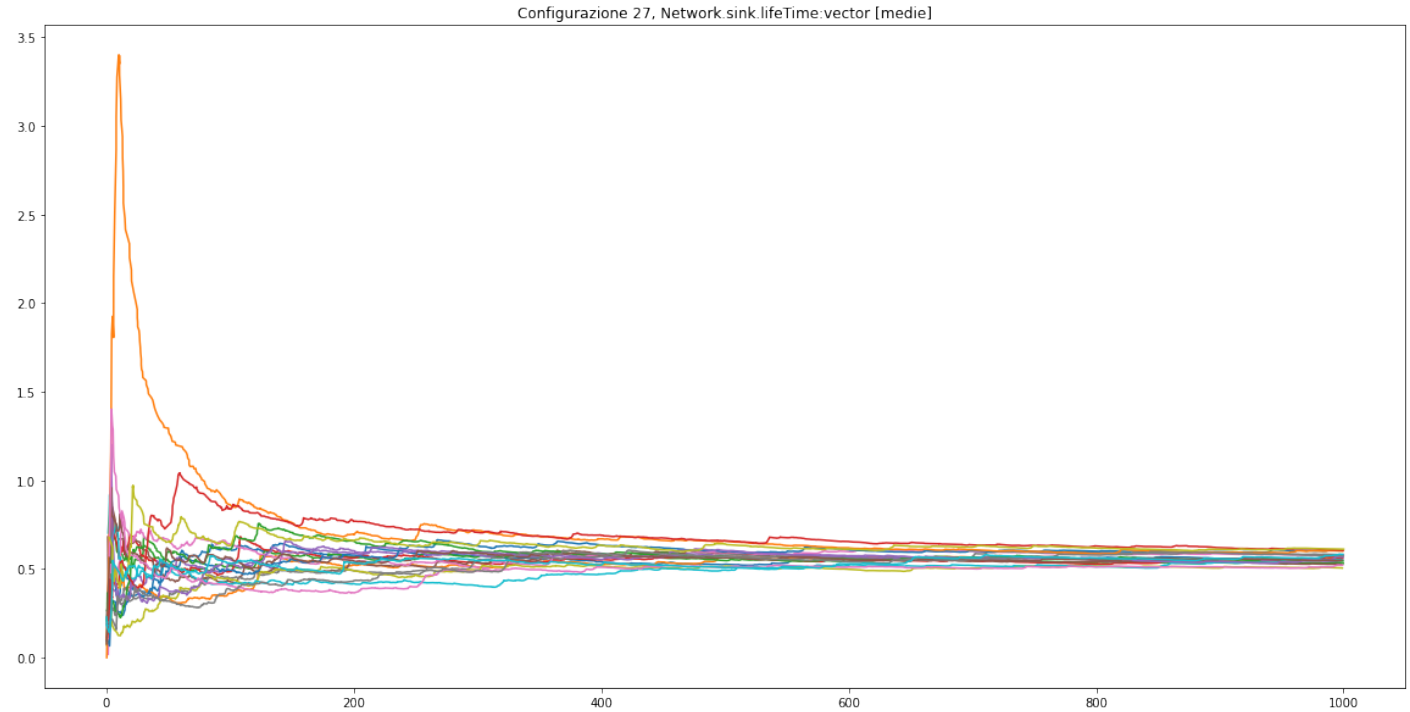

Configurazione 22, Network.sink.lifeTime:vector [medie]
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

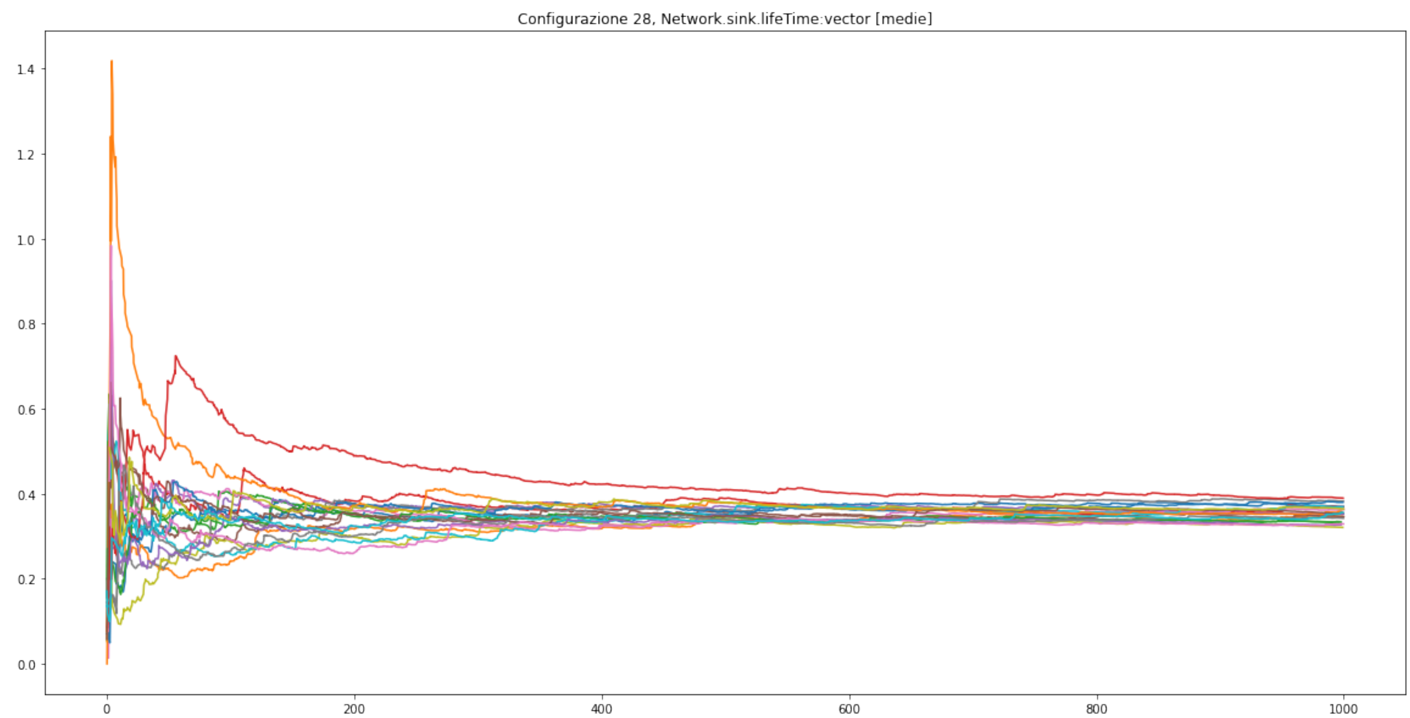

Configurazione 23, Network.sink.lifeTime:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

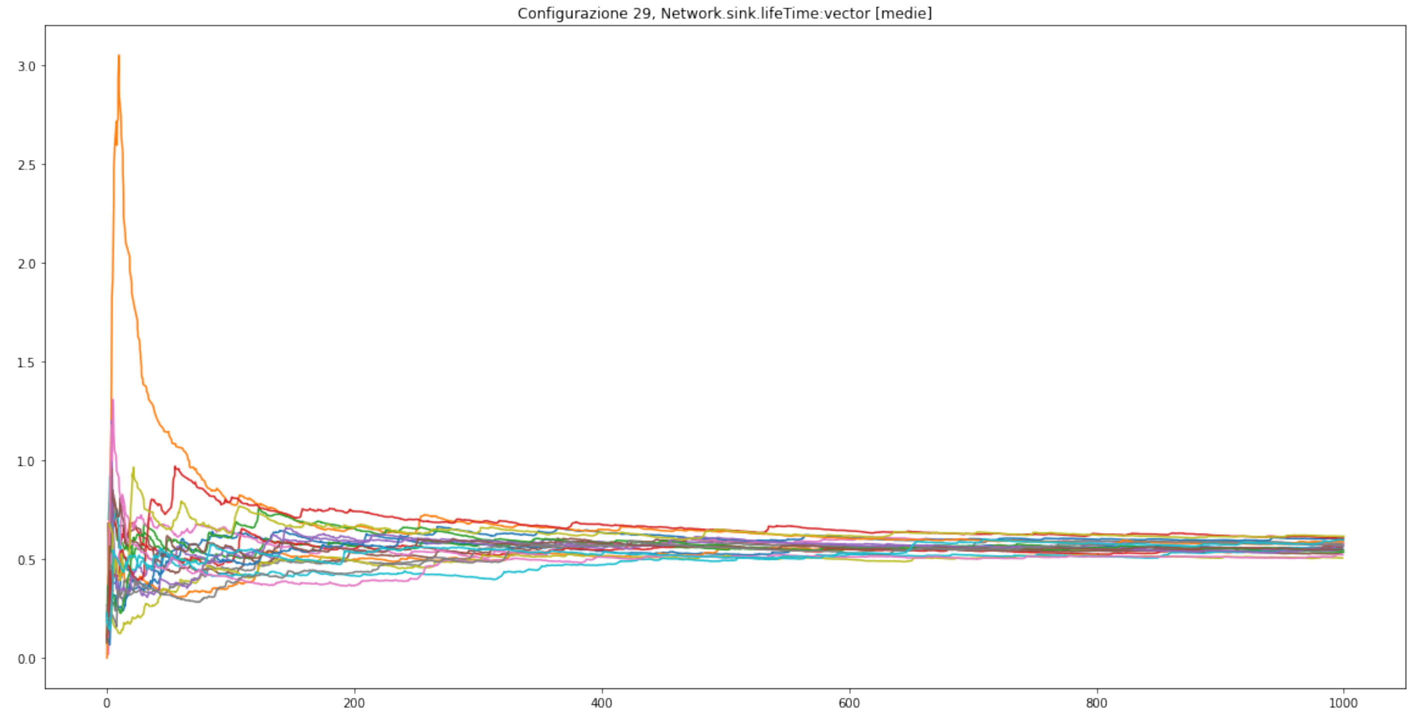

Configurazione 24, Network.sink.lifeTime:vector [medie]
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

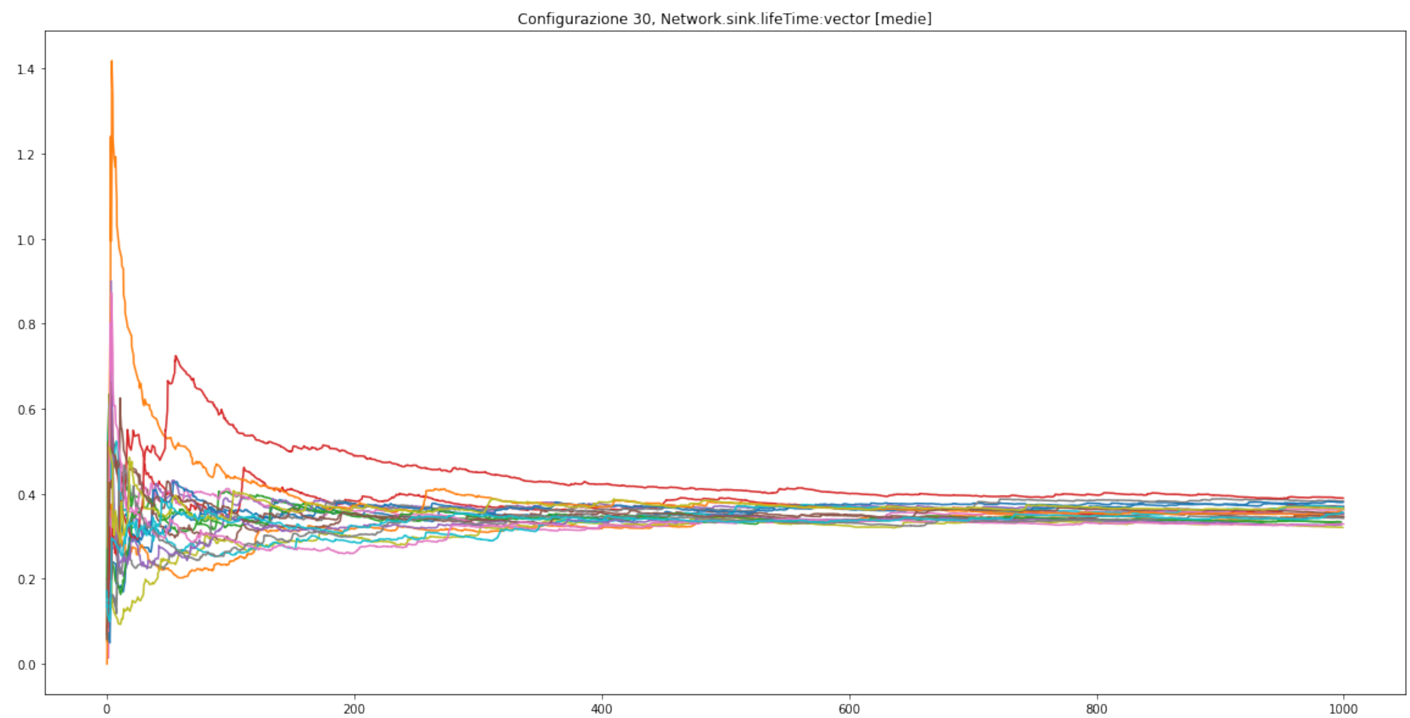

Configurazione 25, Network.sink.lifeTime:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

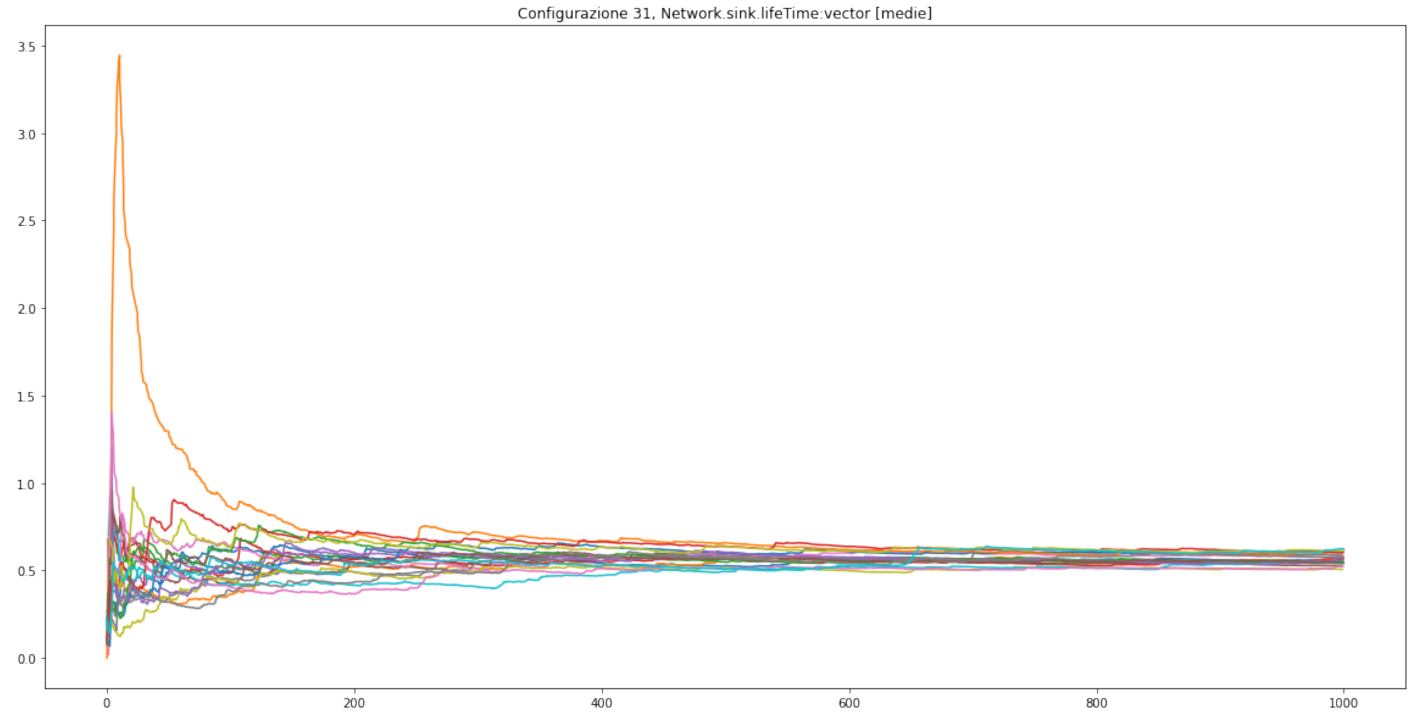

Configurazione 26, Network.sink.lifeTime:vector [medie]
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

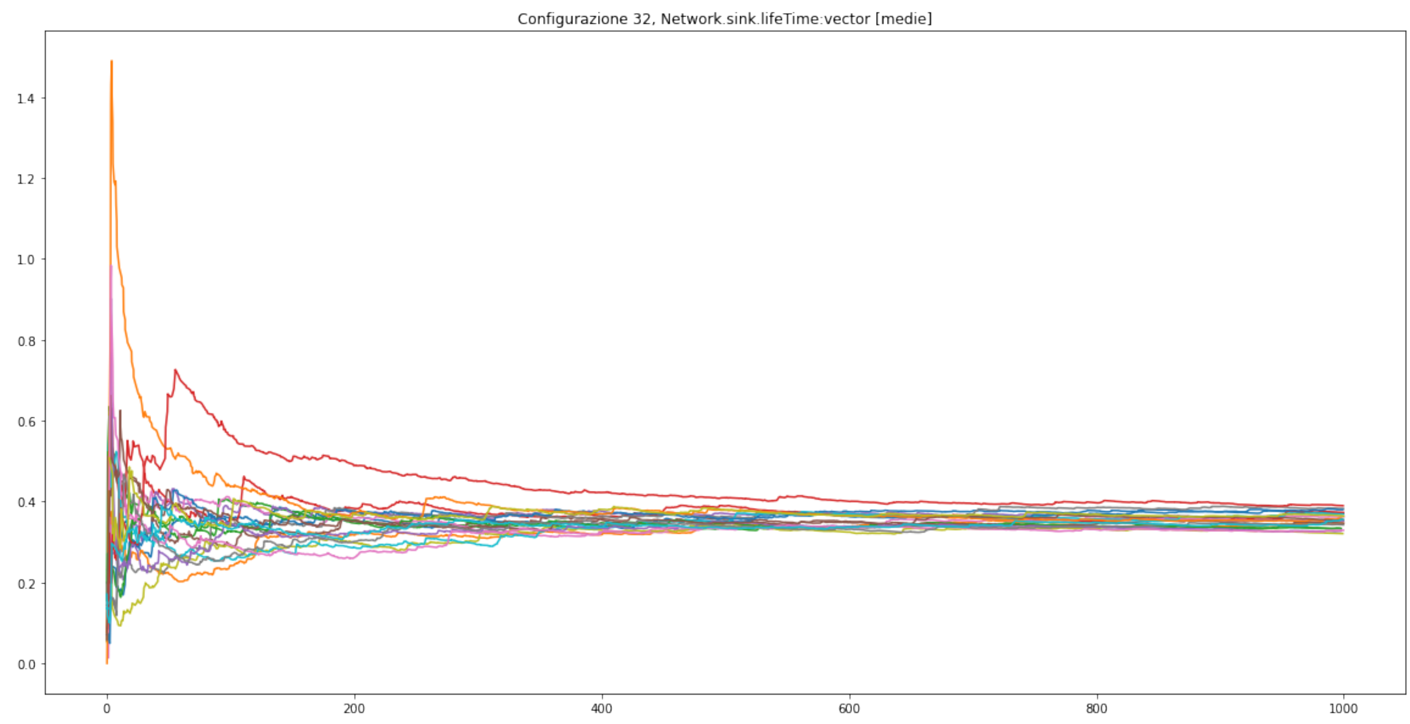

Configurazione 27, Network.sink.lifeTime:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

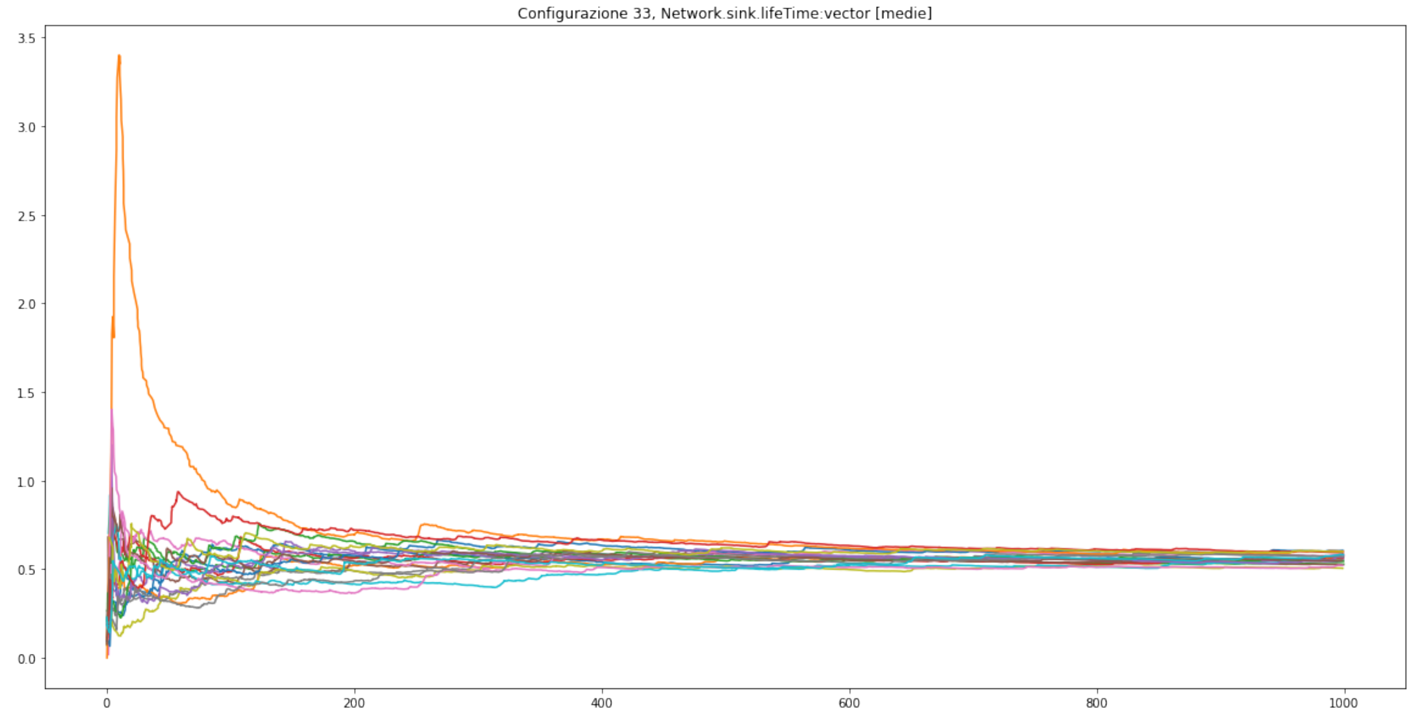

Configurazione 28, Network.sink.lifeTime:vector [medie]
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

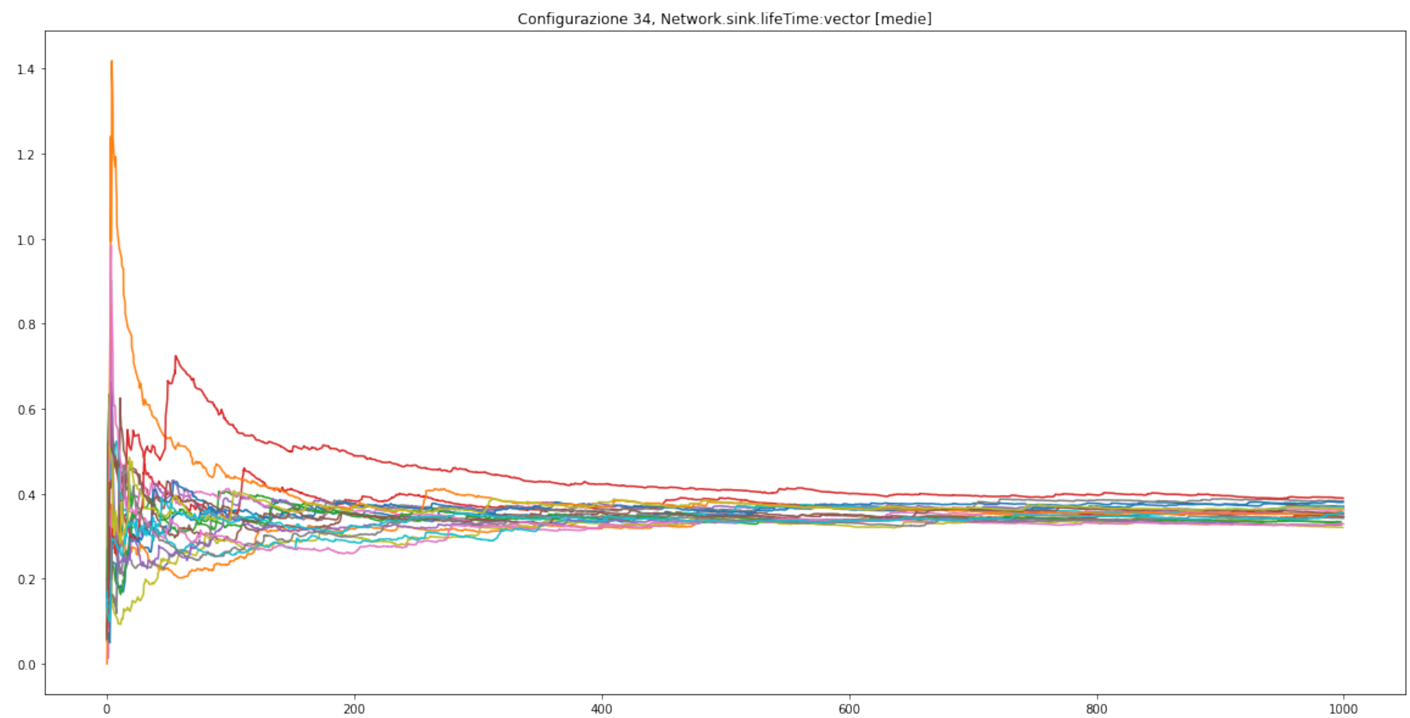

Configurazione 29, Network.sink.lifeTime:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

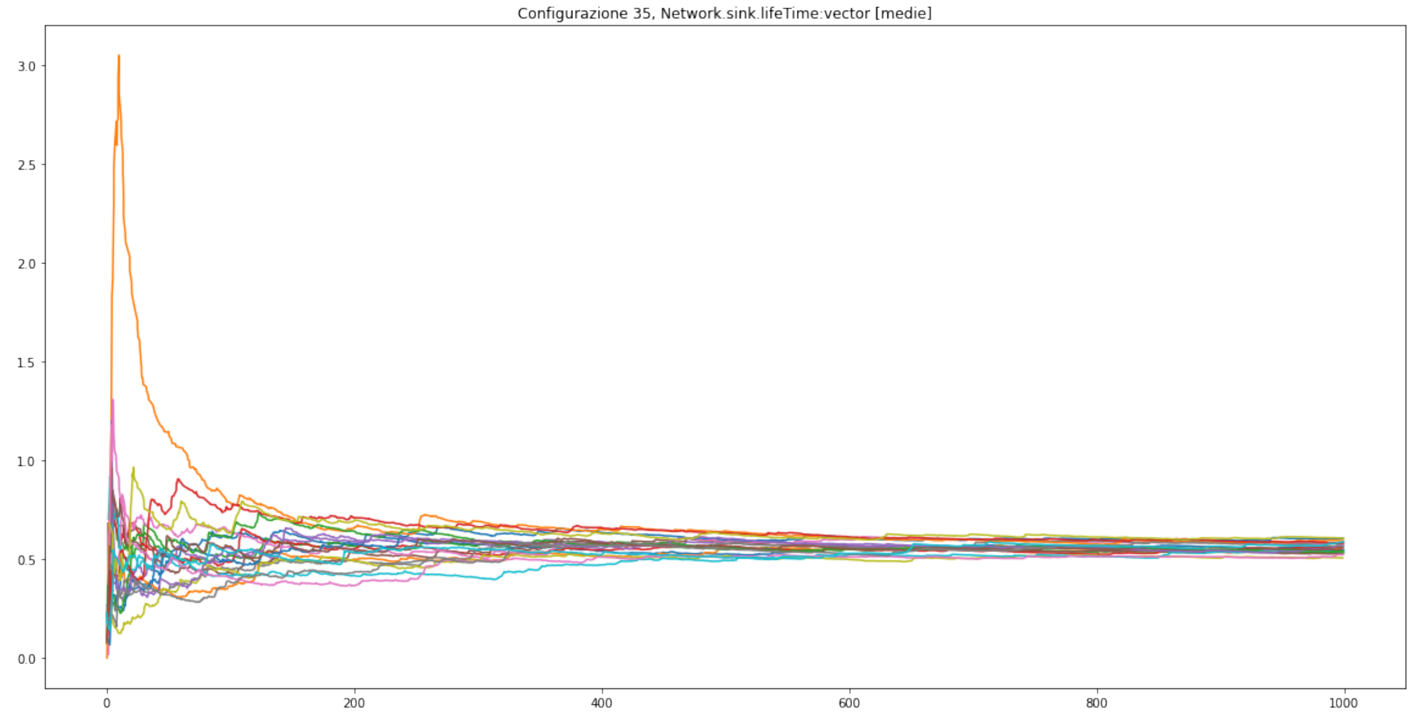

Configurazione 30, Network.sink.lifeTime:vector [medie]
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

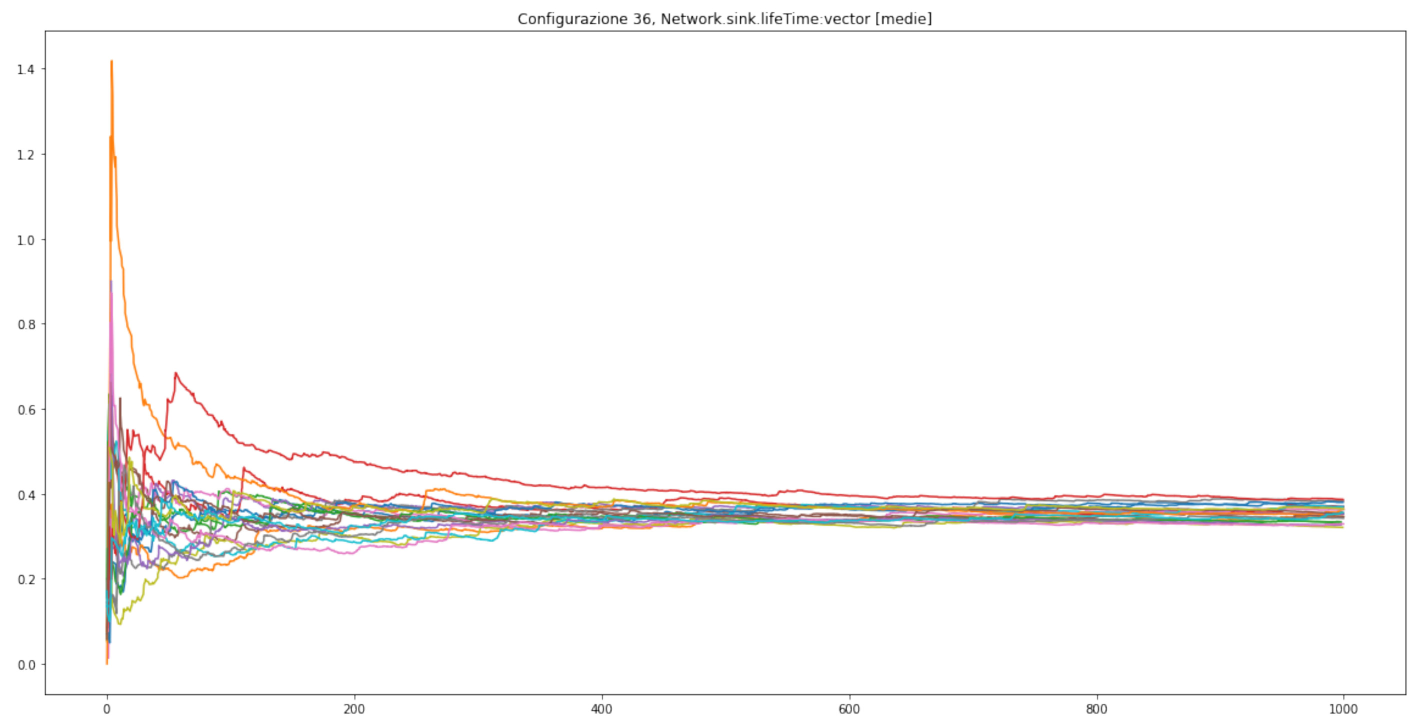

Configurazione 31, Network.sink.lifeTime:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

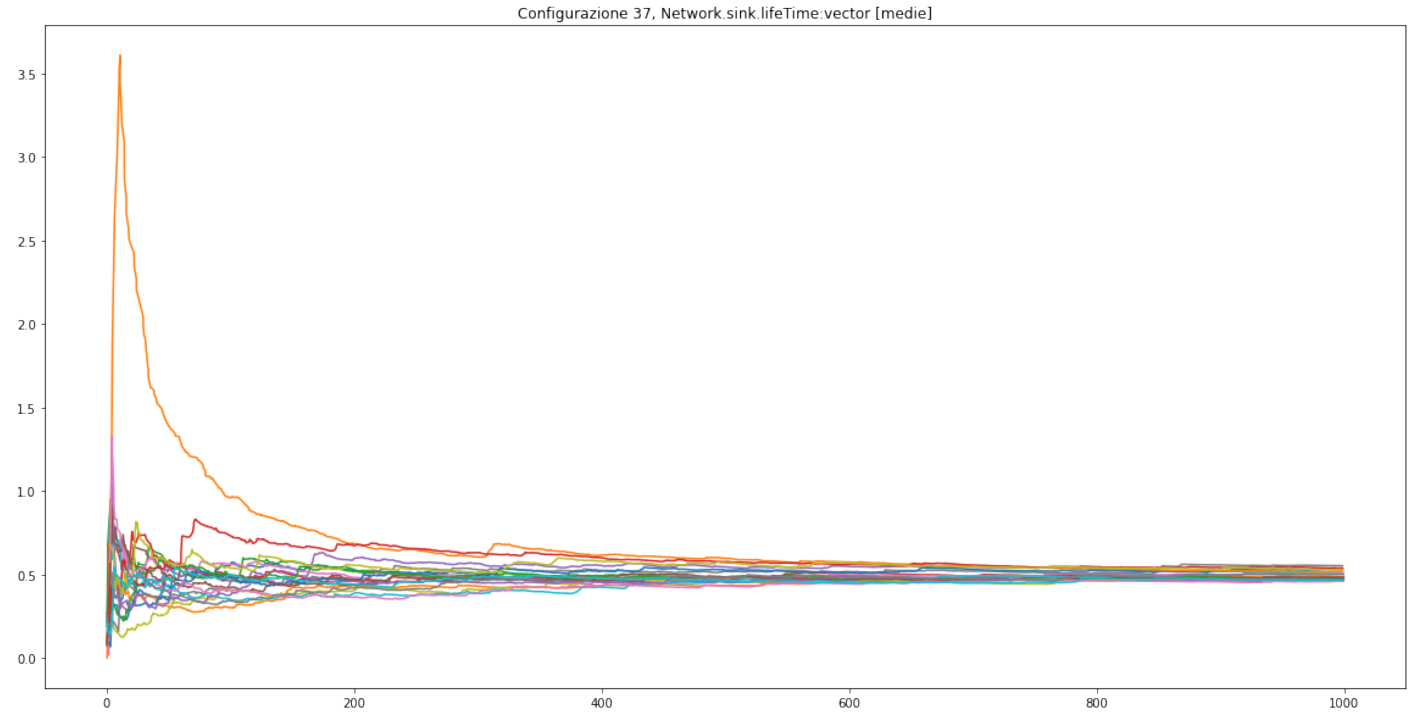

Configurazione 32, Network.sink.lifeTime:vector [medie]
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

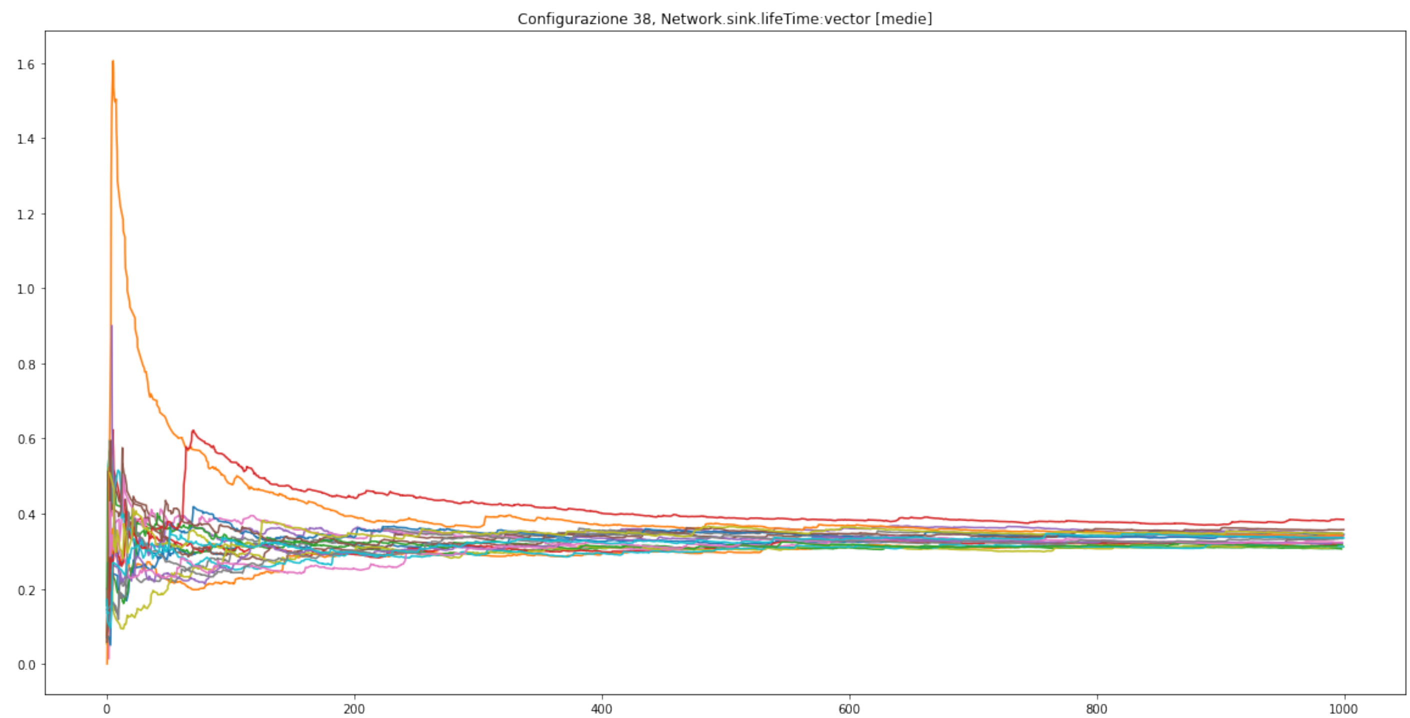

Configurazione 33, Network.sink.lifeTime:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

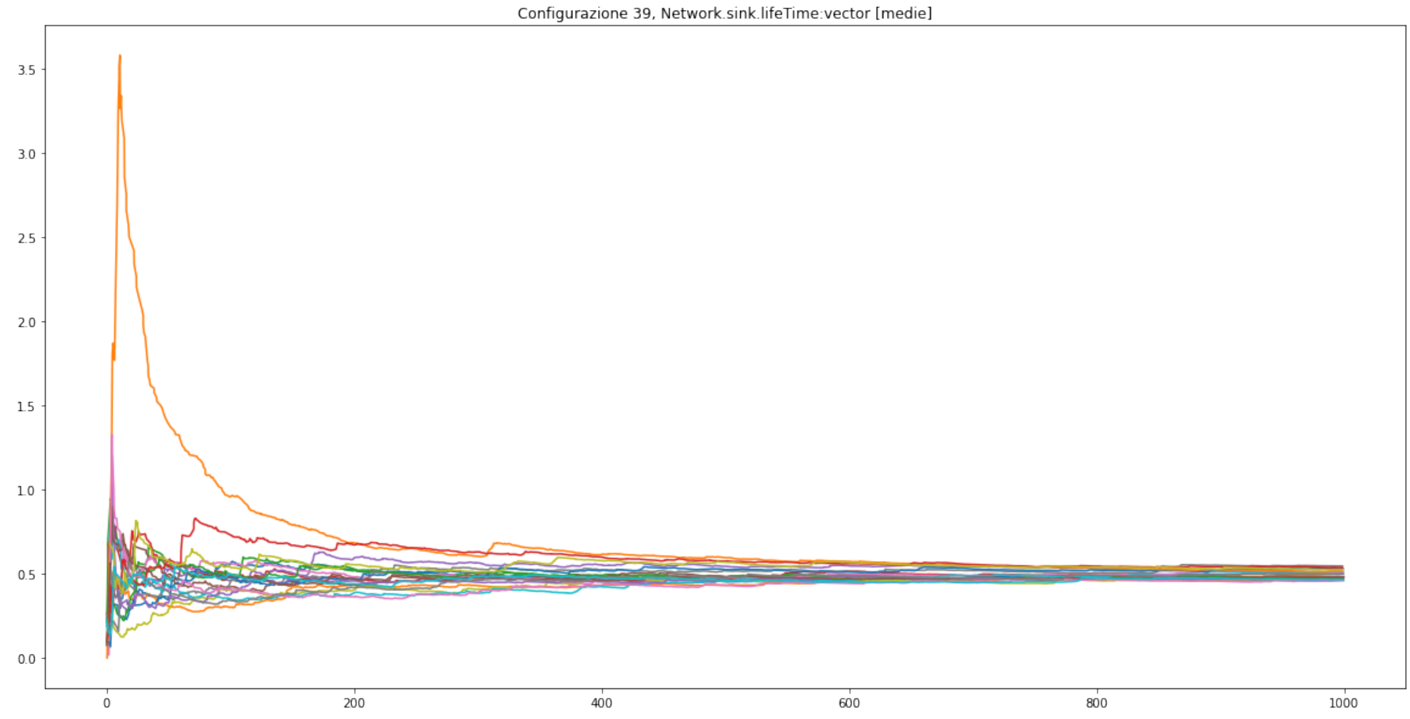

Configurazione 34, Network.sink.lifeTime:vector [medie]
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

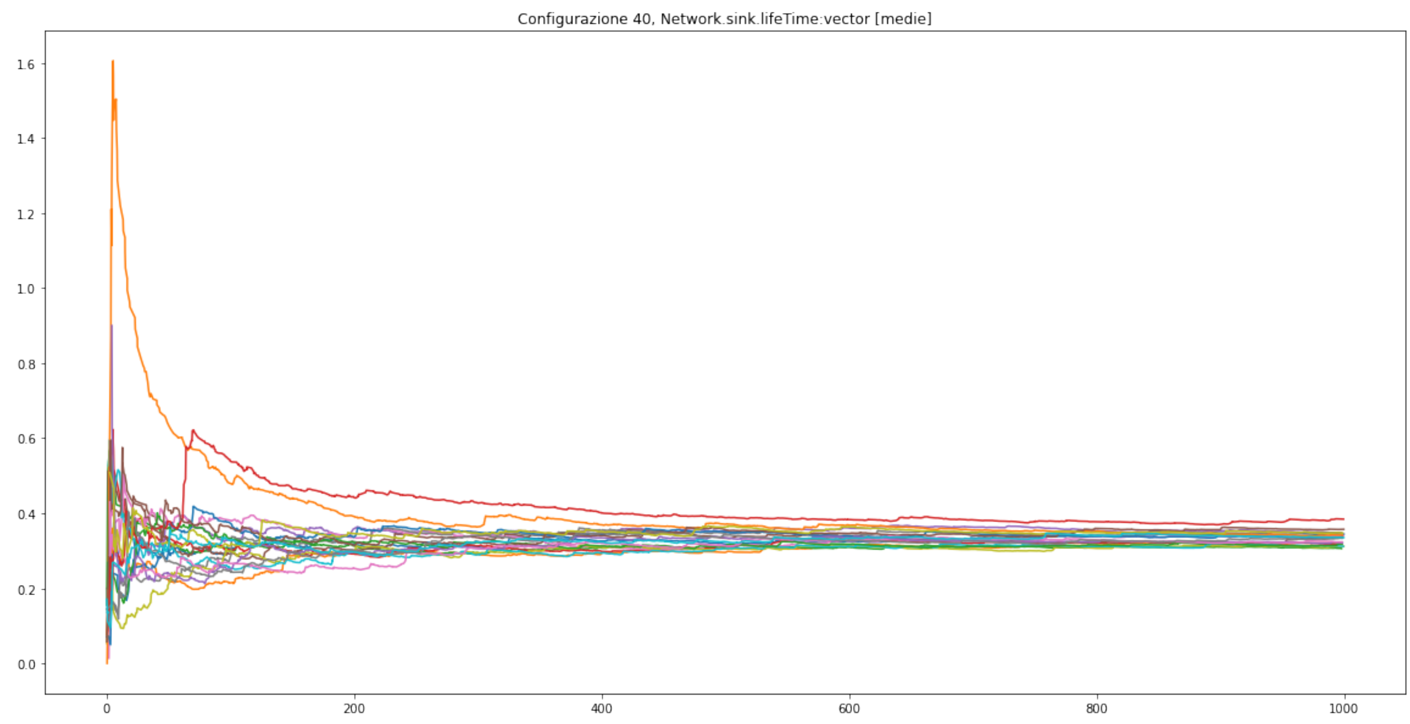

Configurazione 35, Network.sink.lifeTime:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

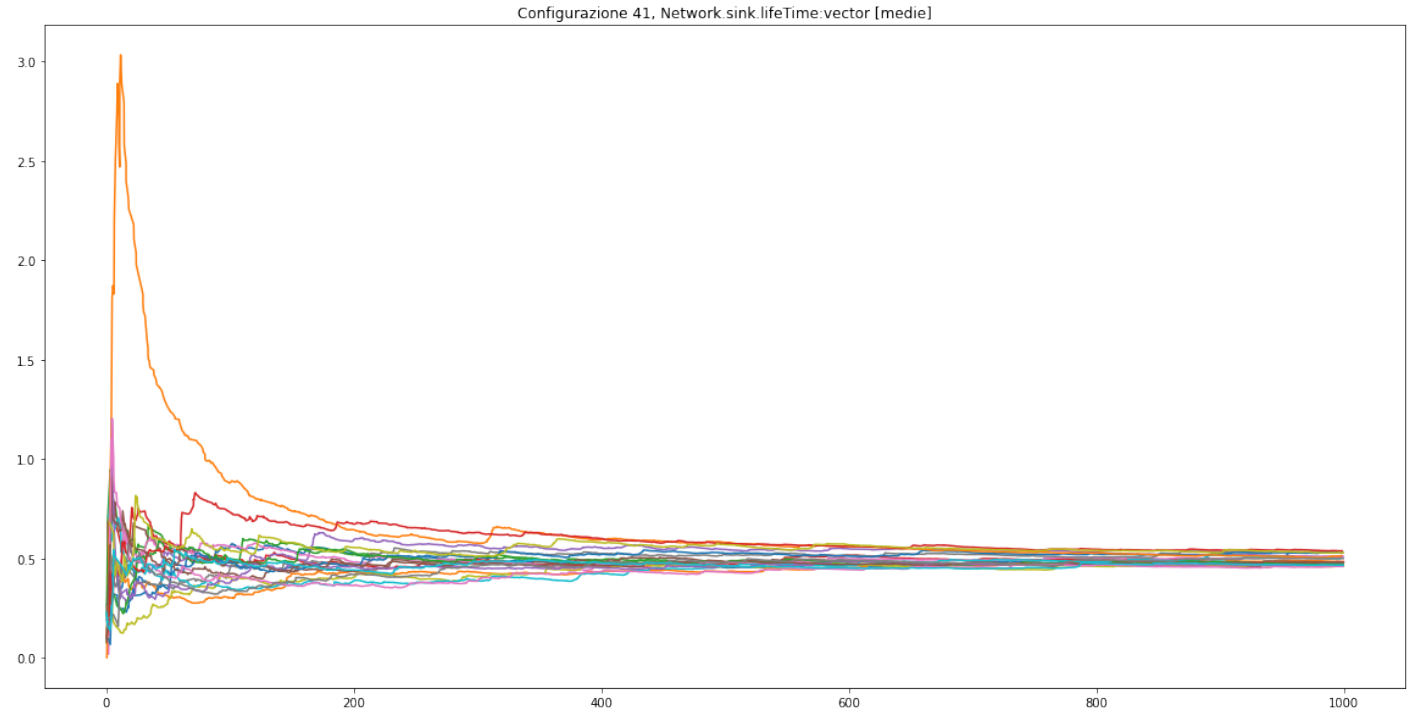

Configurazione 36, Network.sink.lifeTime:vector [medie]
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

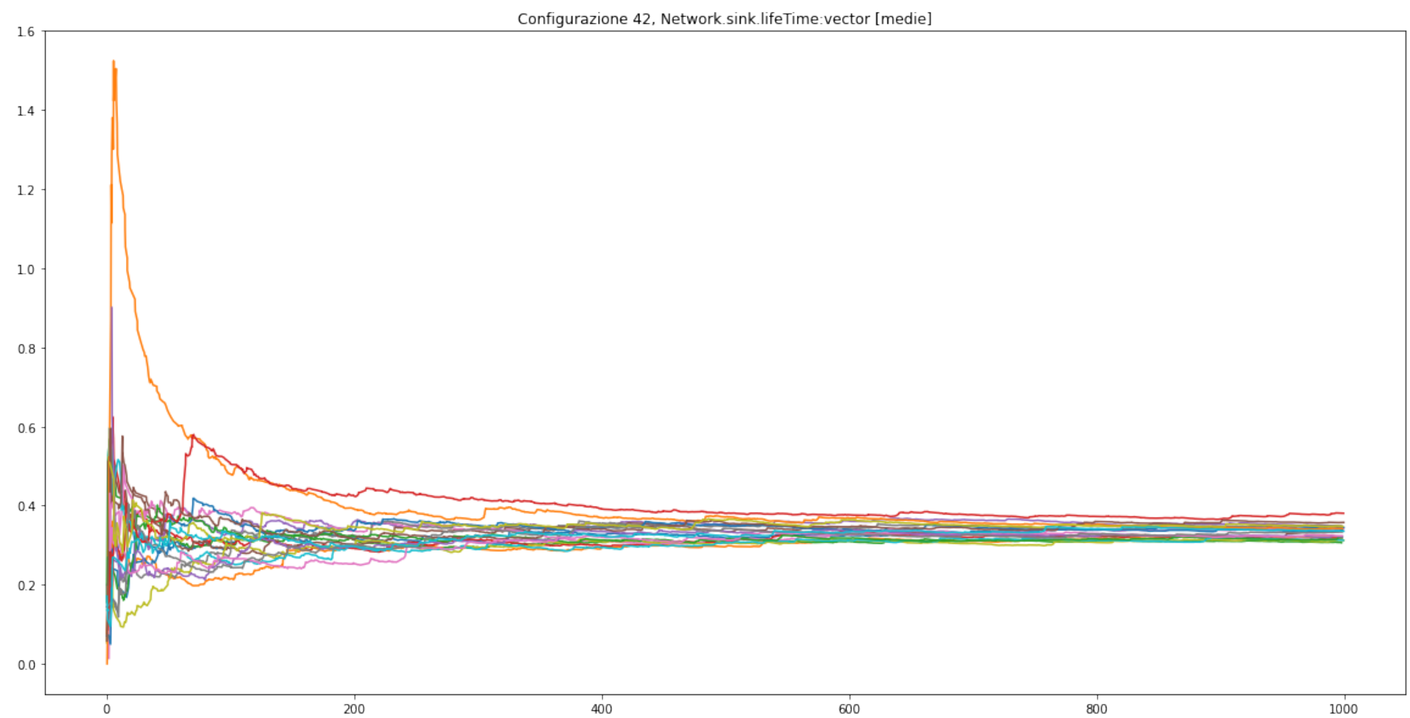

Configurazione 37, Network.sink.lifeTime:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

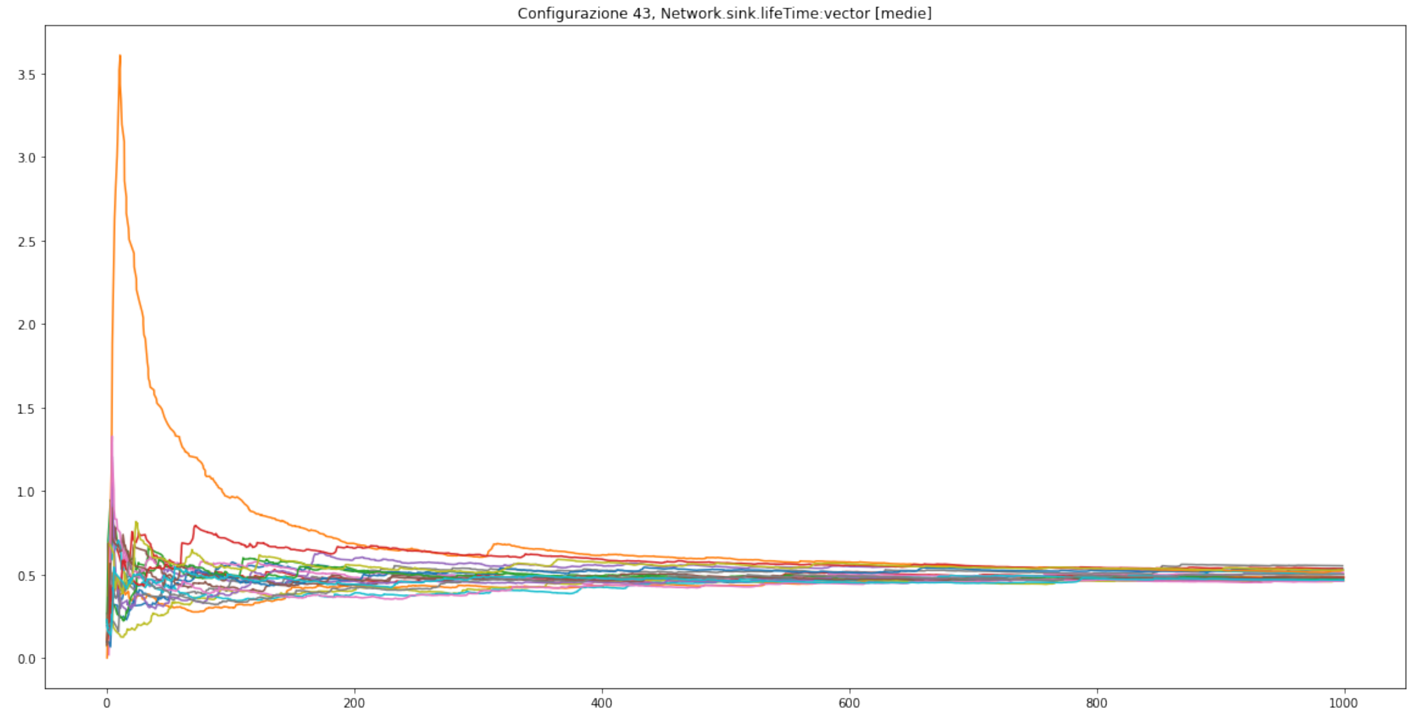

Configurazione 38, Network.sink.lifeTime:vector [medie]
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

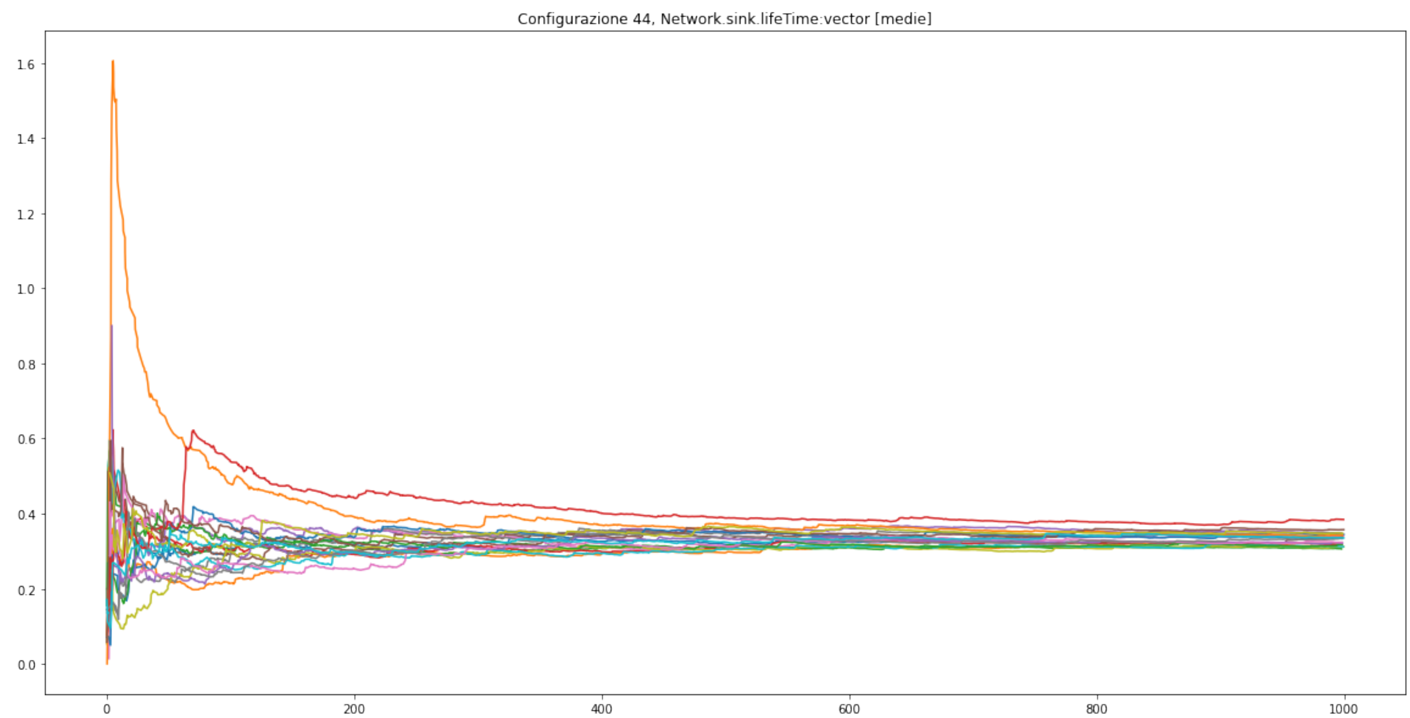

Configurazione 39, Network.sink.lifeTime:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

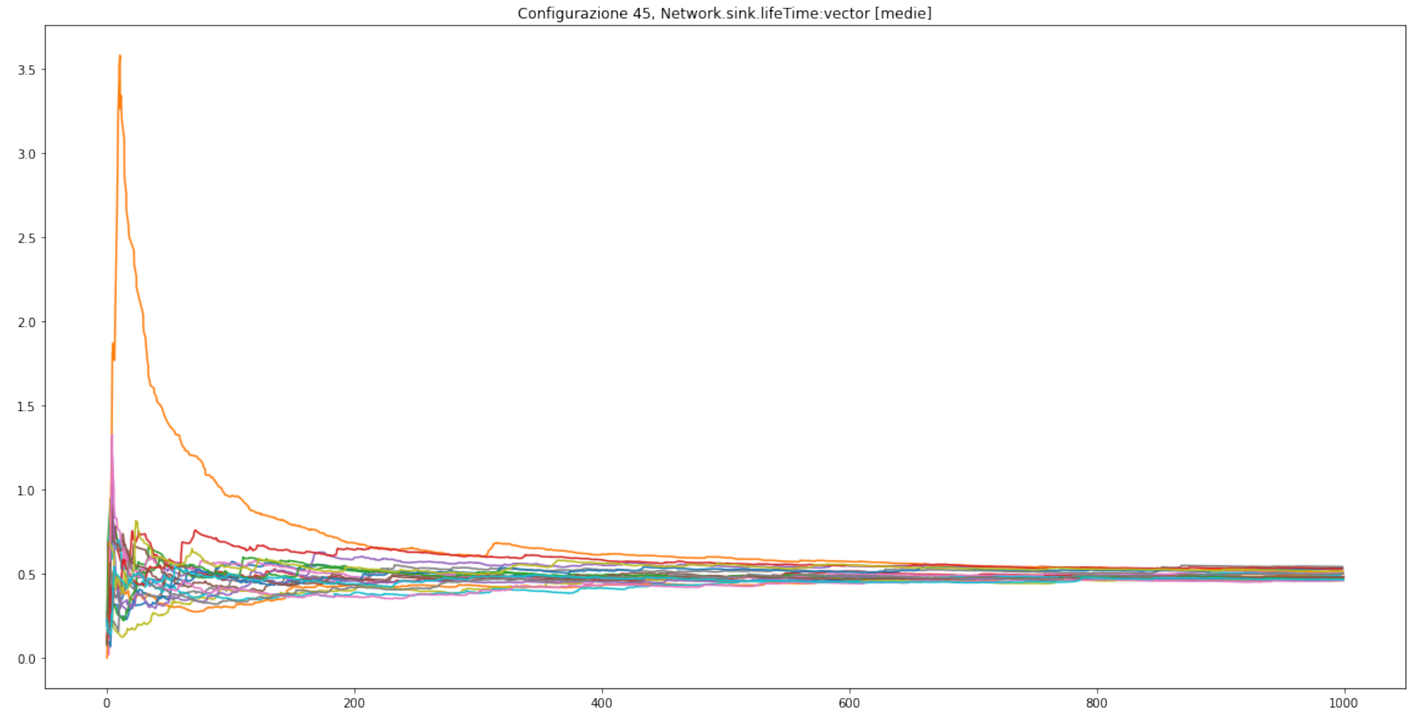

Configurazione 40, Network.sink.lifeTime:vector [medie]
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

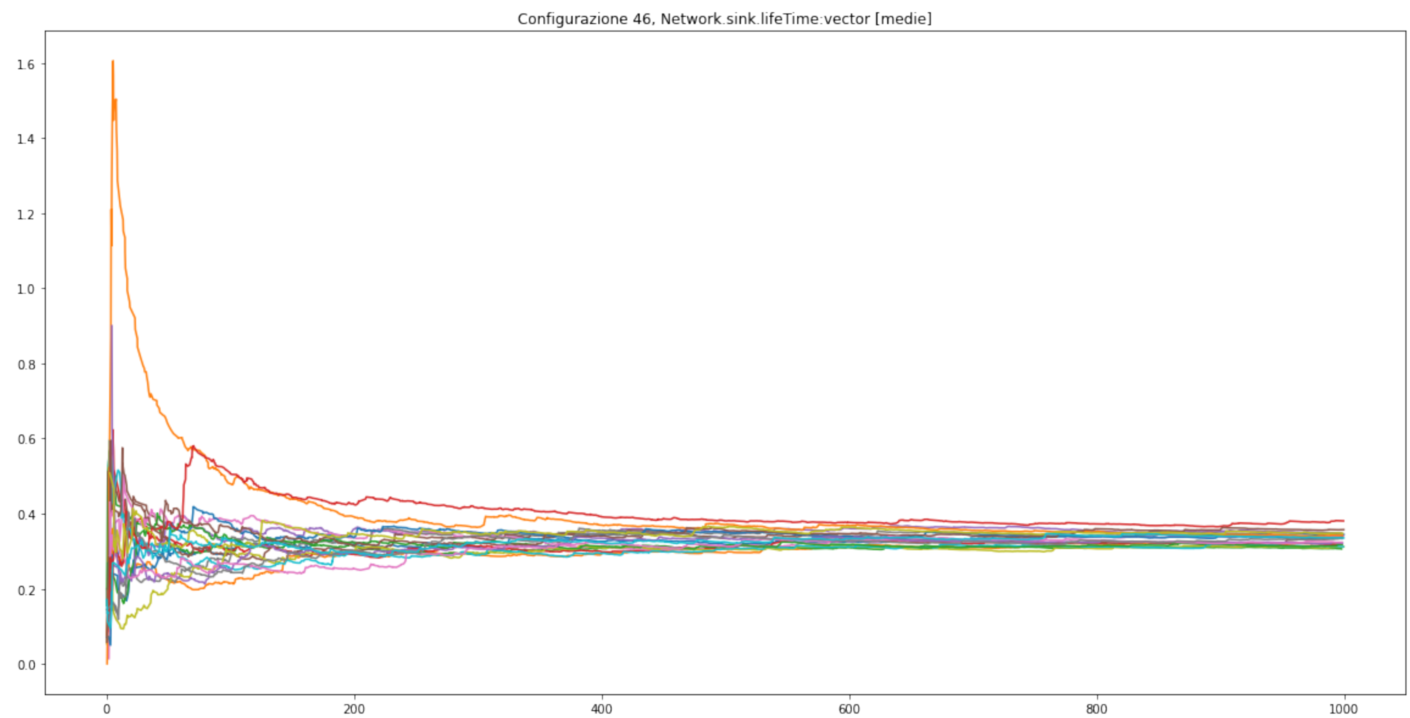

Configurazione 41, Network.sink.lifeTime:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

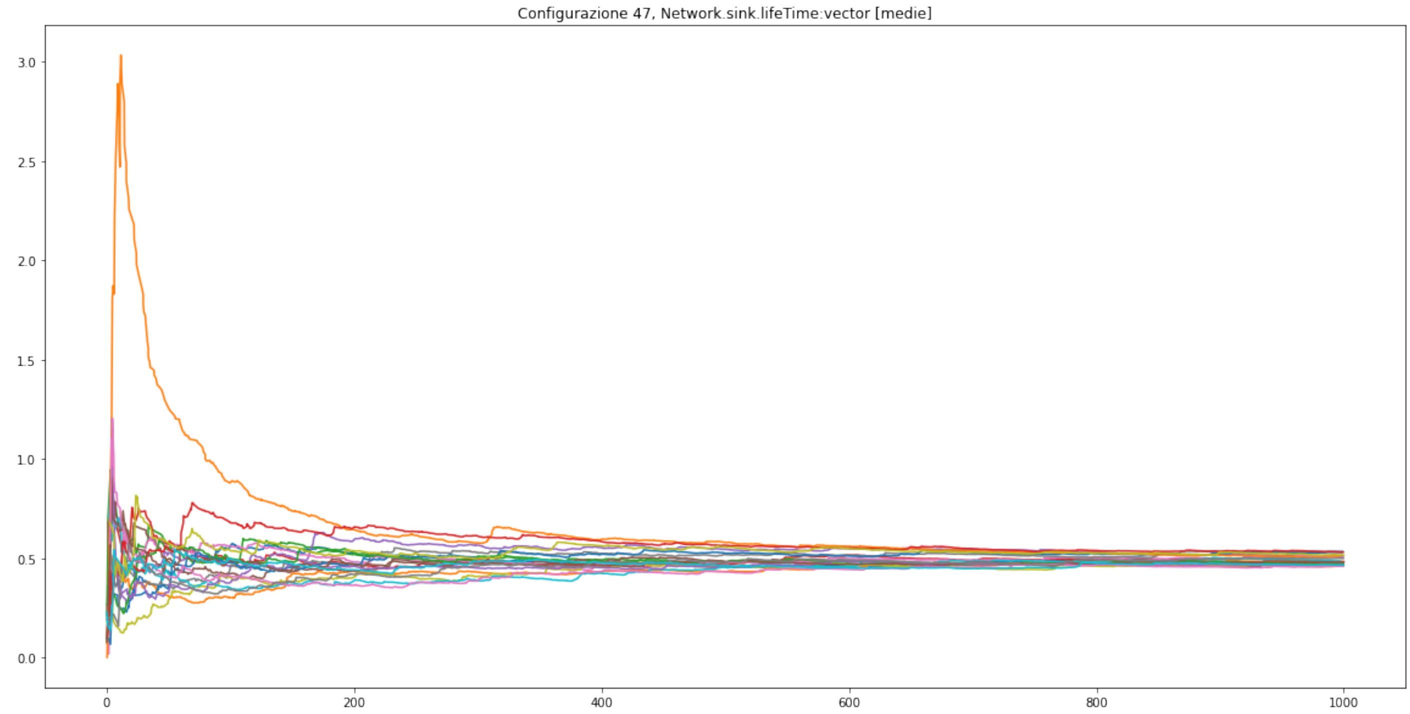

Configurazione 42, Network.sink.lifeTime:vector [medie]
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

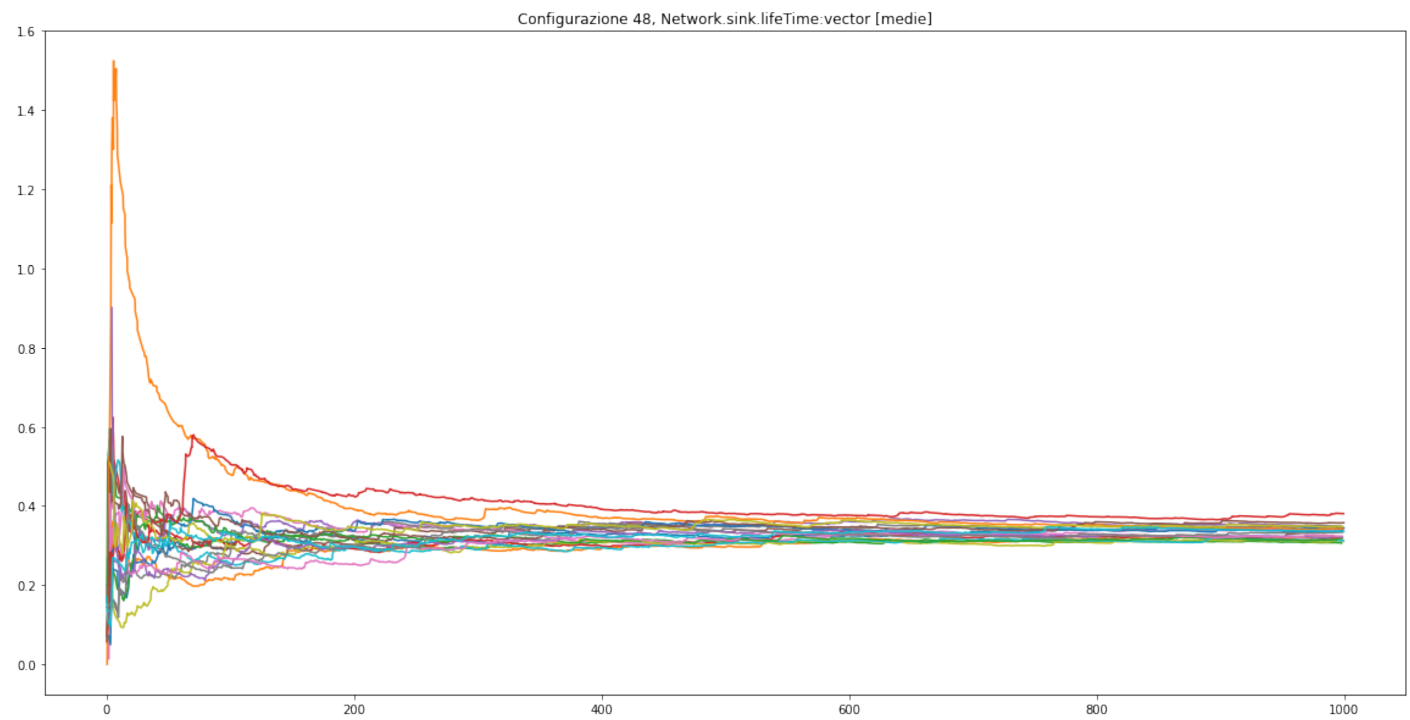

Configurazione 43, Network.sink.lifeTime:vector [medie]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
200
400
600
800
1000


Configurazione 44, Network.sink.lifeTime:vector [medie]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
200
400
600
800
1000

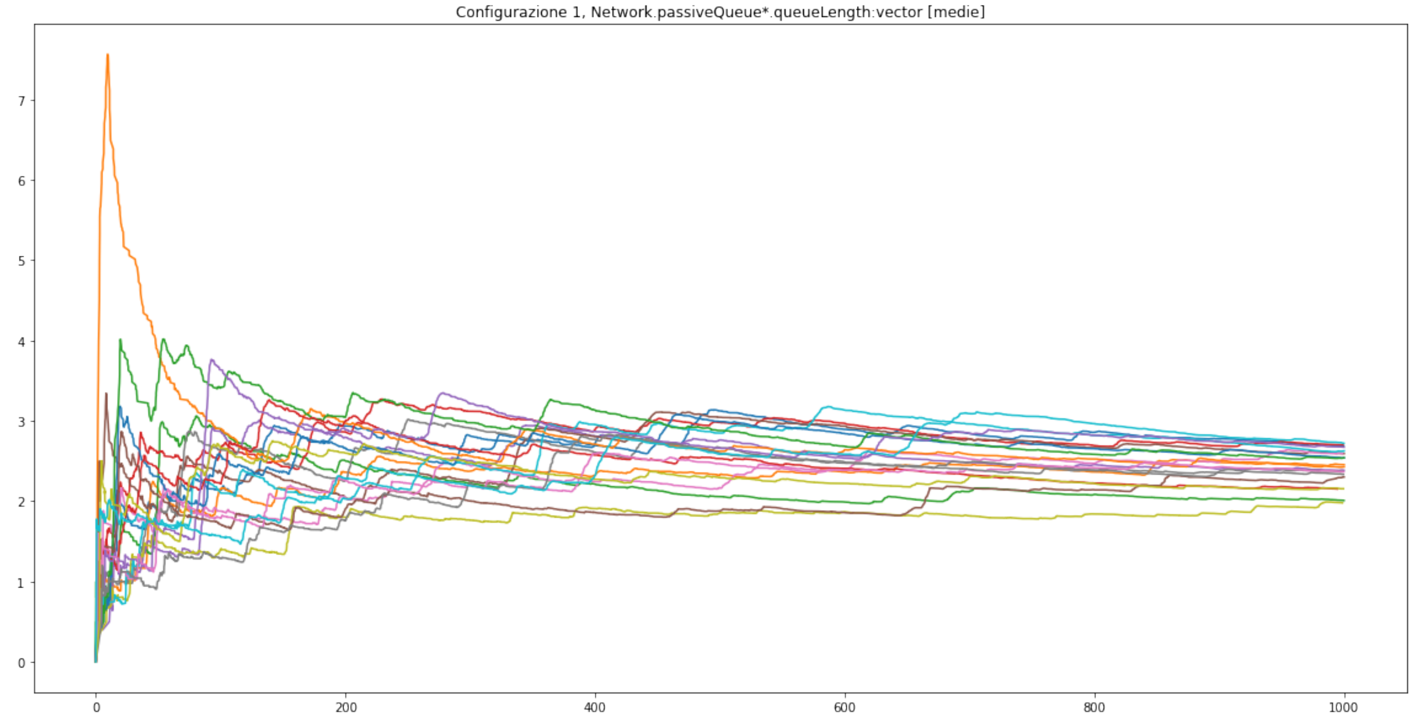

Configurazione 45, Network.sink.lifeTime:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

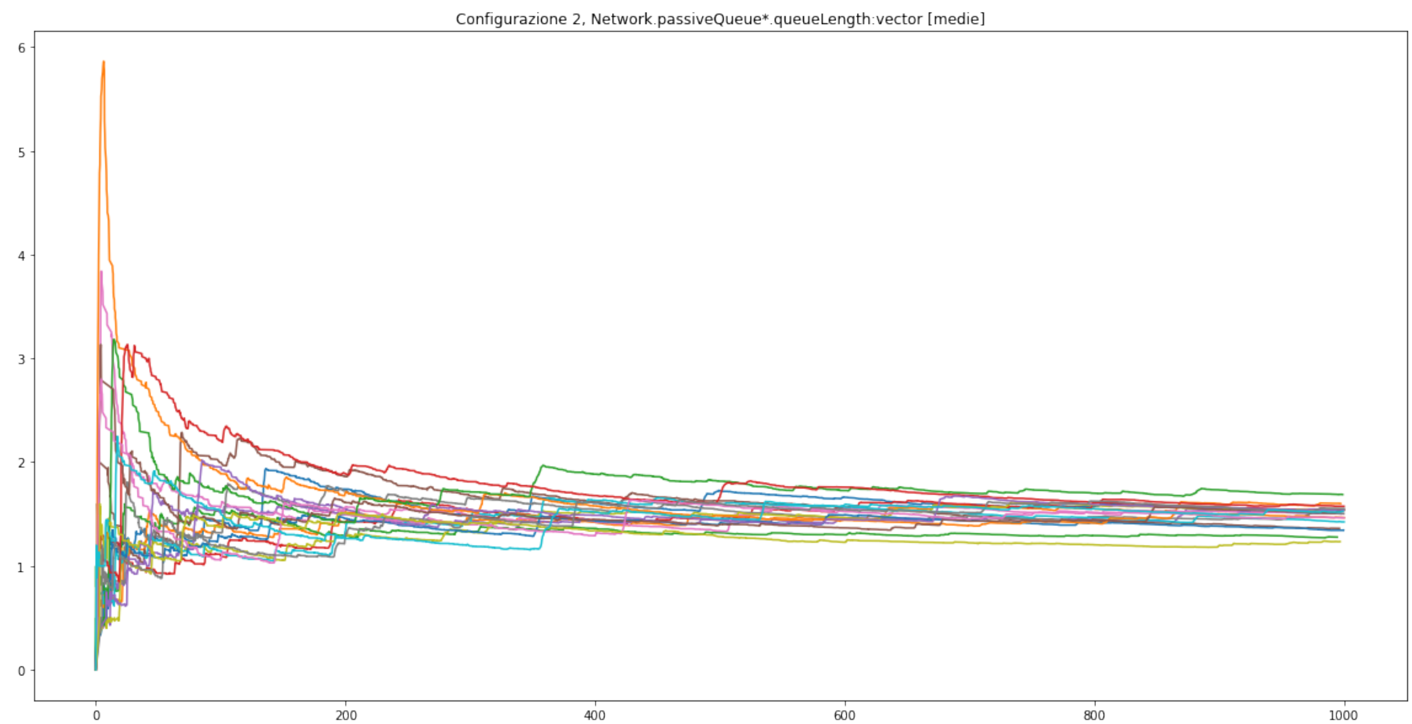

Configurazione 46, Network.sink.lifeTime:vector [medie]
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

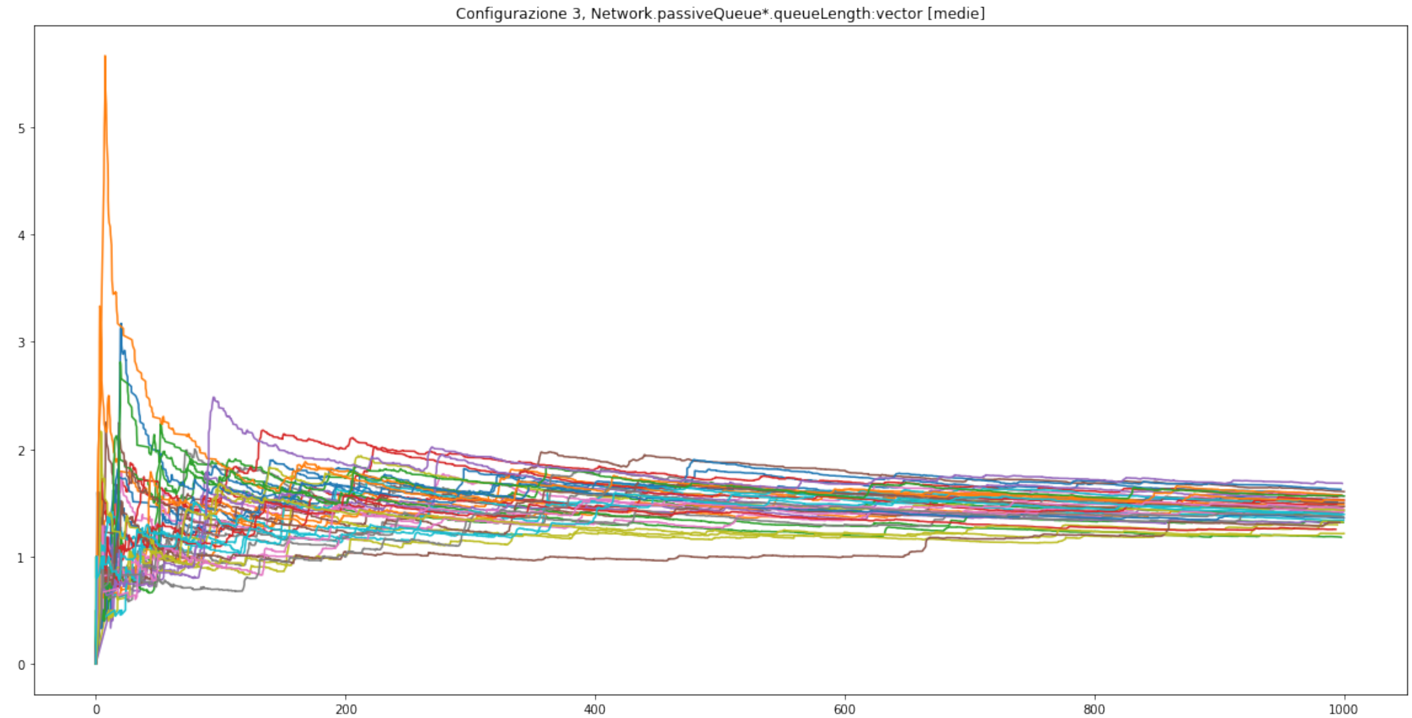

Configurazione 47, Network.sink.lifeTime:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

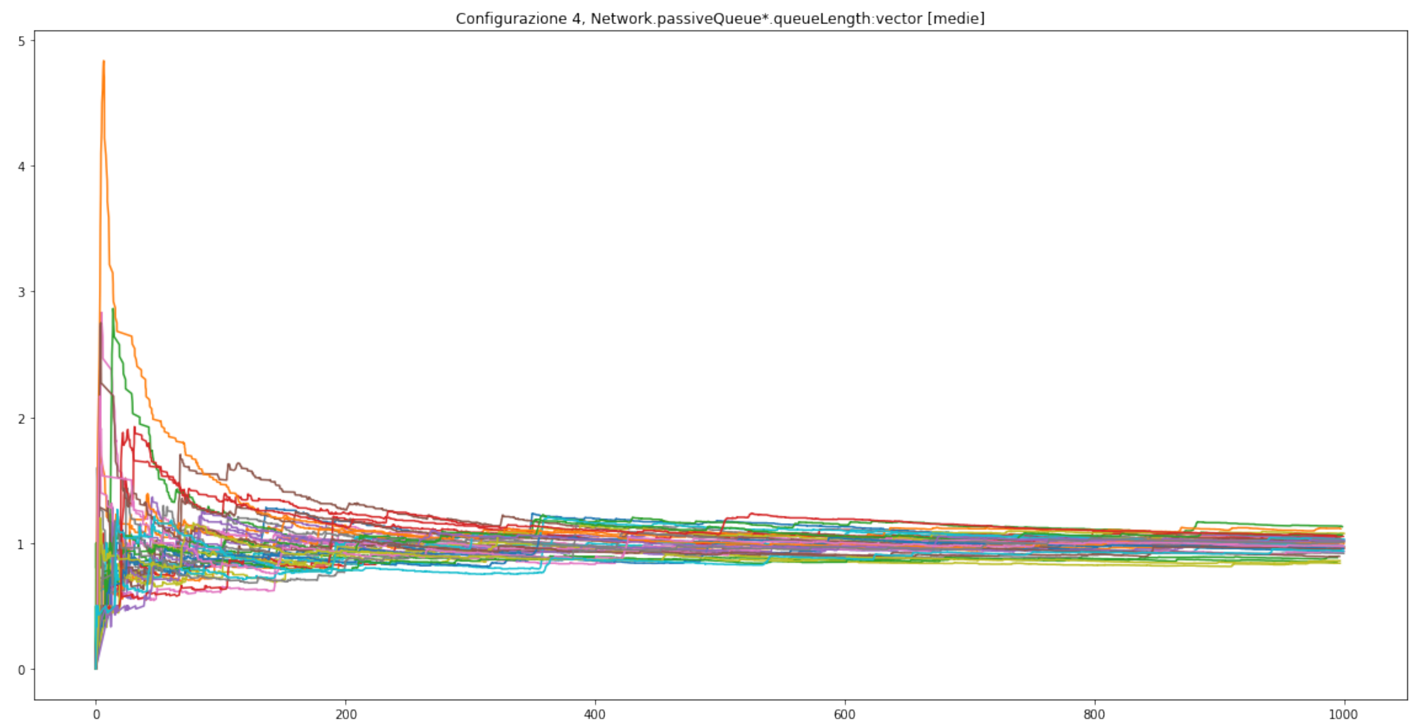

Configurazione 48, Network.sink.lifeTime:vector [medie]
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

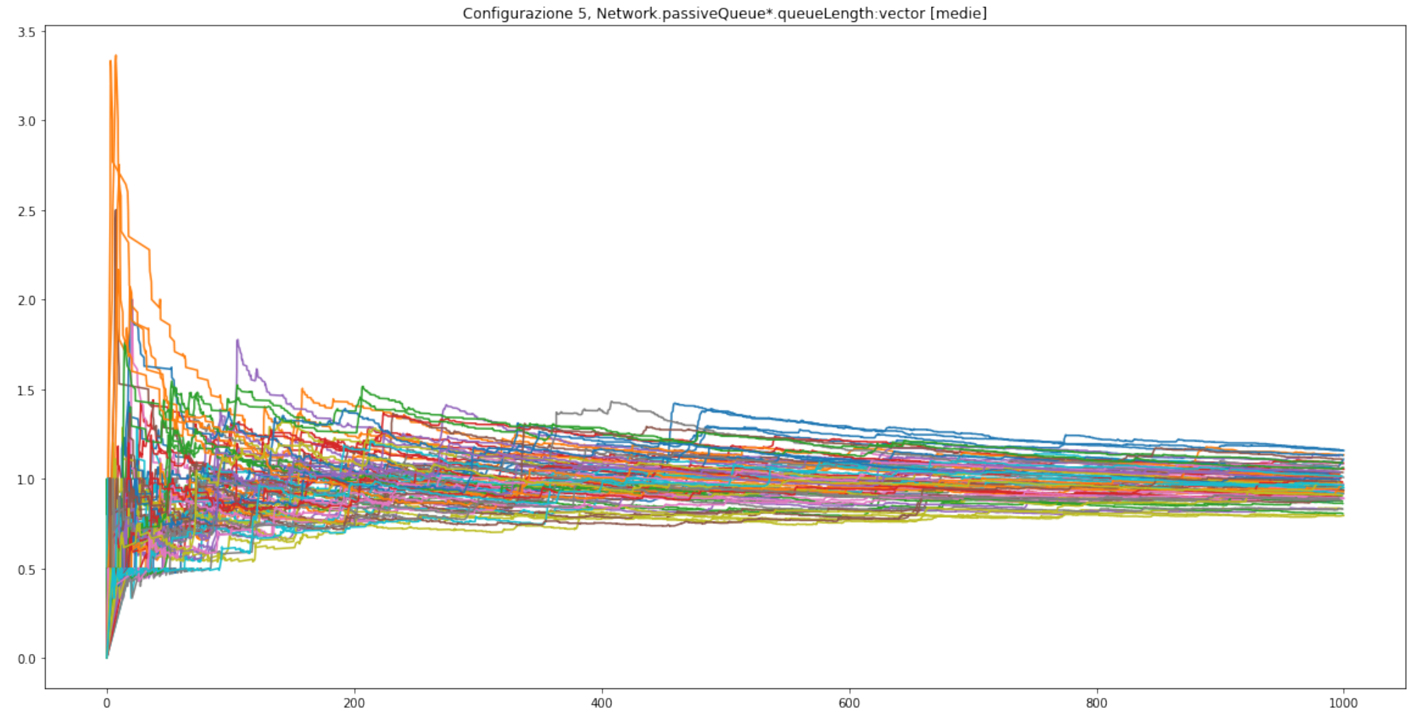
B.2 Network.passiveQueue*.queueLength:vector [medie]

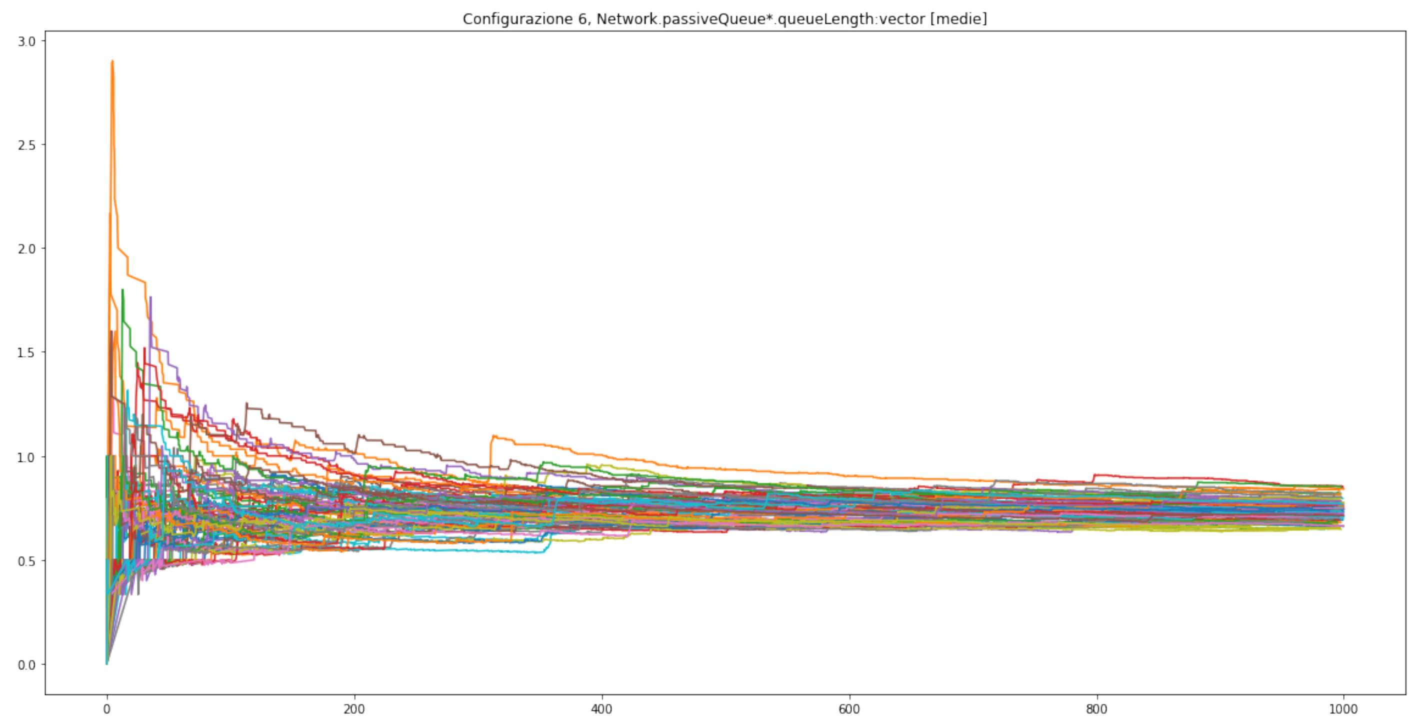


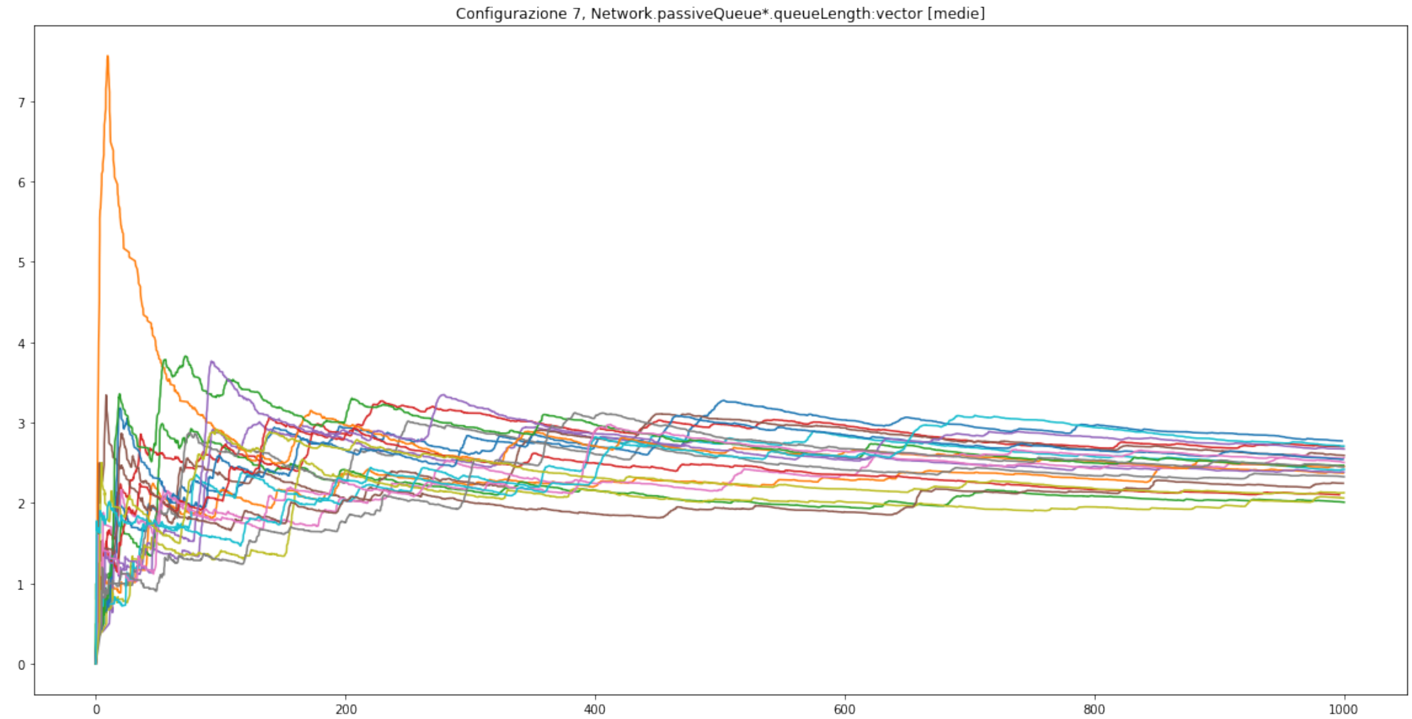


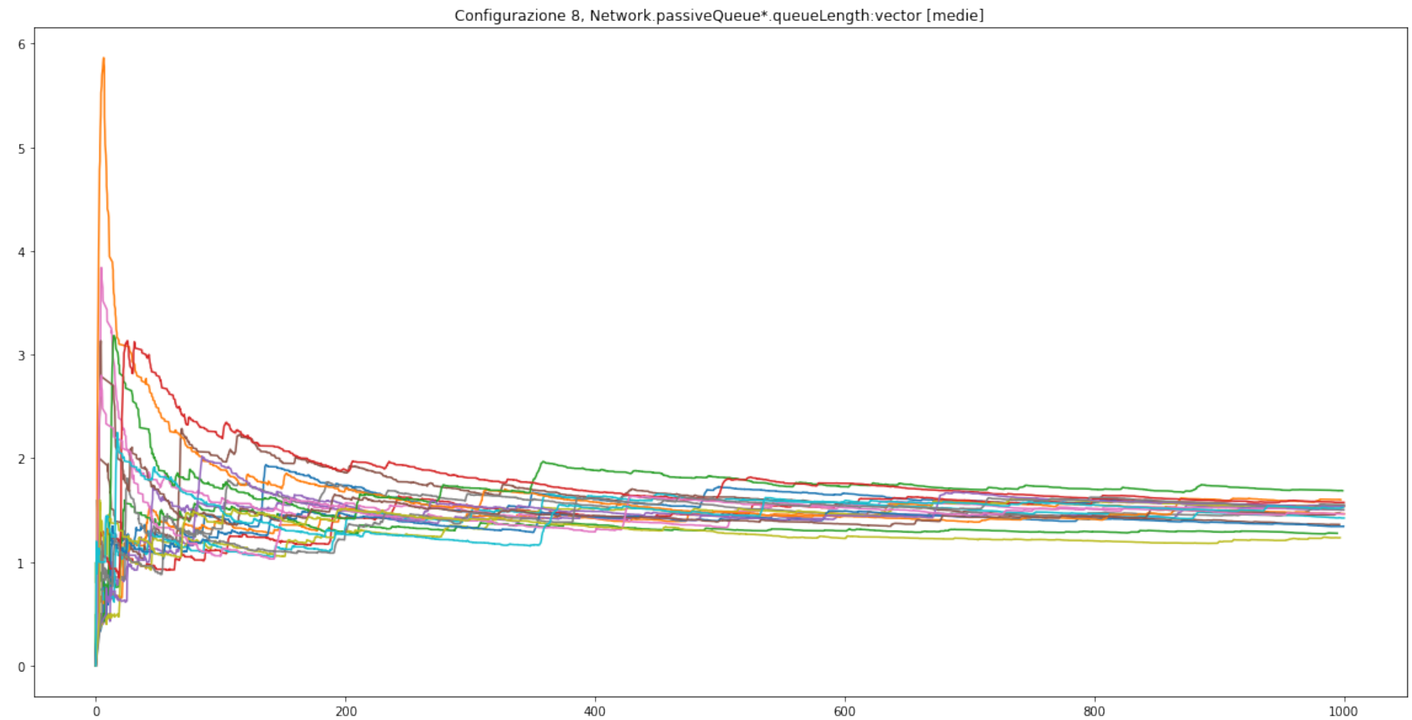

Configurazione 3, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
0
200
400
600
800
1000

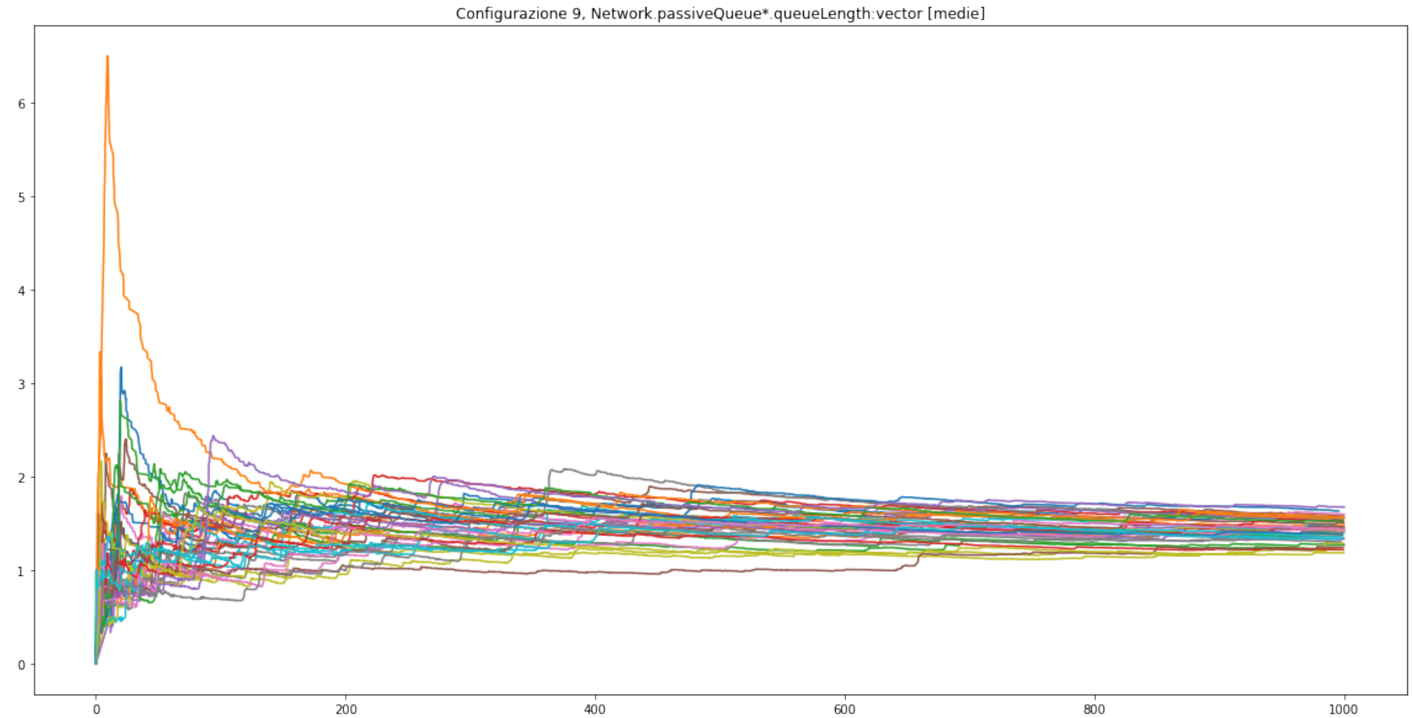

Configurazione 4, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
0
200
400
600
800
1000


Configurazione 5, Network.passiveQueue*.queueLength:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

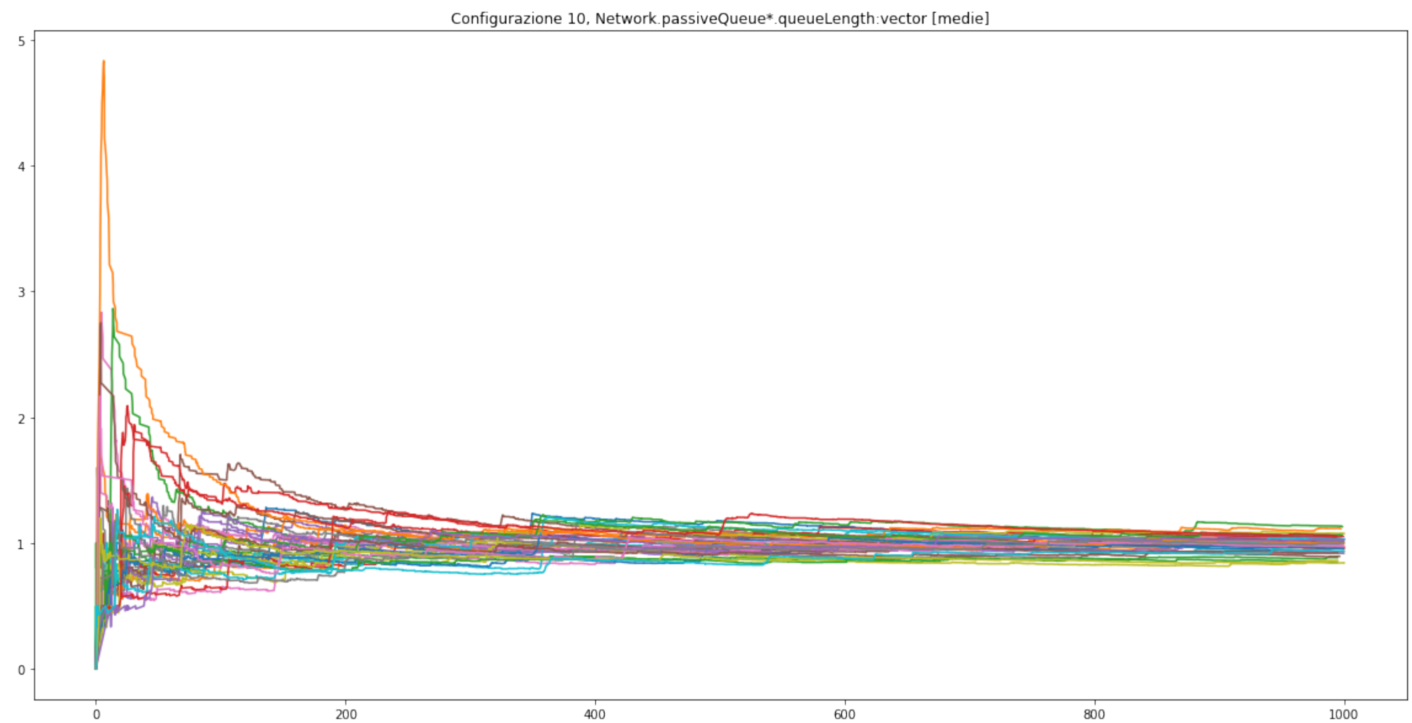

Configurazione 6, Network.passiveQueue*.queueLength:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000


Configurazione 7, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
6
7
0
200
400
600
800
1000

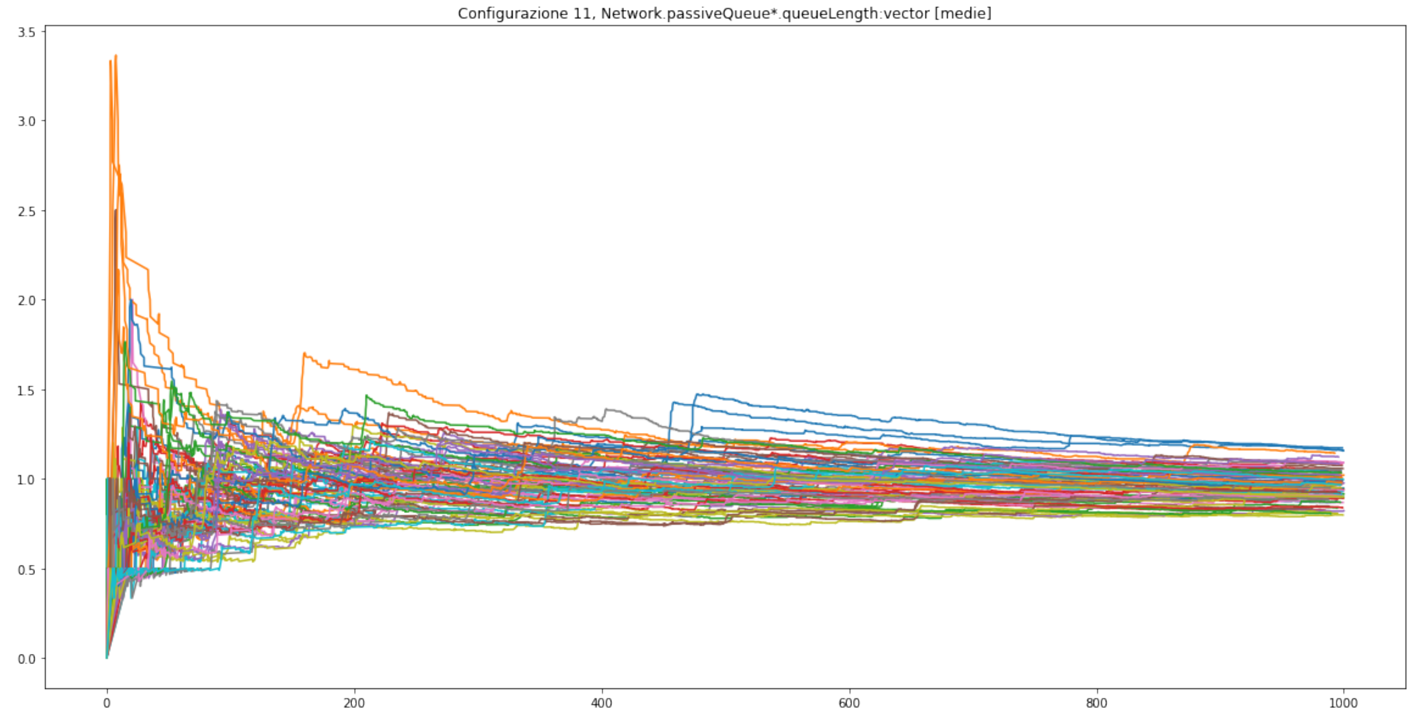

Configurazione 8, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
6
0
200
400
600
800
1000

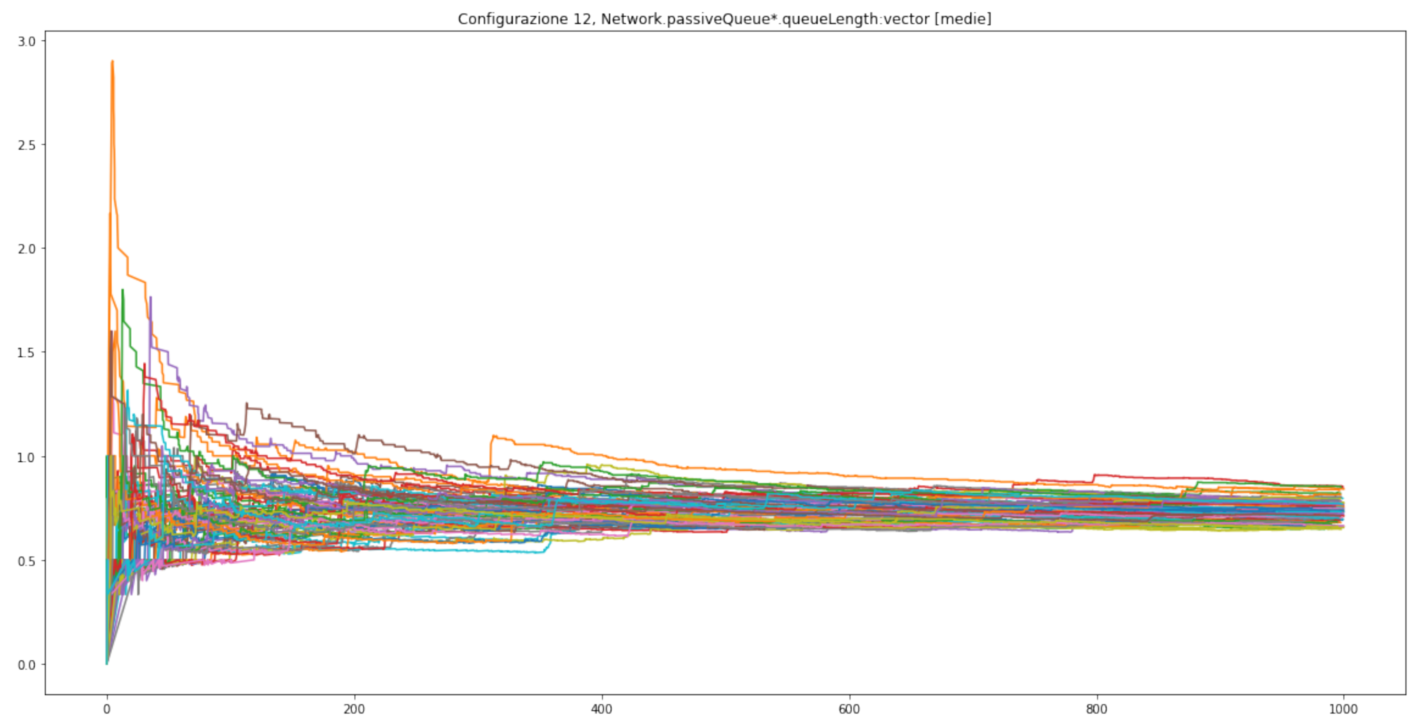


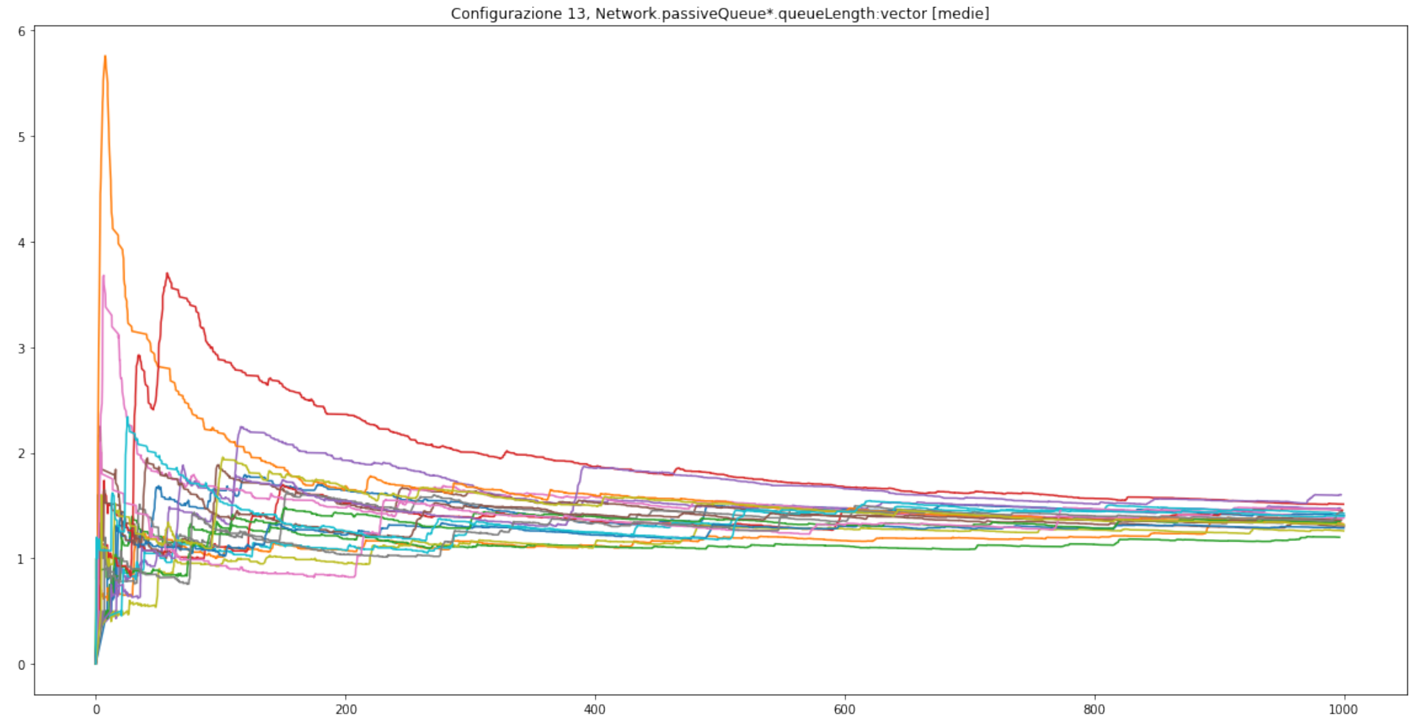
Configurazione 9, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
6
0
200
400
600
800
1000

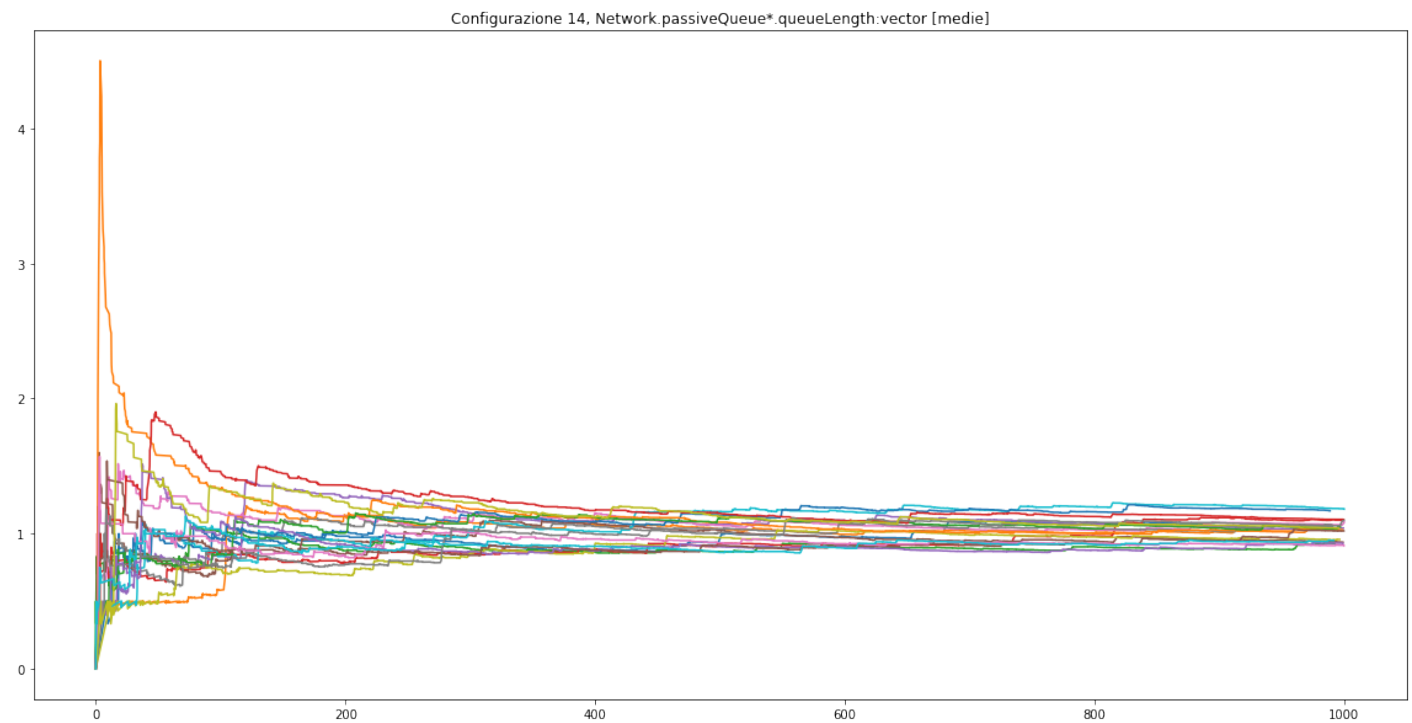


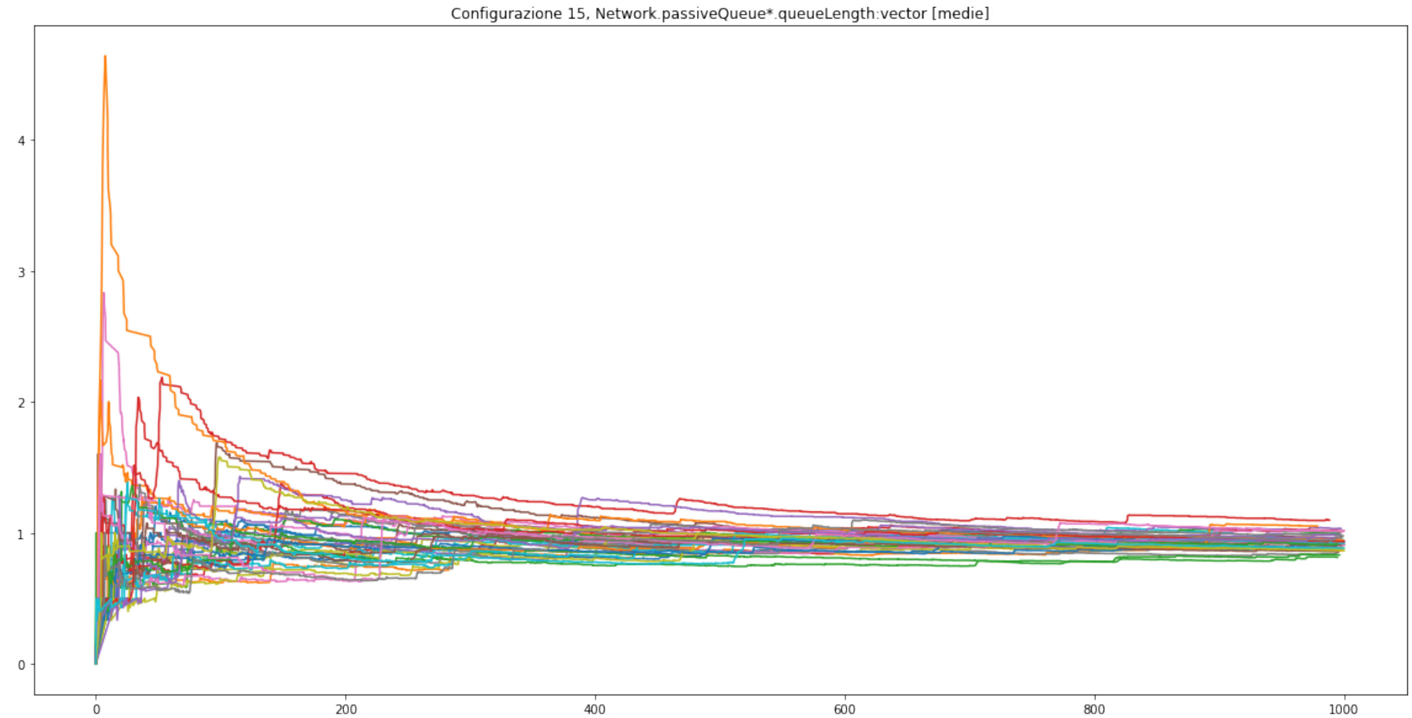
Configurazione 10, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
0
200
400
600
800
1000

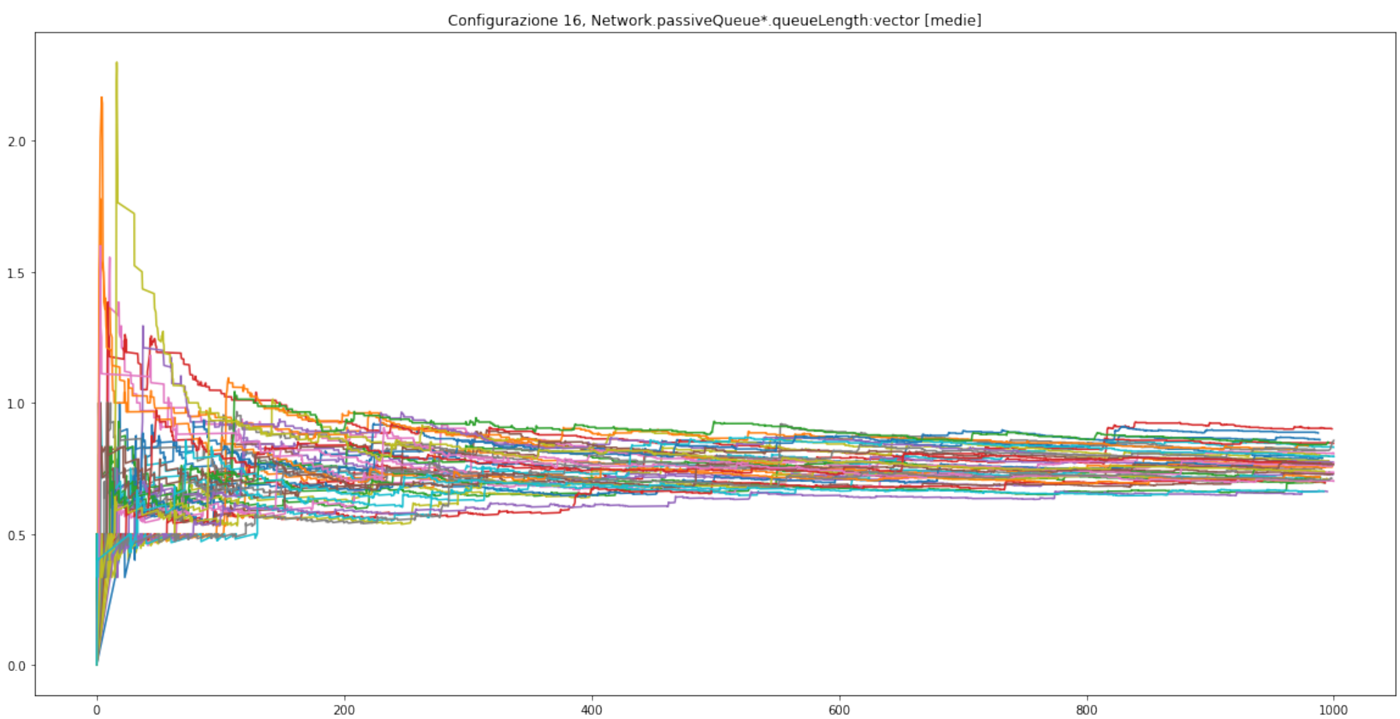

Configurazione 11, Network.passiveQueue*.queueLength:vector [medie]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

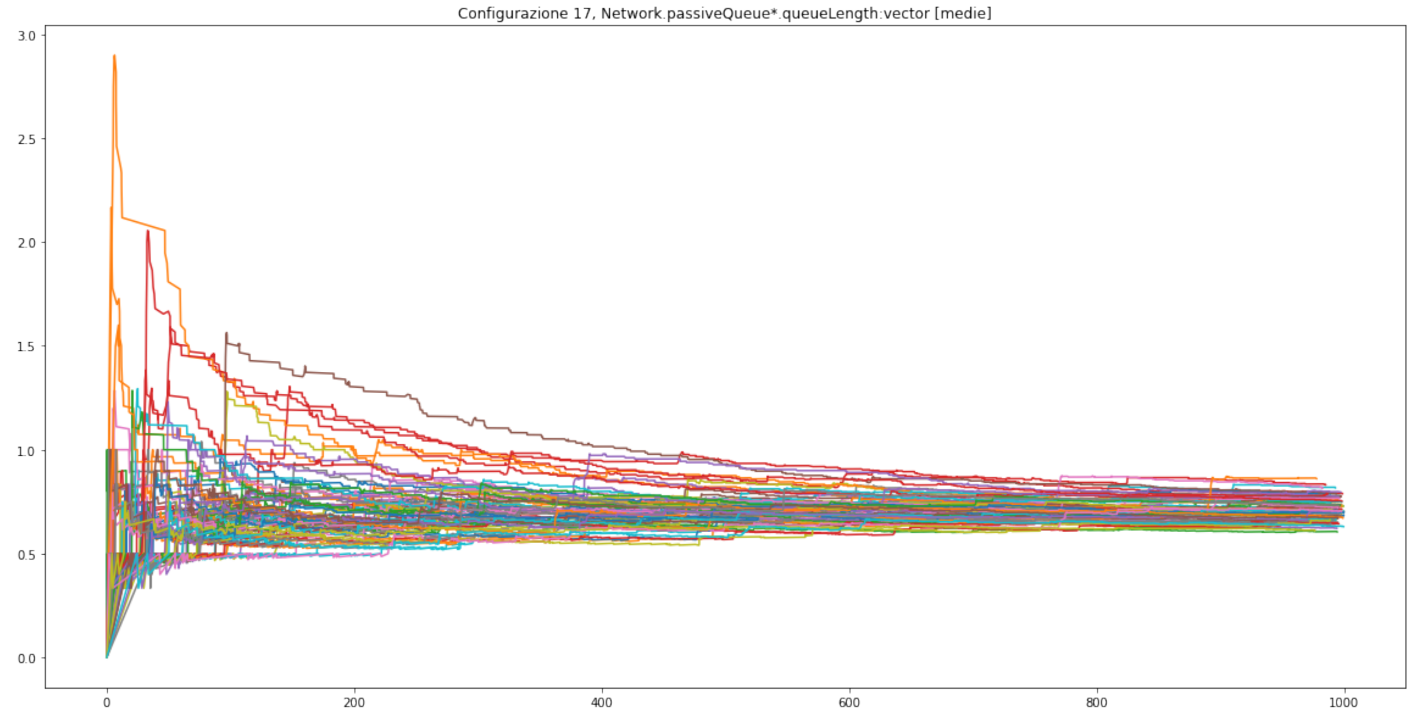

Configurazione 12, Network.passiveQueue*.queueLength:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

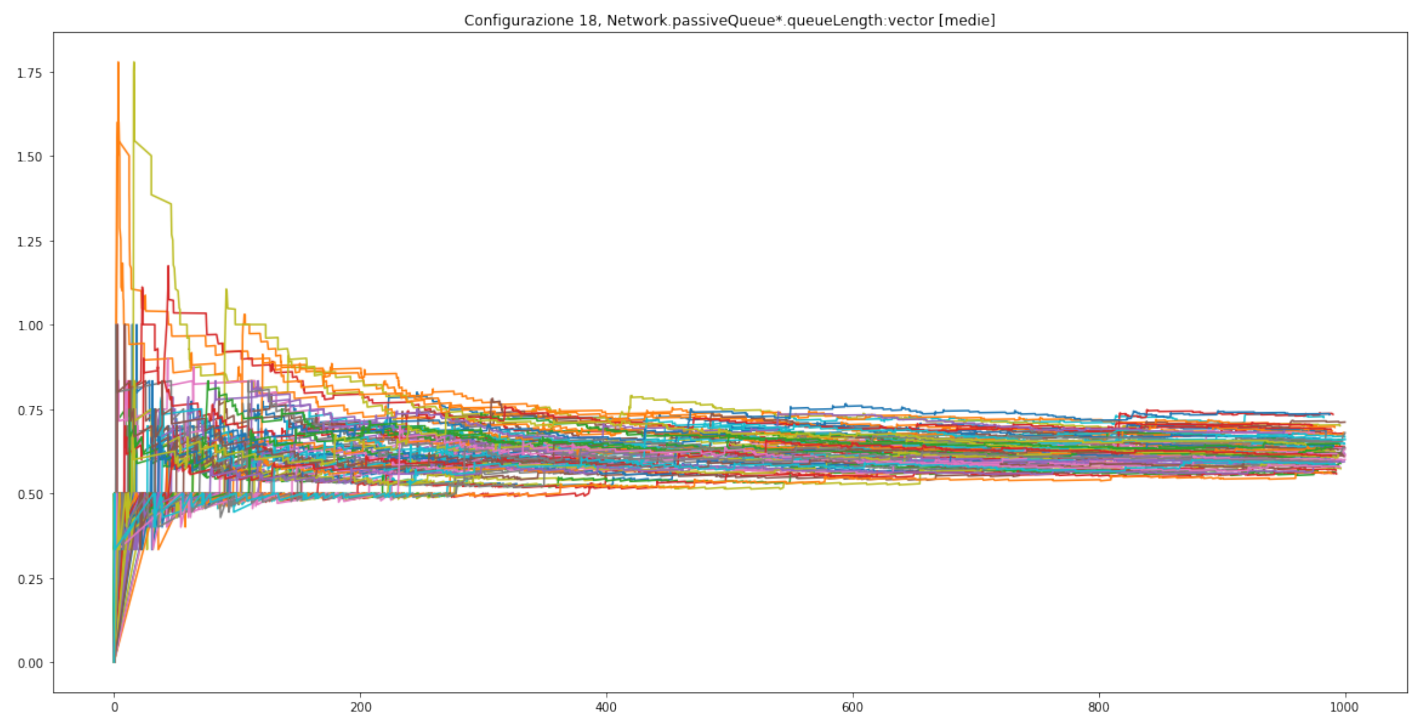

Configurazione 13, Network.passiveQueue*.queueLength:vector [medie]
6
5
4
3
2
1
0
0
200
400
600
800
1000

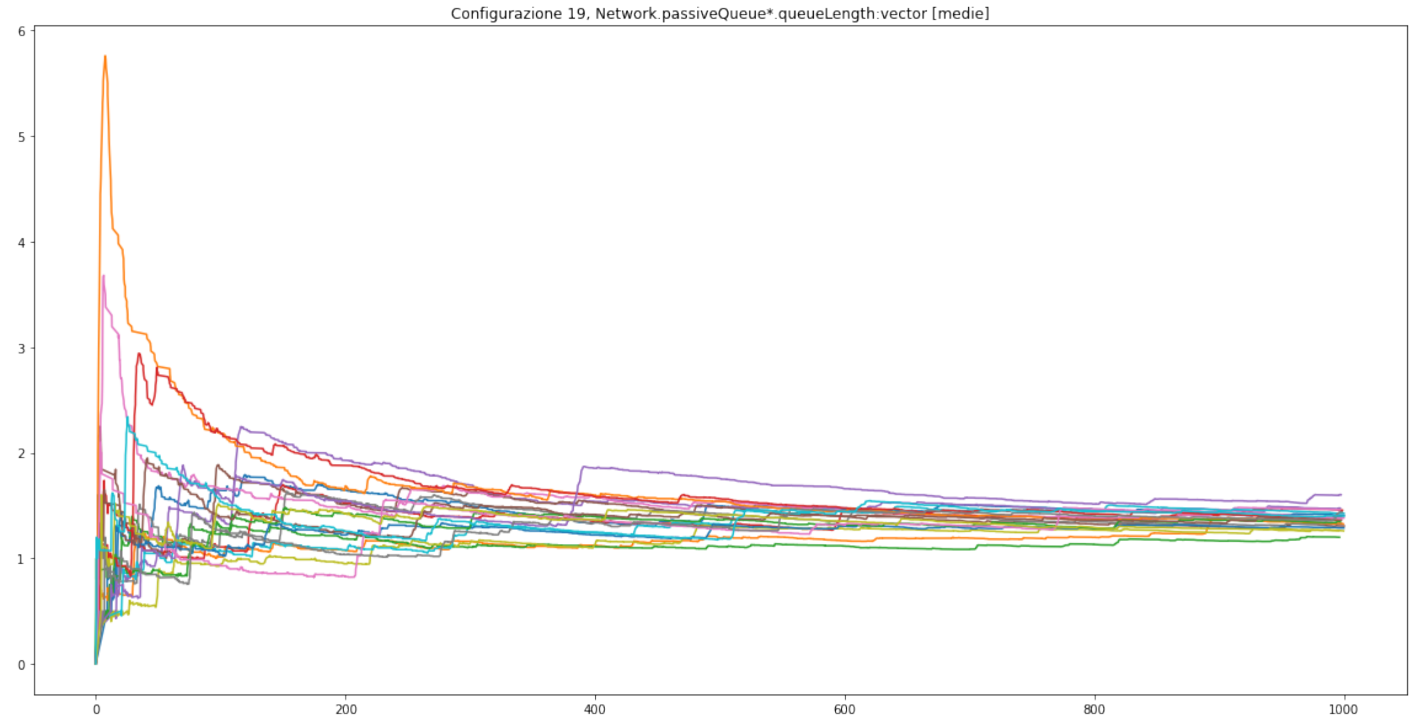

Configurazione 14, Network.passiveQueue*.queueLength:vector [medie]
4
3
2
1
0
0
200
400
600
800
1000

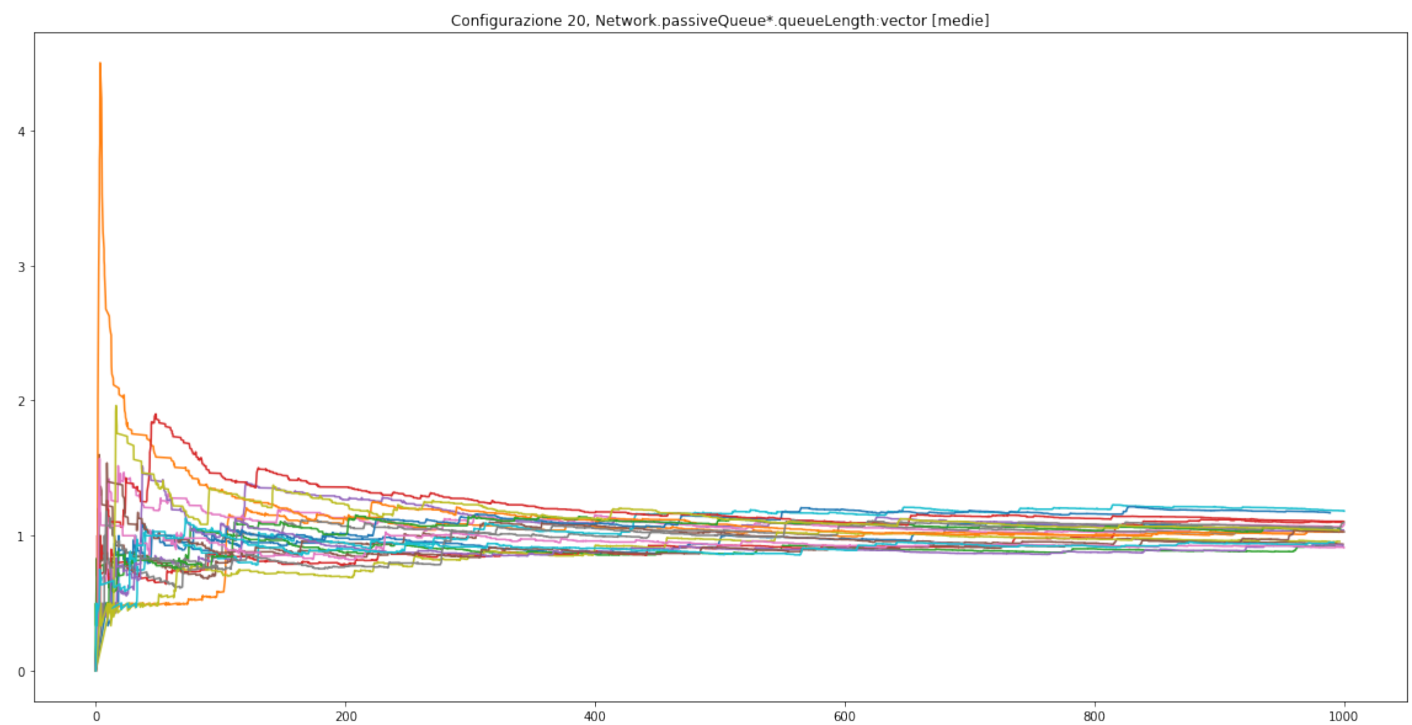

Configurazione 15, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
0
200
400
600
800
1000

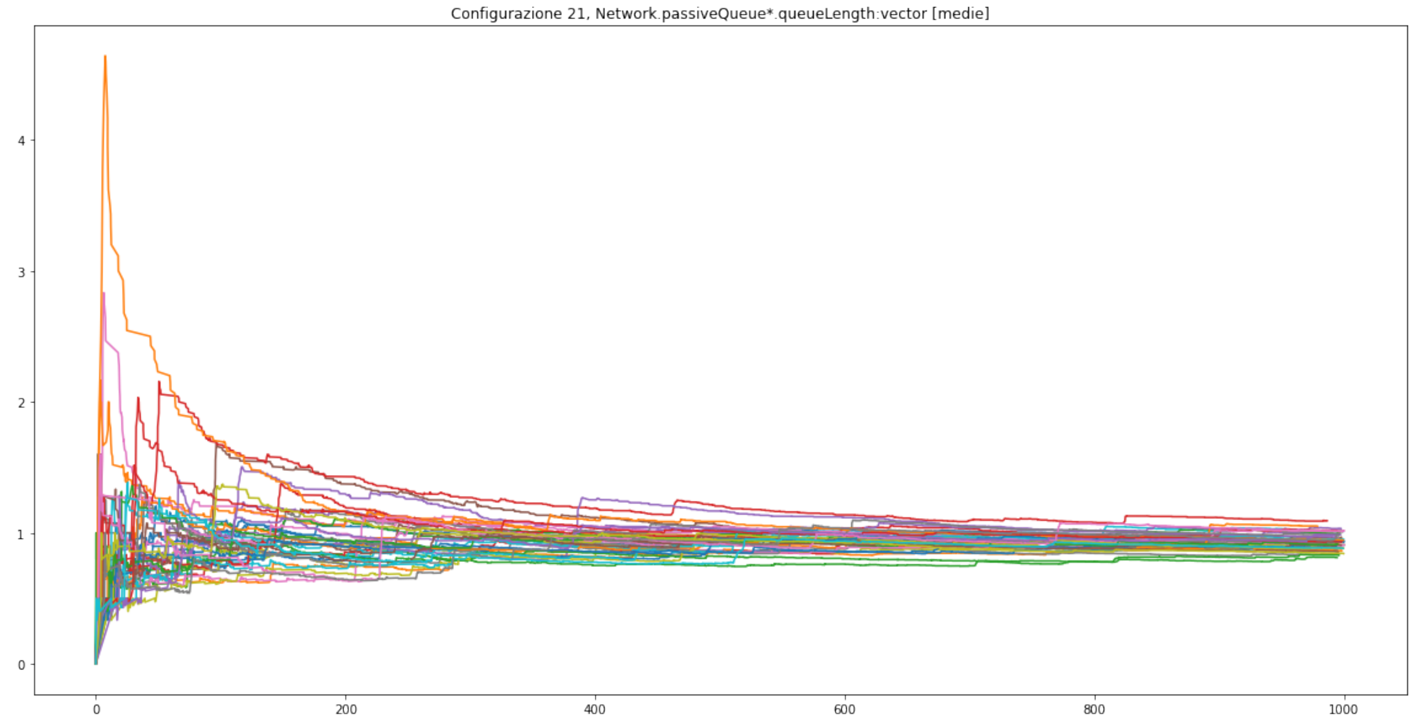

Configurazione 16, Network.passiveQueue*.queueLength:vector [medie]
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
1000

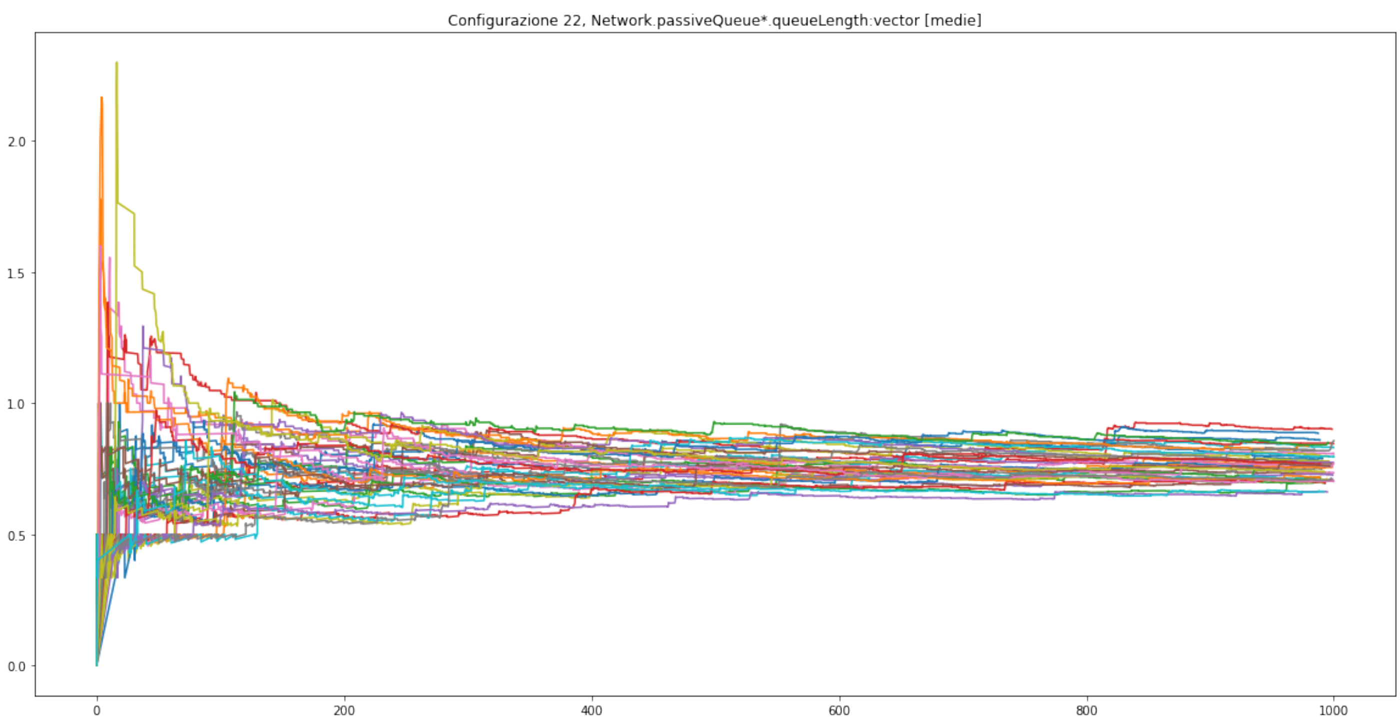

Configurazione 17, Network.passiveQueue*.queueLength:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

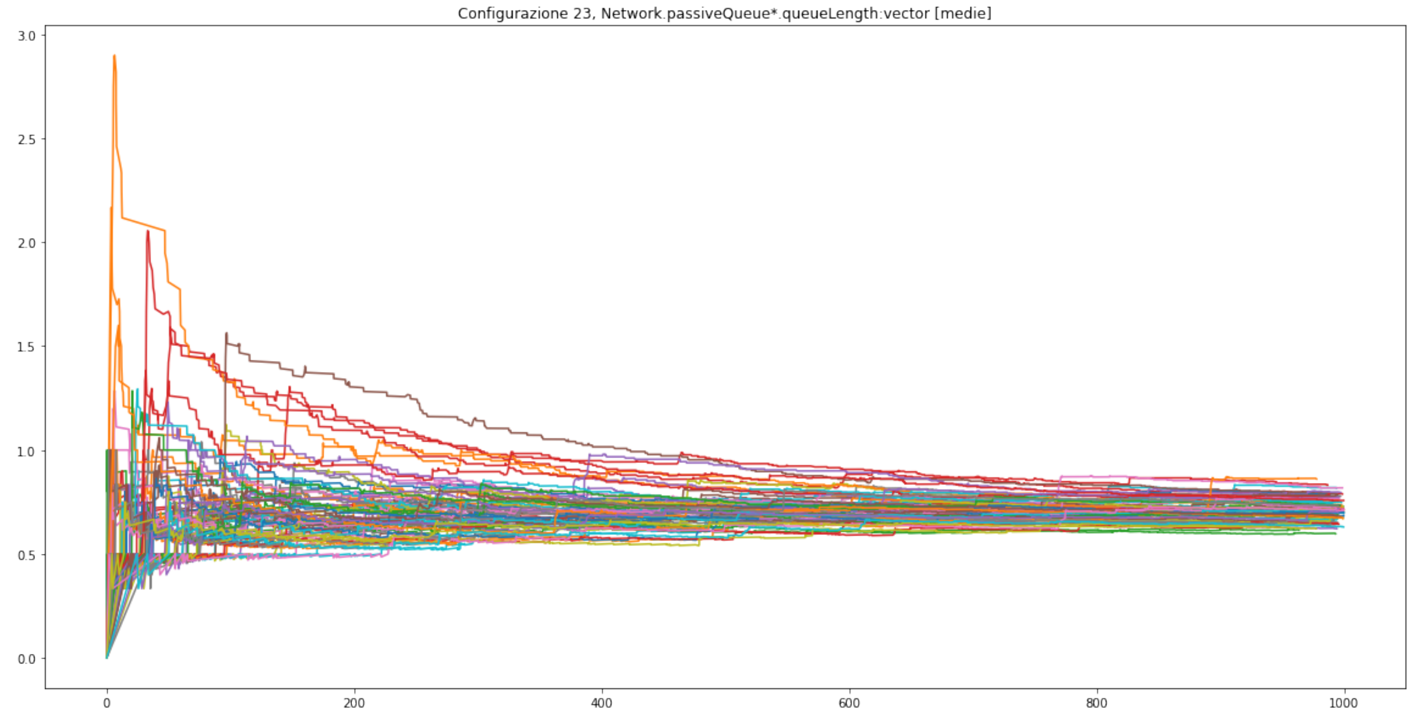

Configurazione 18, Network.passiveQueue*.queueLength:vector [medie]
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

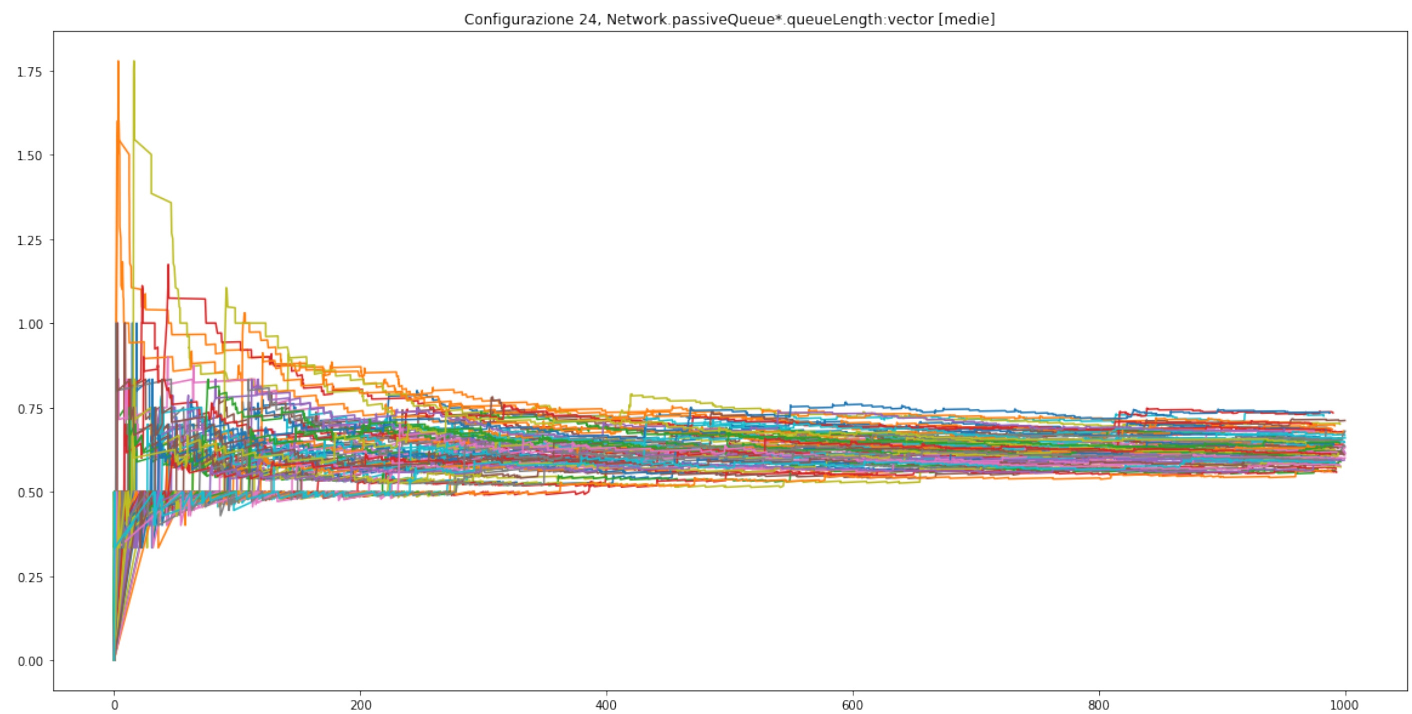

Configurazione 19, Network.passiveQueue*.queueLength:vector [medie]
6
5
4
3
2
1
0
0
200
400
600
800
1000

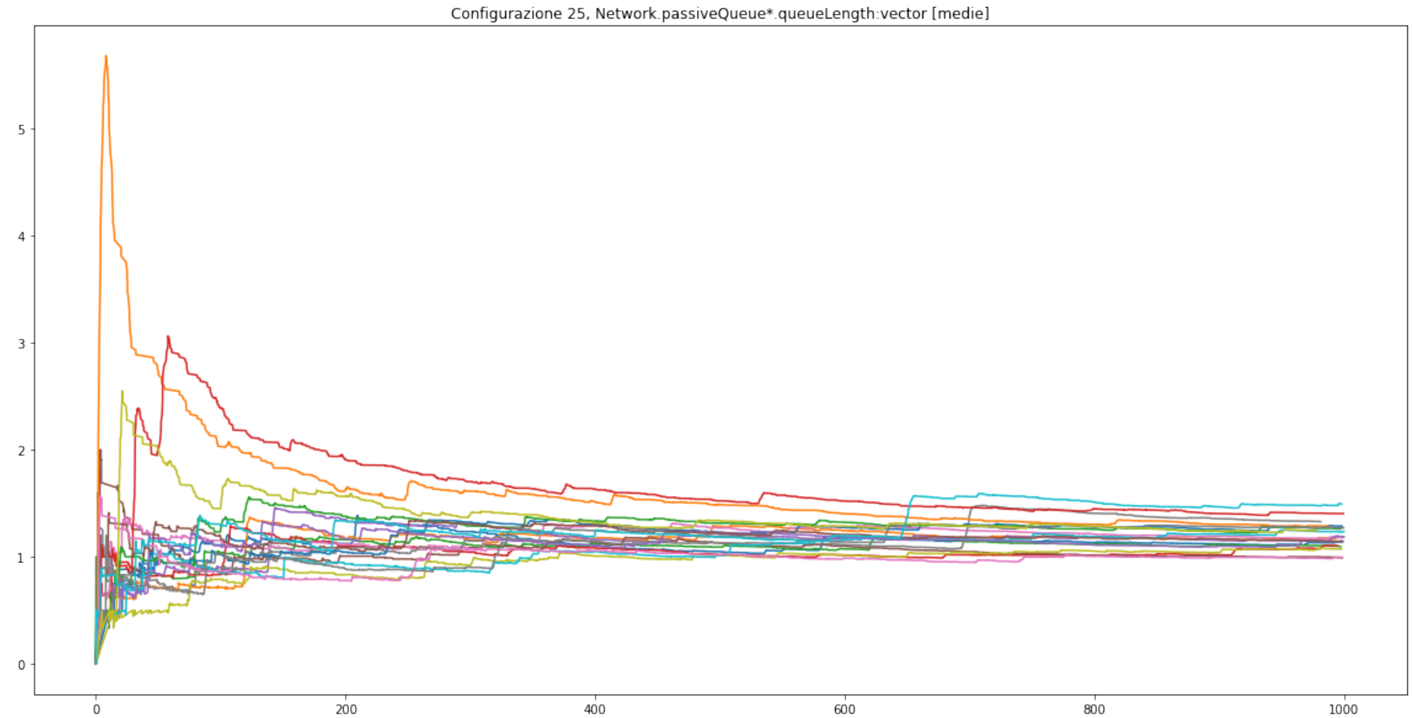

Configurazione 20, Network.passiveQueue*.queueLength:vector [medie]
4
3
2
1
0
0
200
400
600
800
1000

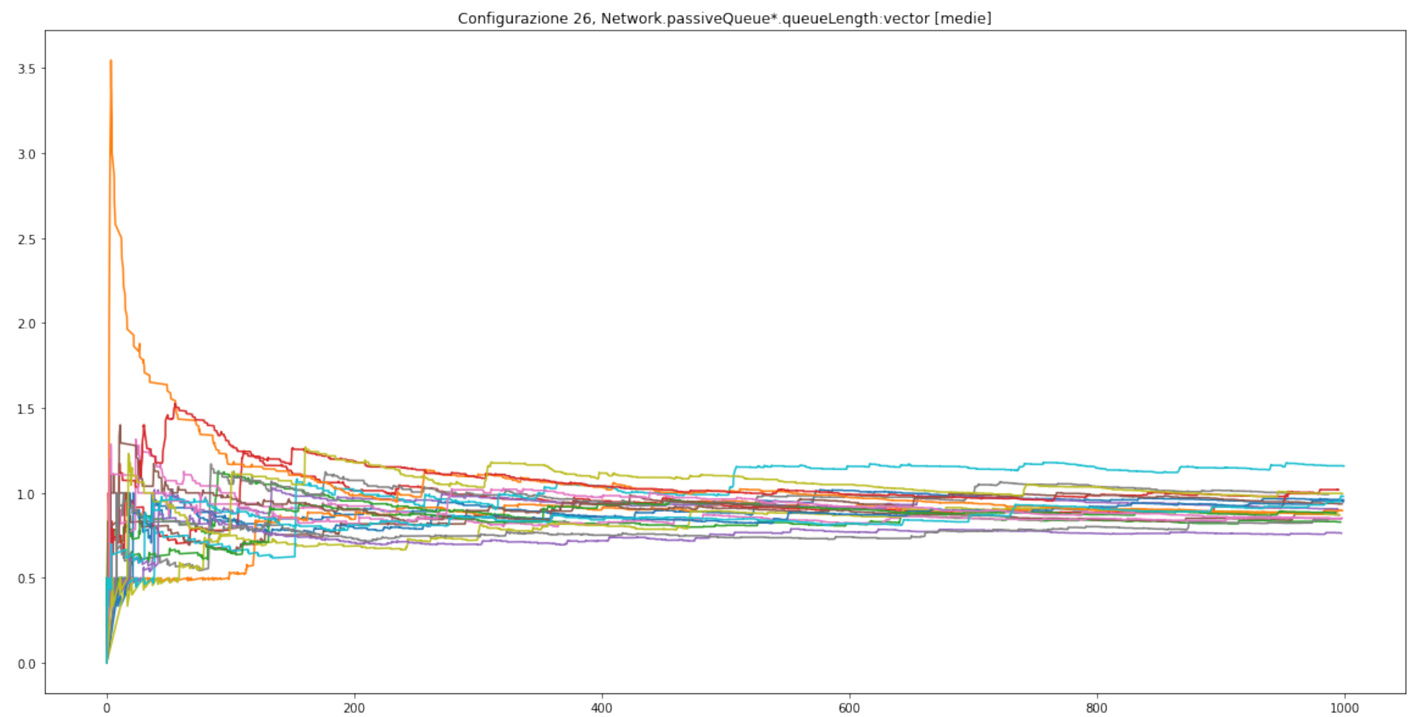

Configurazione 21, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
0
200
400
600
800
1000

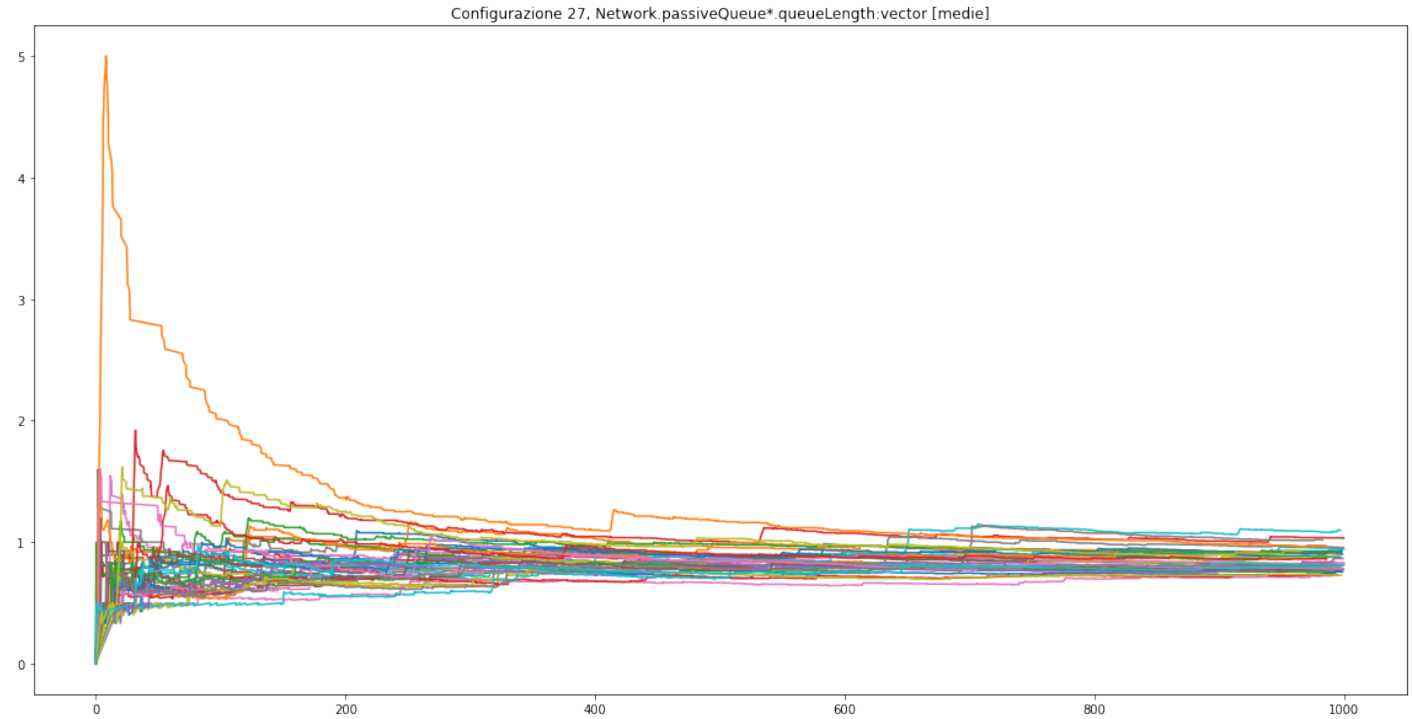

Configurazione 22, Network.passiveQueue*.queueLength:vector [medie]
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
1000

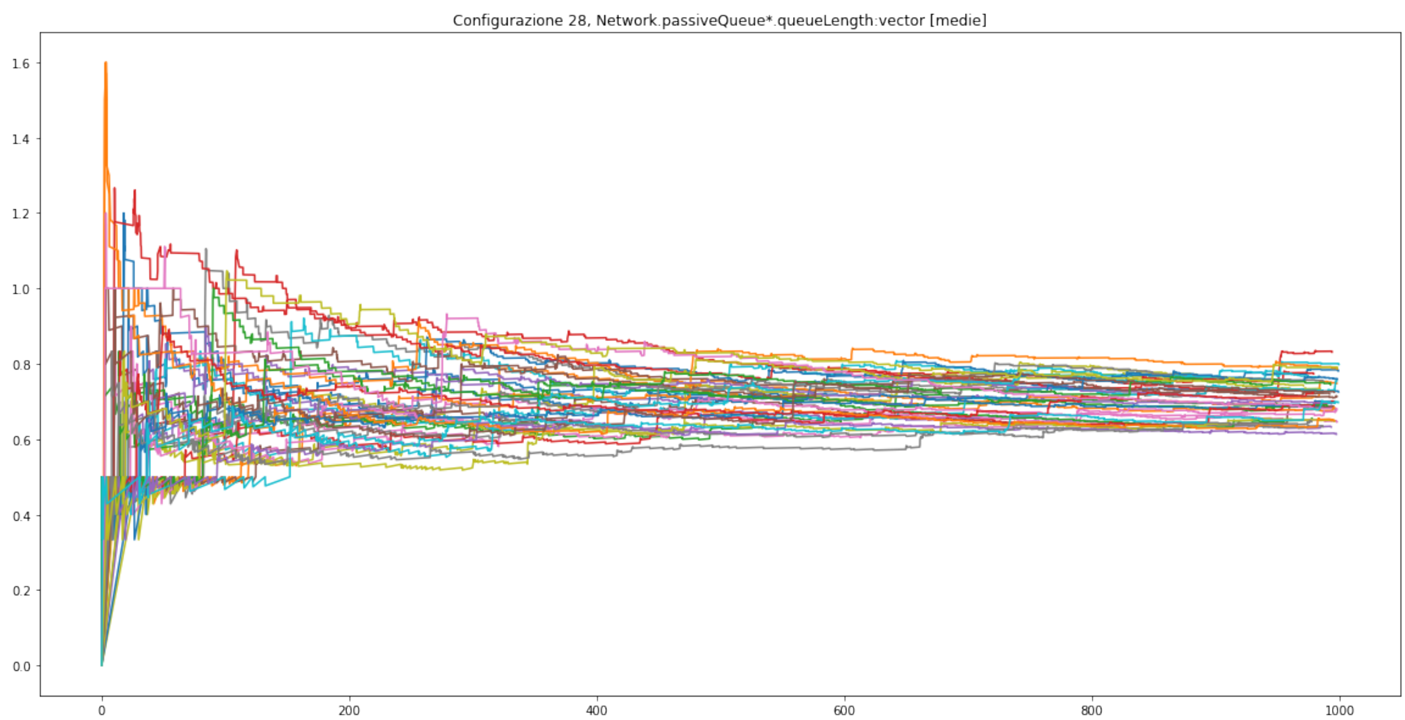

Configurazione 23, Network.passiveQueue*.queueLength:vector [medie]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

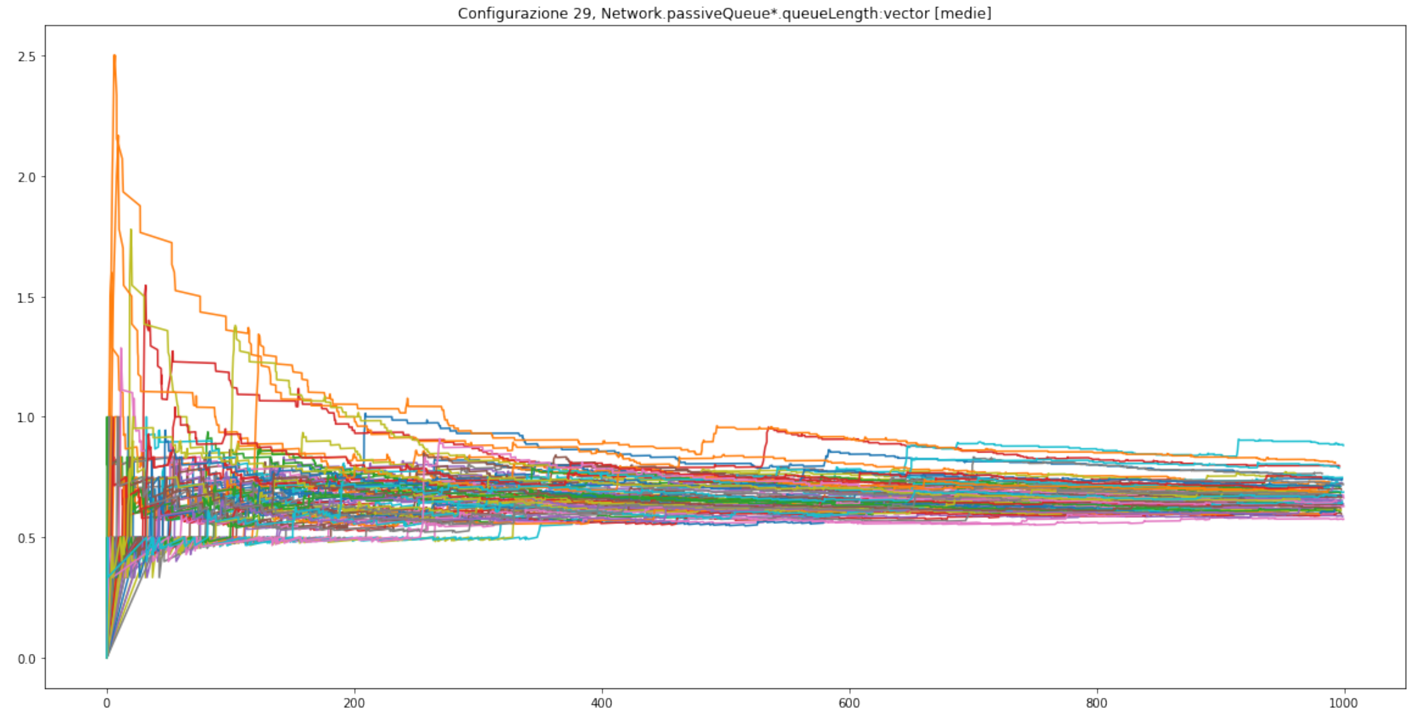

Configurazione 24, Network.passiveQueue*.queueLength:vector [medie]
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

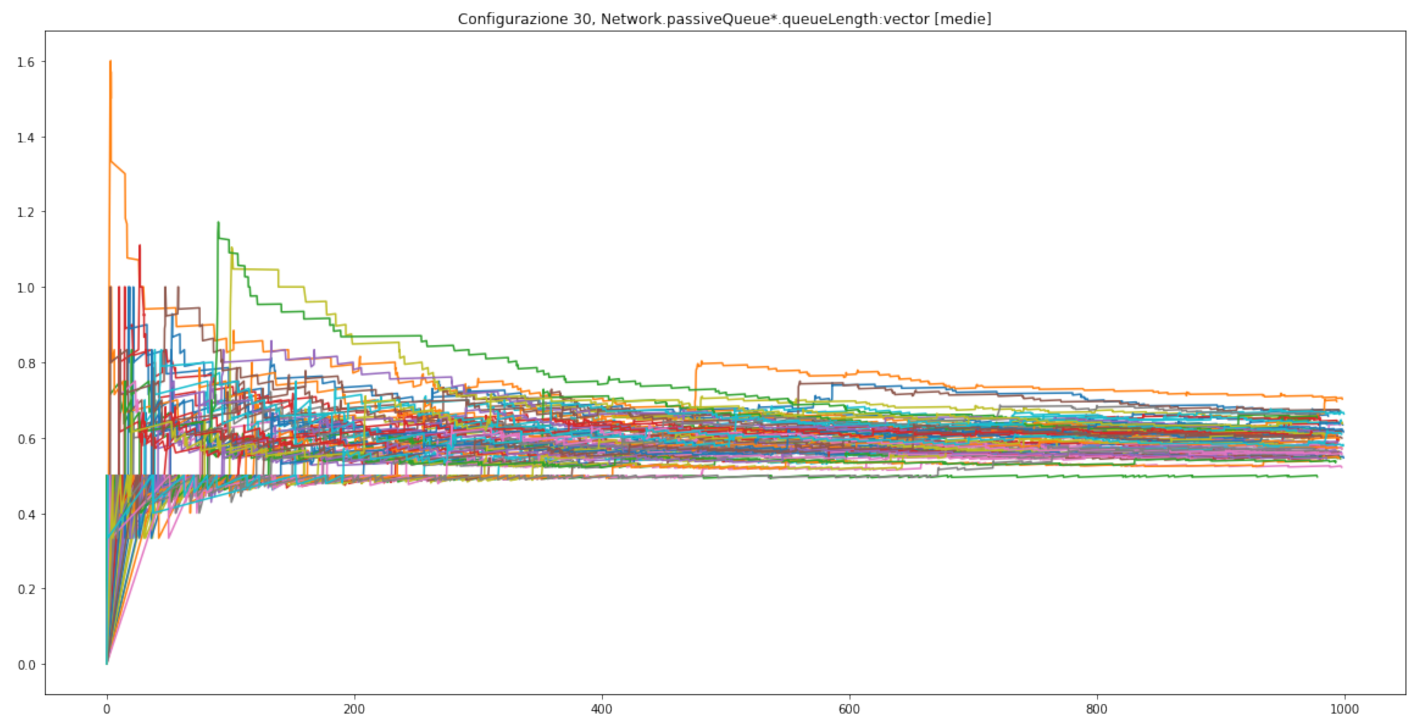

Configurazione 25, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
0
200
400
600
800
1000

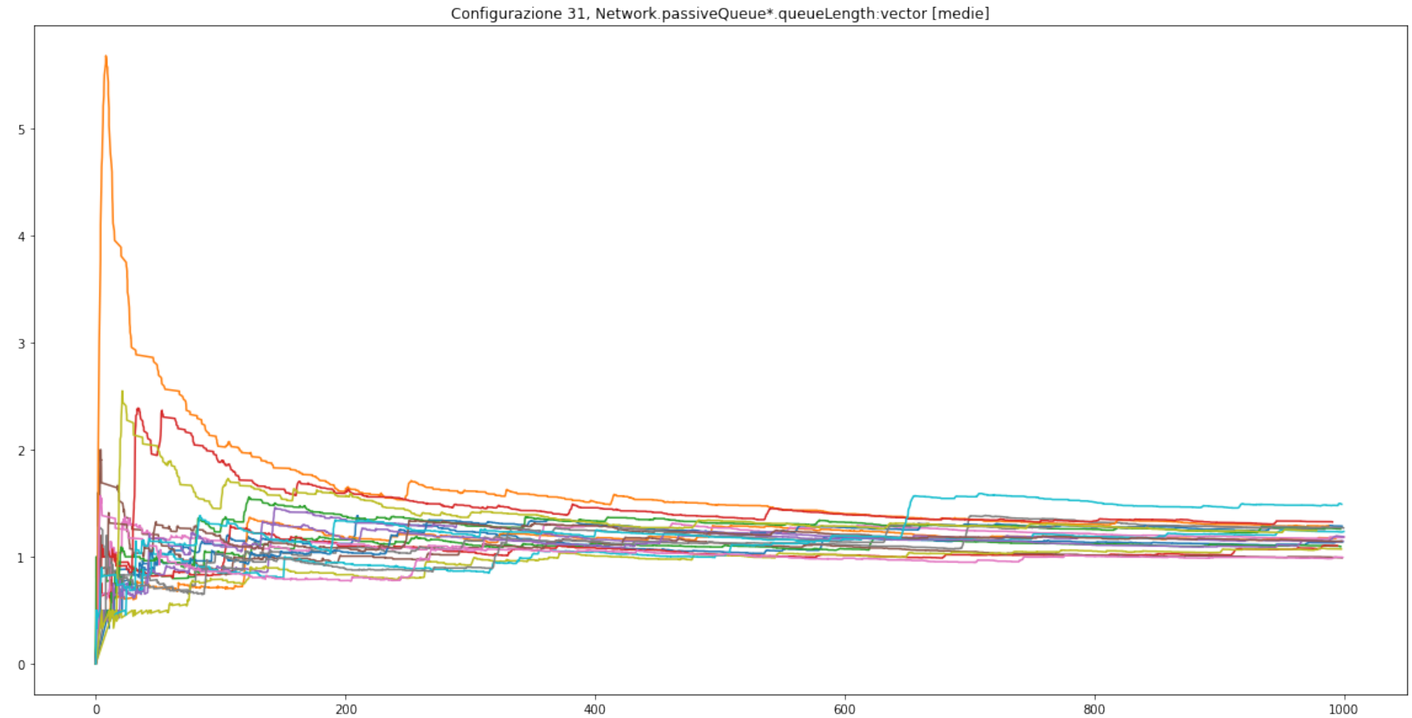

Configurazione 26, Network.passiveQueue*.queueLength:vector [medie]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
200
400
600
800
1000

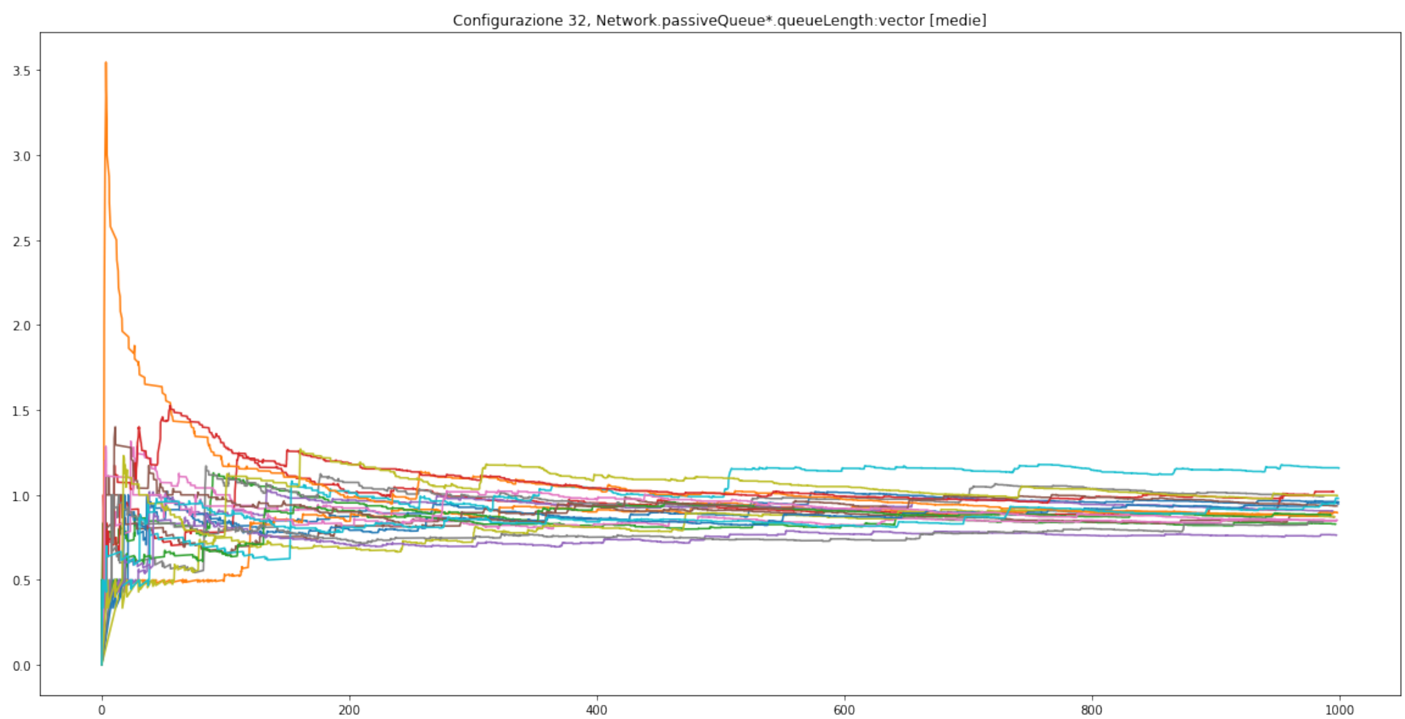

Configurazione 27, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
0
200
400
600
800
1000

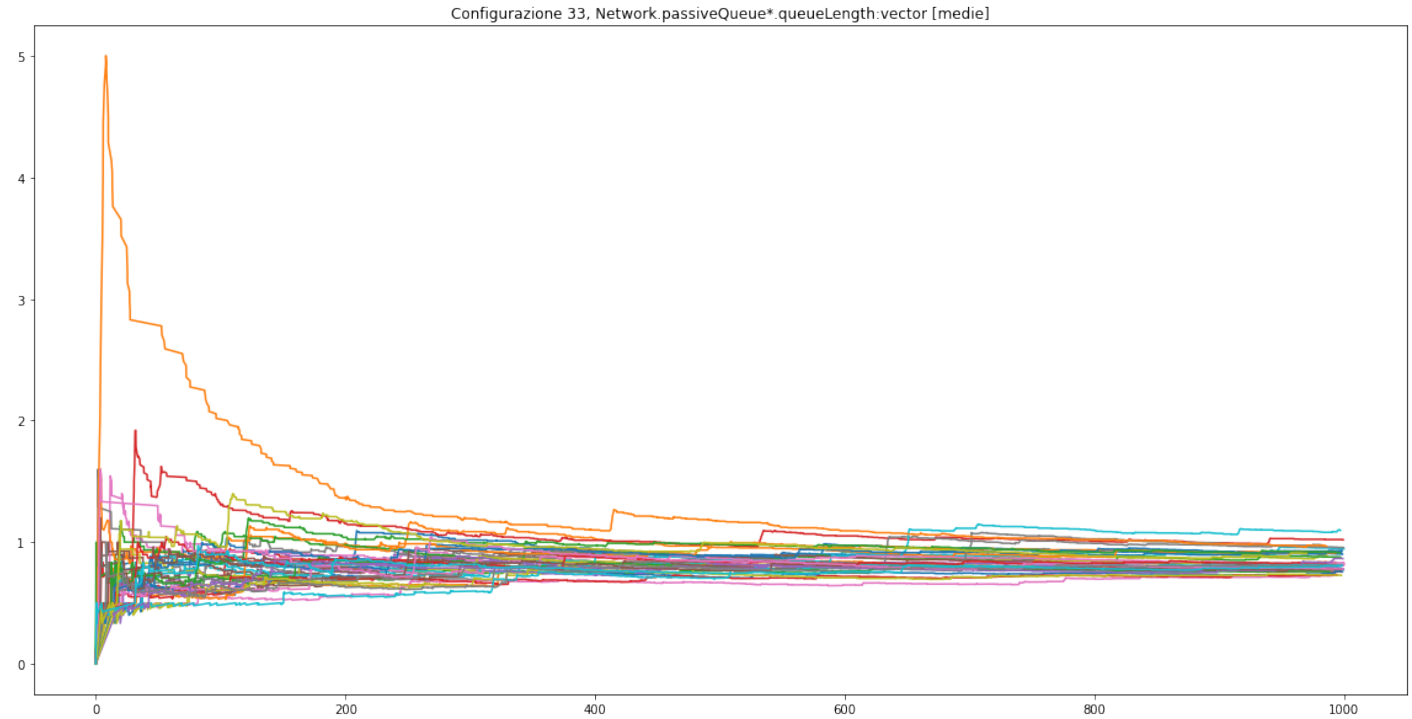

Configurazione 28, Network.passiveQueue*.queueLength:vector [medie]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
200
400
600
800
1000

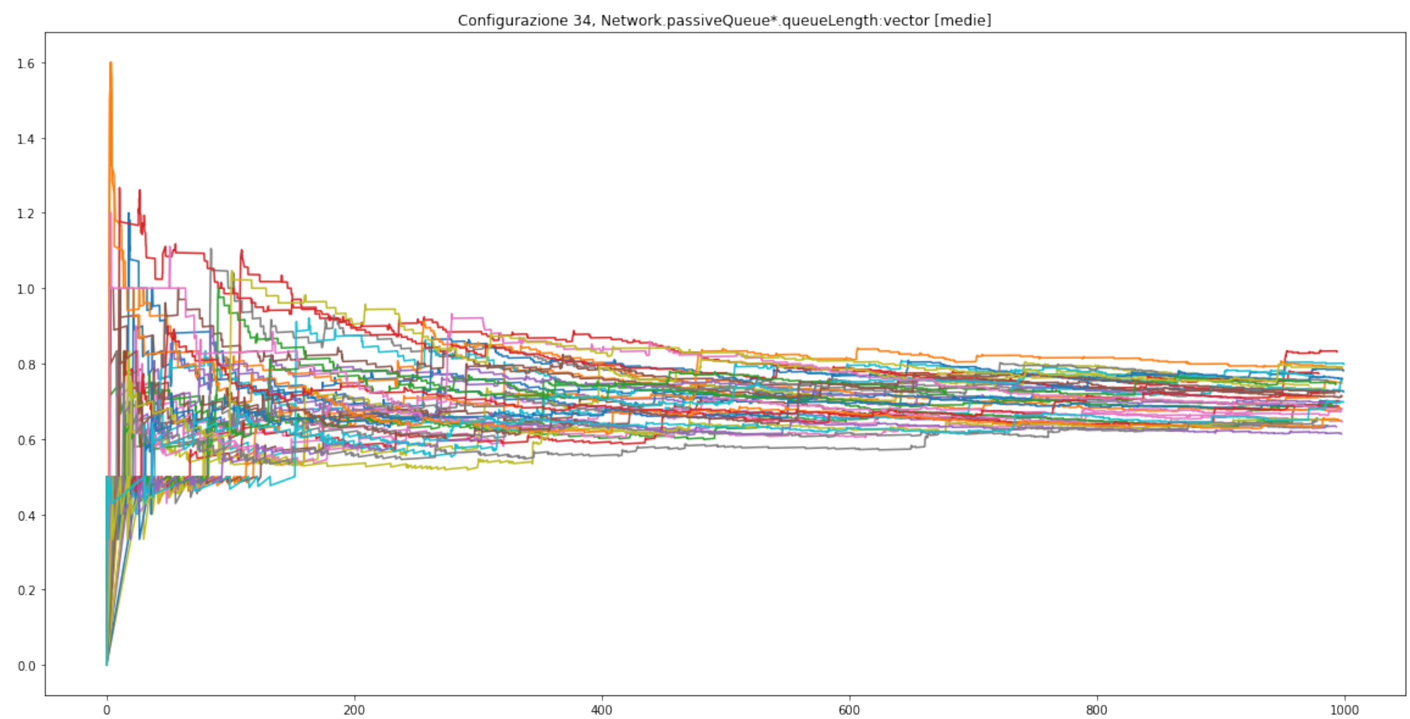

Configurazione 29, Network.passiveQueue*.queueLength:vector [medie]
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000


Configurazione 30, Network.passiveQueue*.queueLength:vector [medie]
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

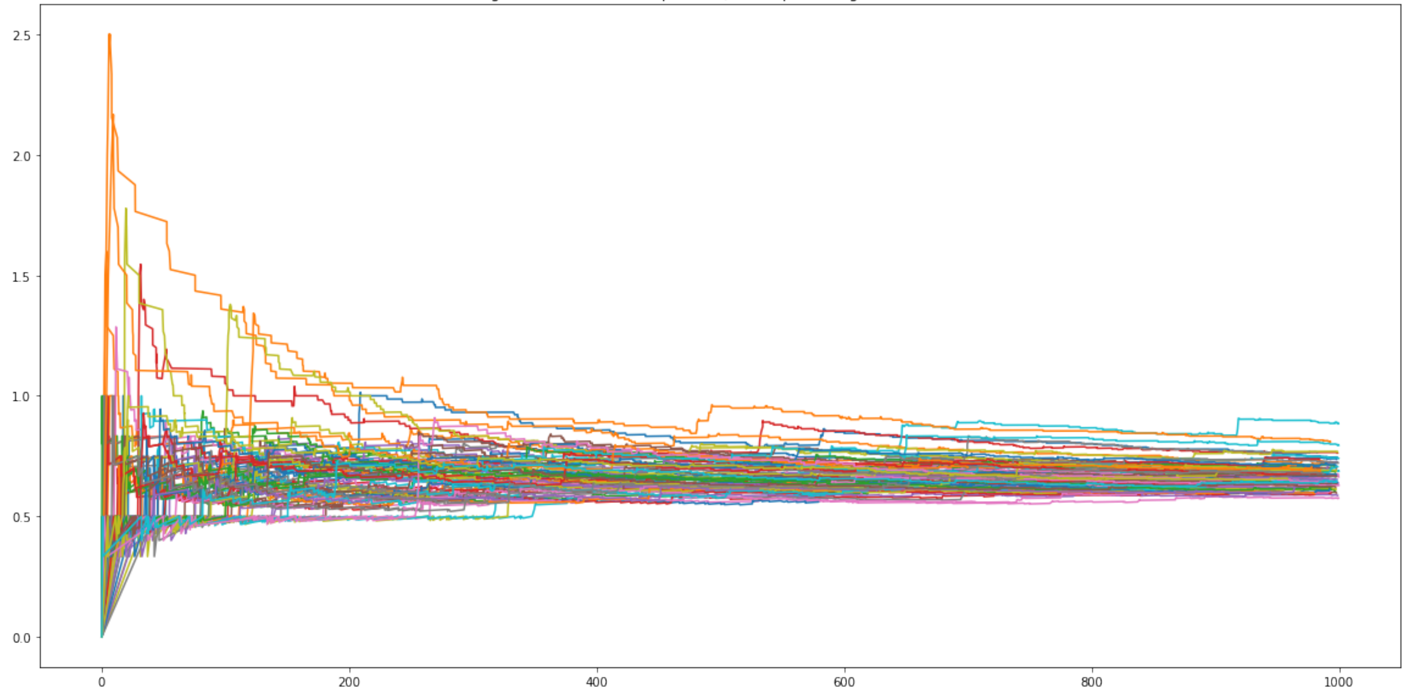

Configurazione 31, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
0
200
400
600
800
1000


Configurazione 32, Network.passiveQueue*.queueLength:vector [medie]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
200
400
600
800
1000

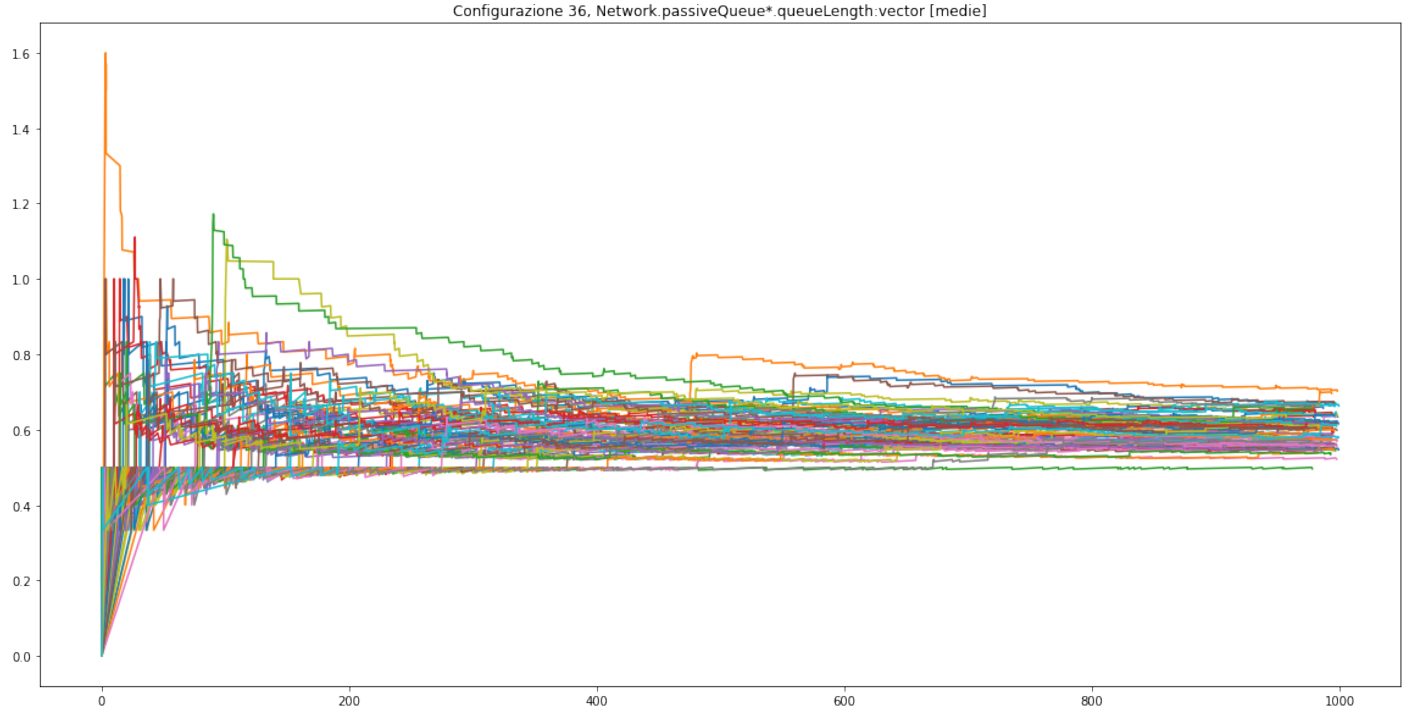

Configurazione 33, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
0
200
400
600
800
1000

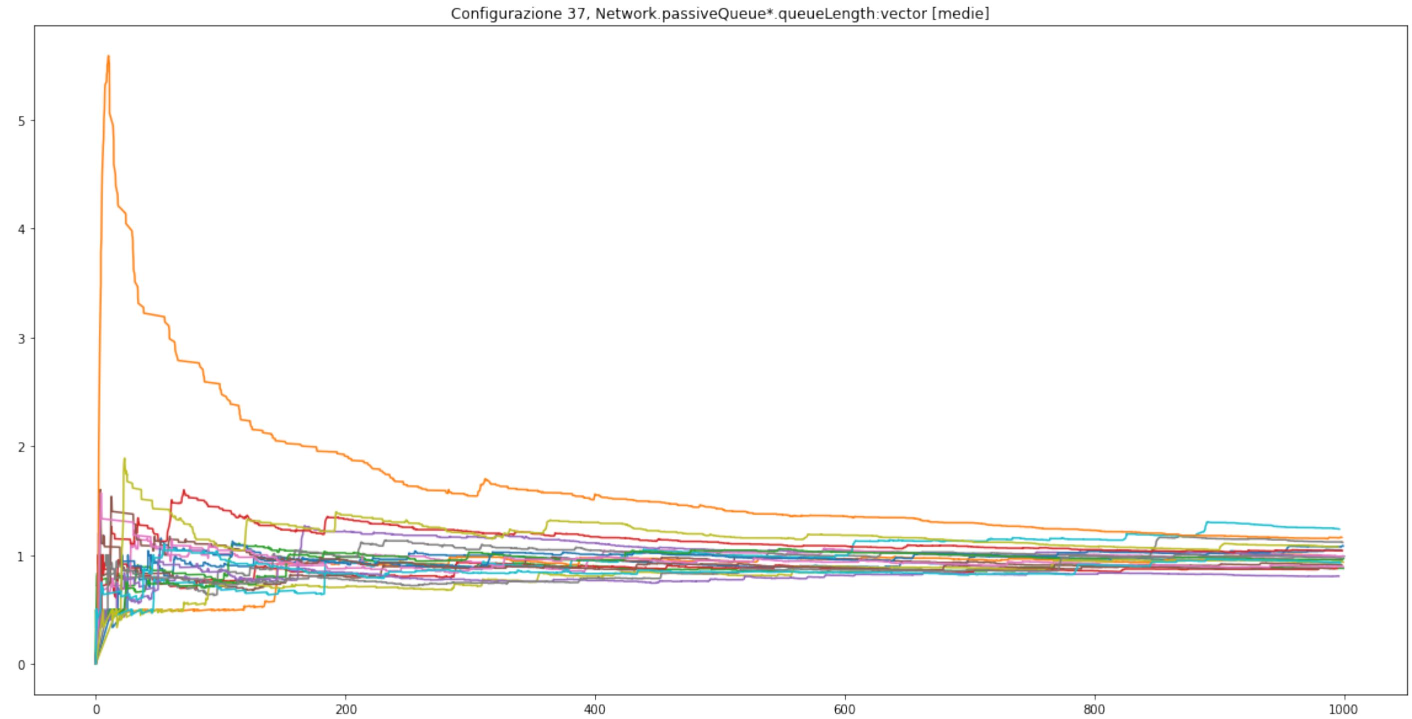

Configurazione 34, Network.passiveQueue*.queueLength:vector [medie]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
200
400
600
800
1000

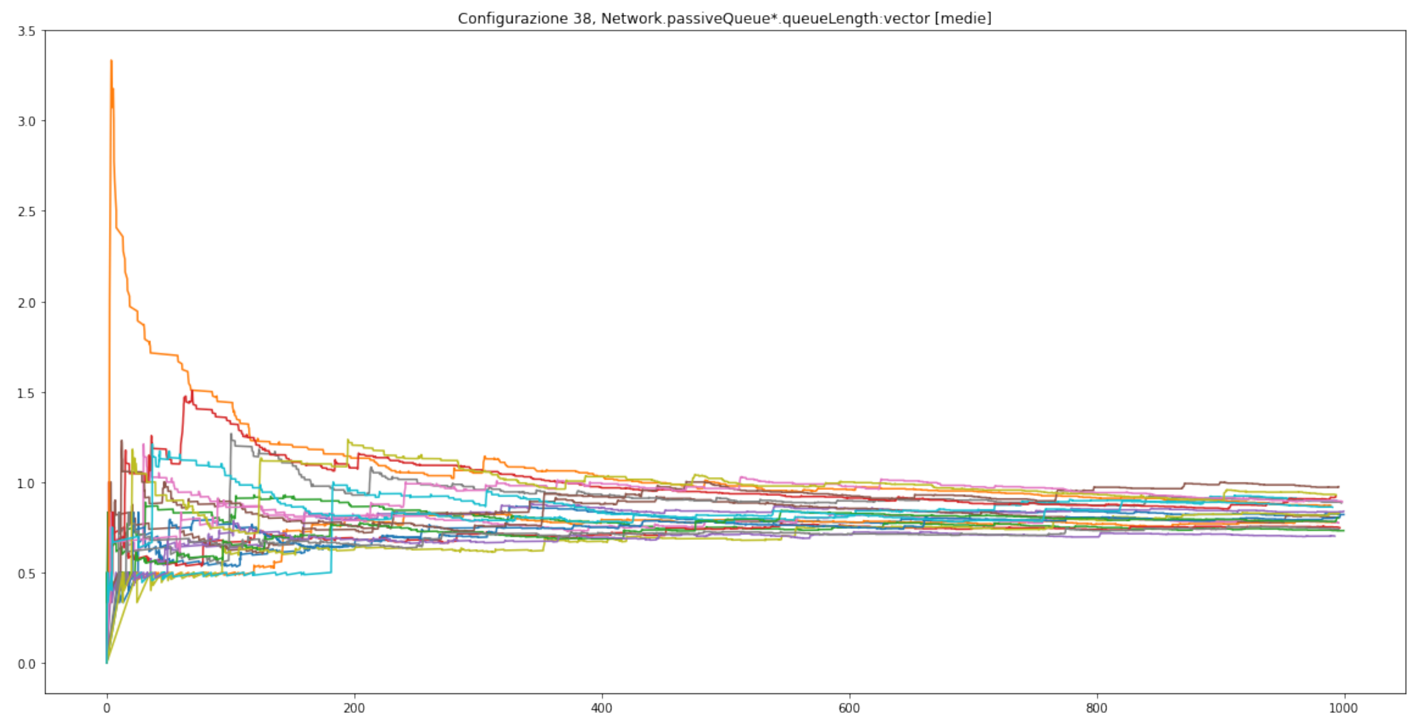
Configurazione 35, Network.passiveQueue*.queueLength:vector [medie]

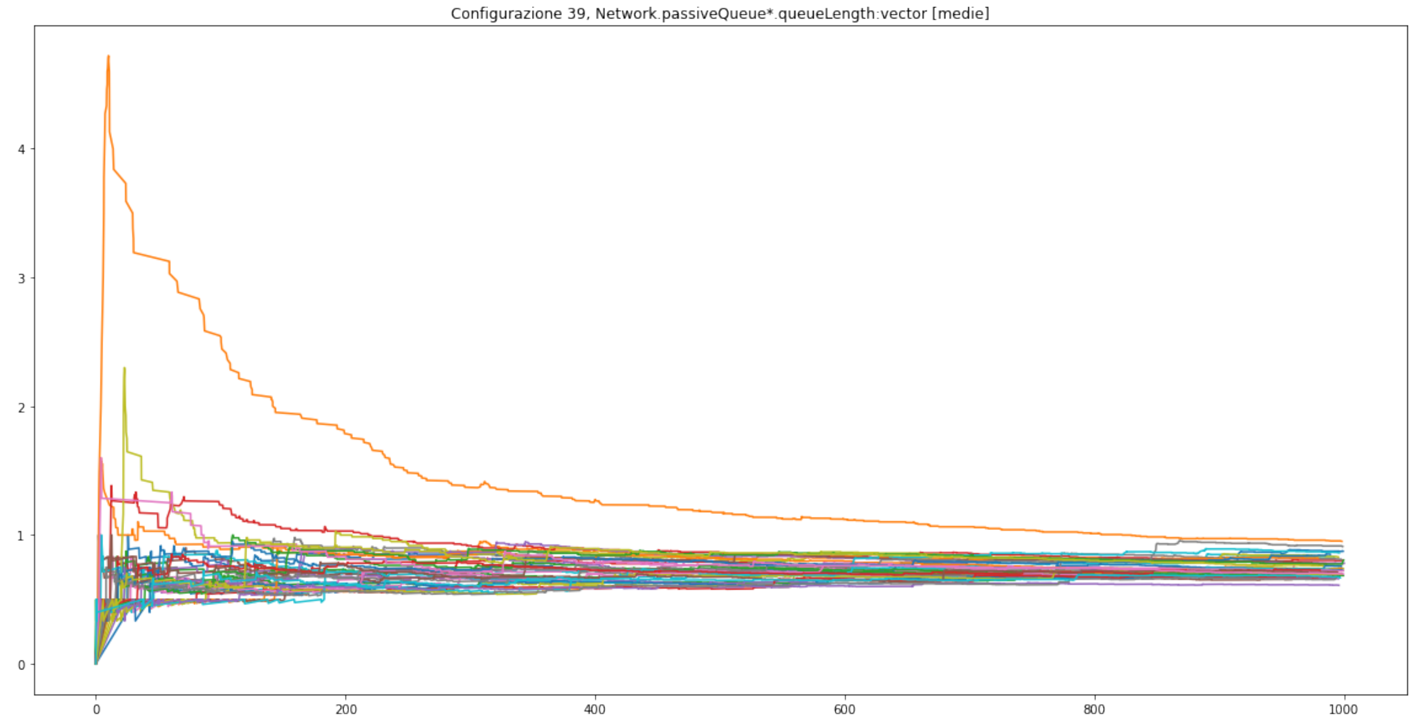


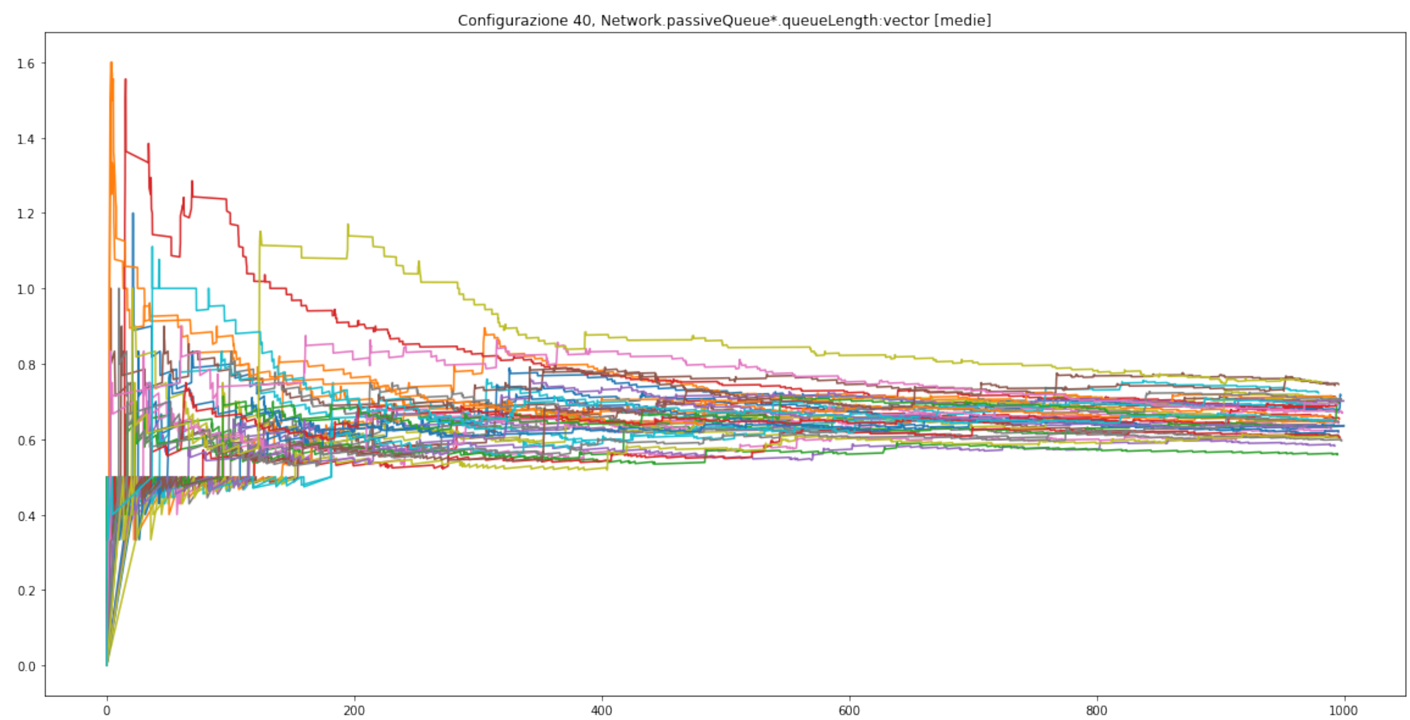
Configurazione 36, Network.passiveQueue*.queueLength:vector [medie]

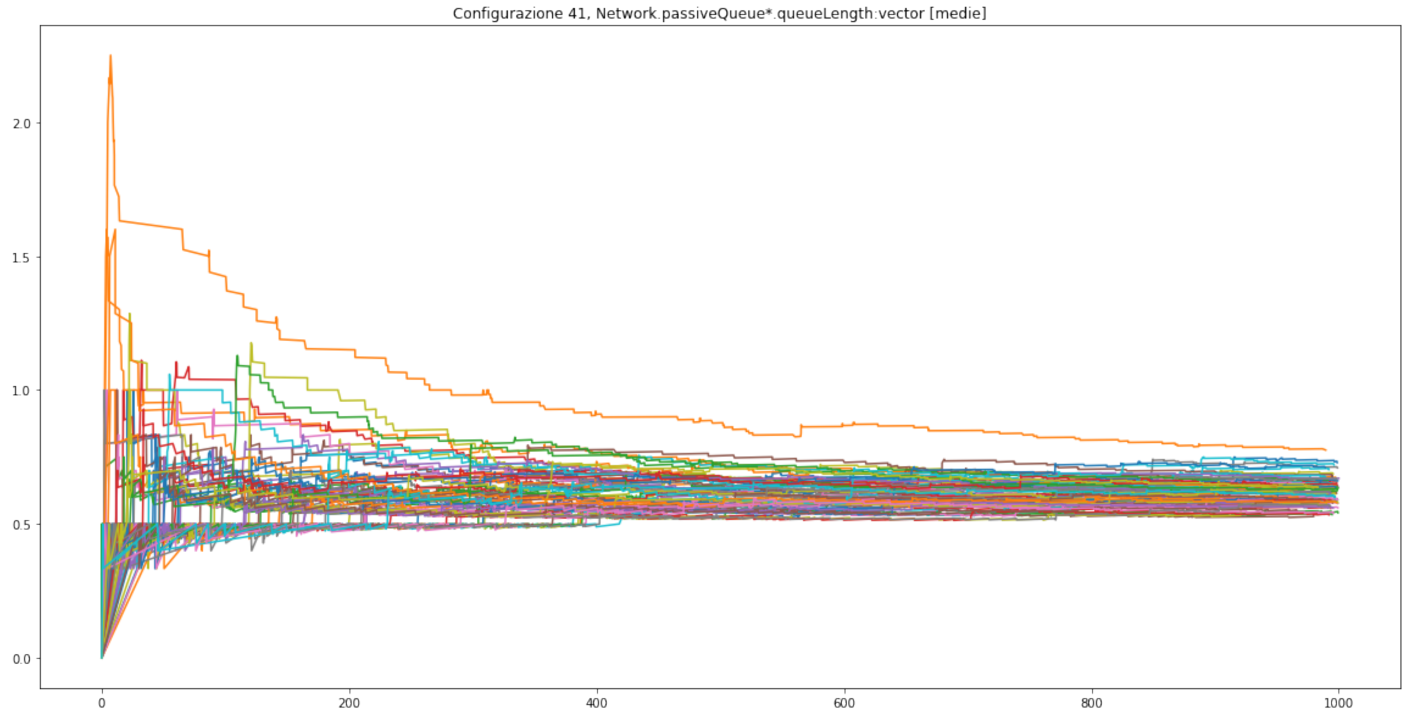


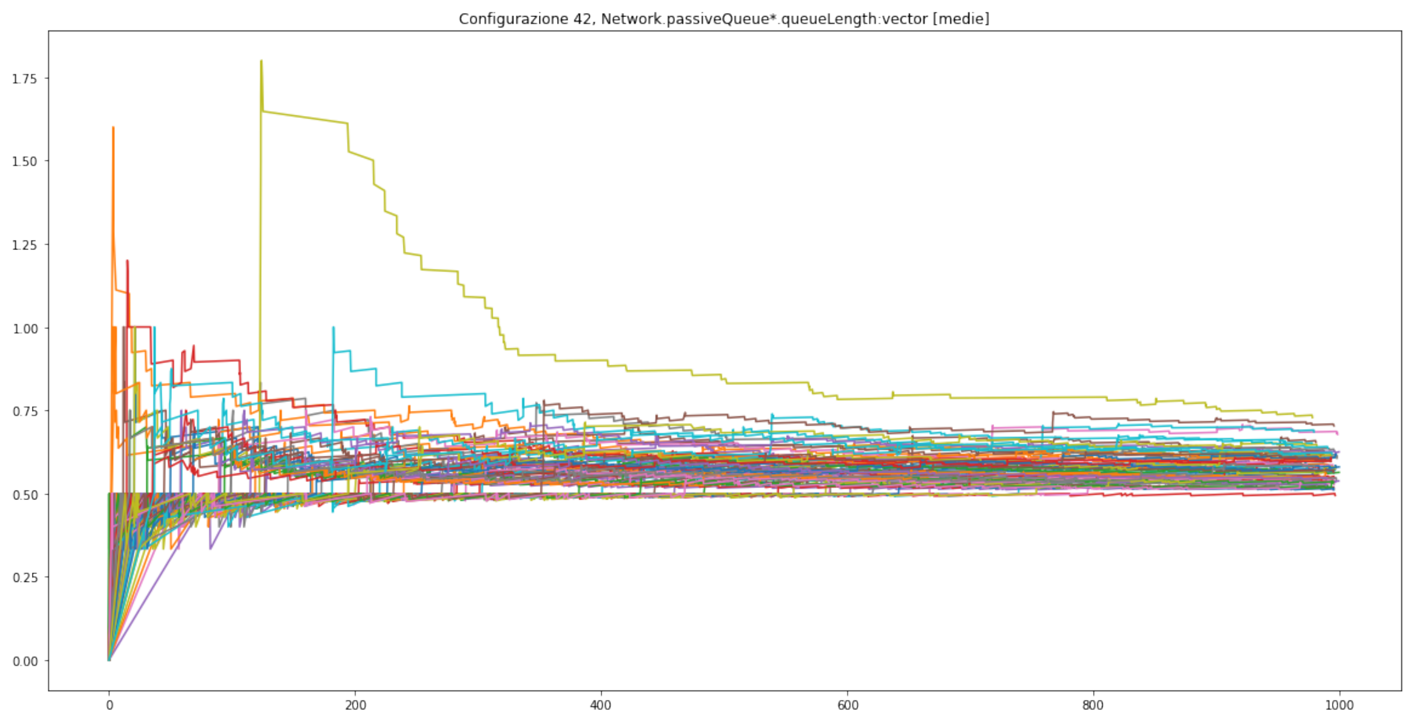

Configurazione 37, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
0
200
400
600
800
1000

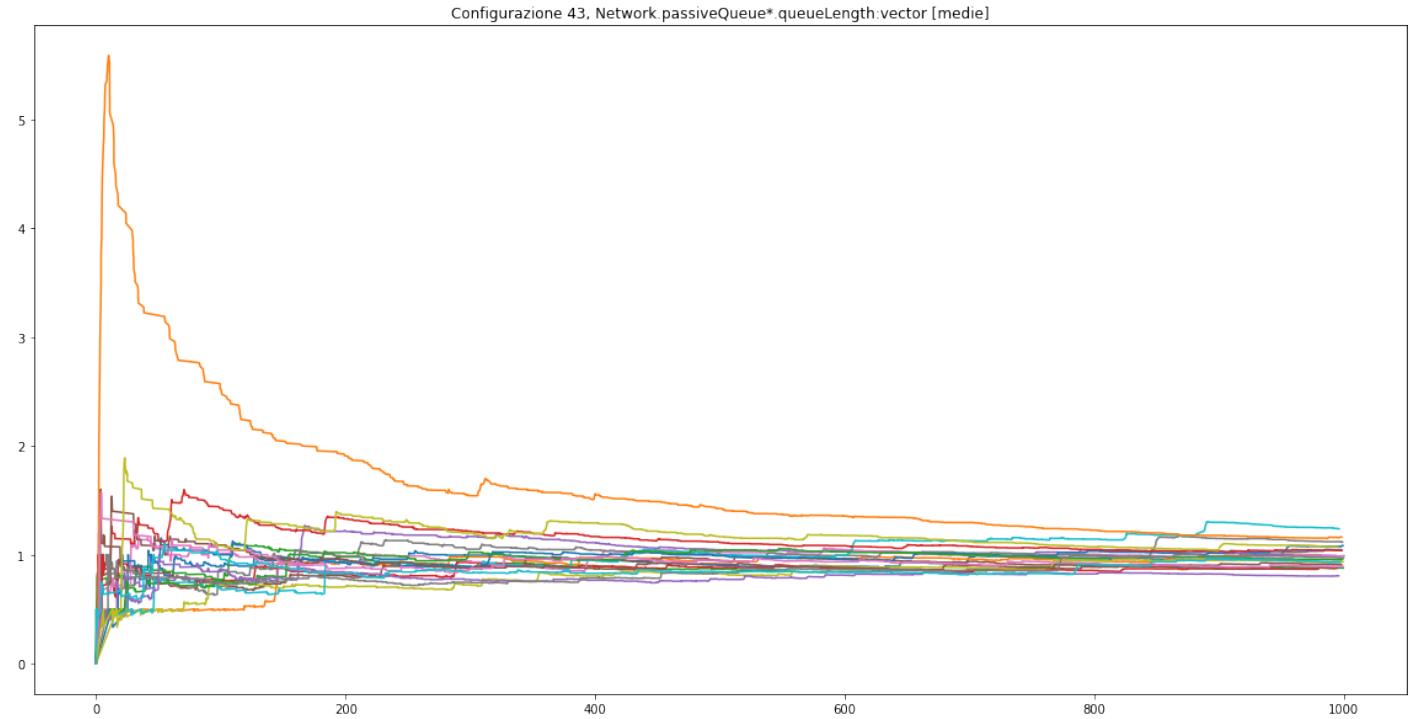

Configurazione 38, Network.passiveQueue*.queueLength:vector [medie]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
200
400
600
800
1000

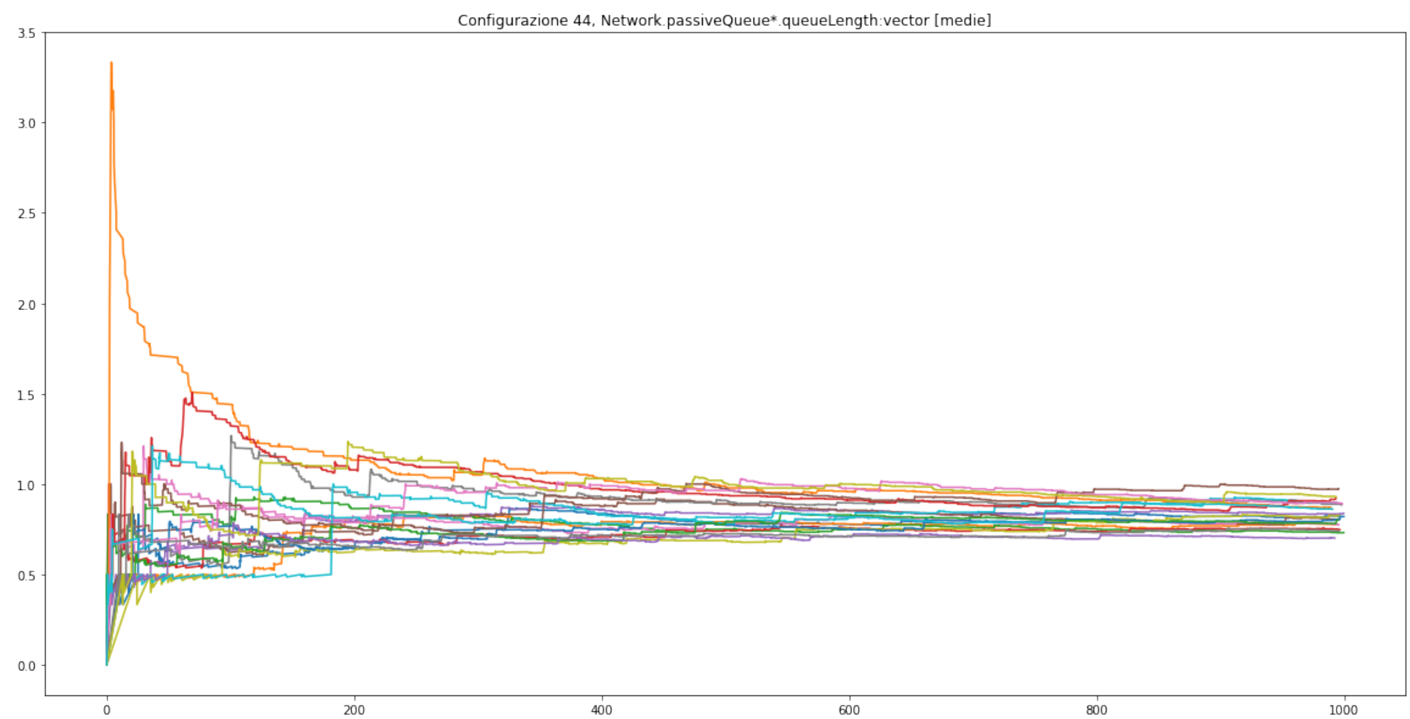

Configurazione 39, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
0
200
400
600
800
1000


Configurazione 40, Network.passiveQueue*.queueLength:vector [medie]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
200
400
600
800
1000

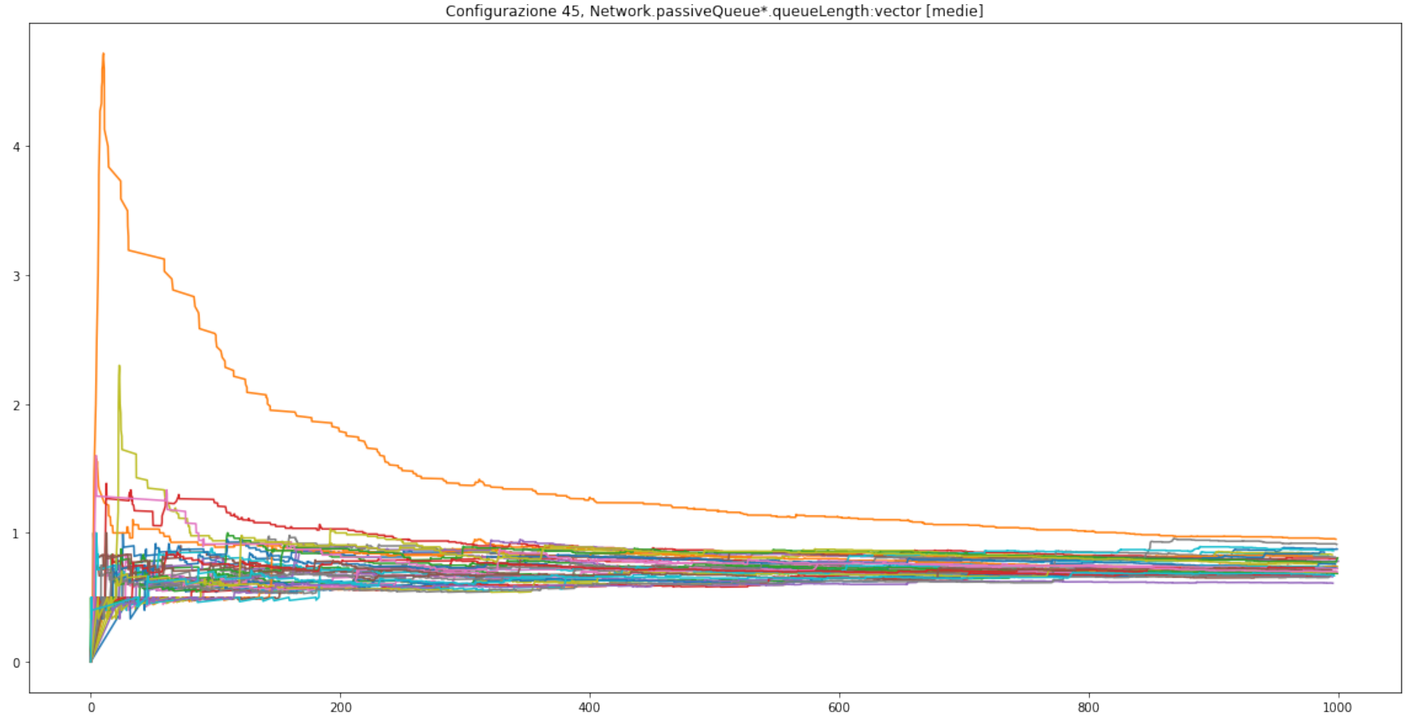

Configurazione 41, Network.passiveQueue*.queueLength:vector [medie]
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000


Configurazione 42, Network.passiveQueue*.queueLength:vector [medie]
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

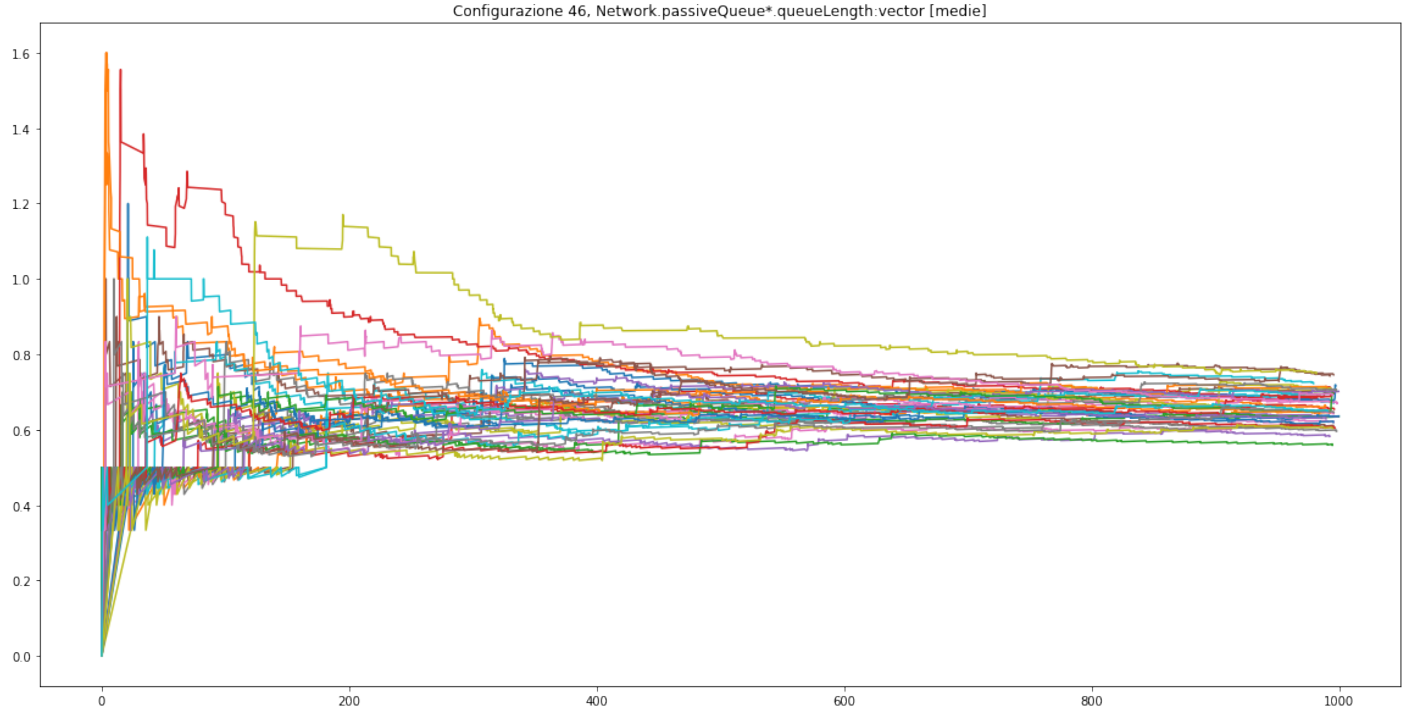

Configurazione 43, Network.passiveQueue*.queueLength:vector [medie]
0
1
2
3
4
5
0
200
400
600
800
1000

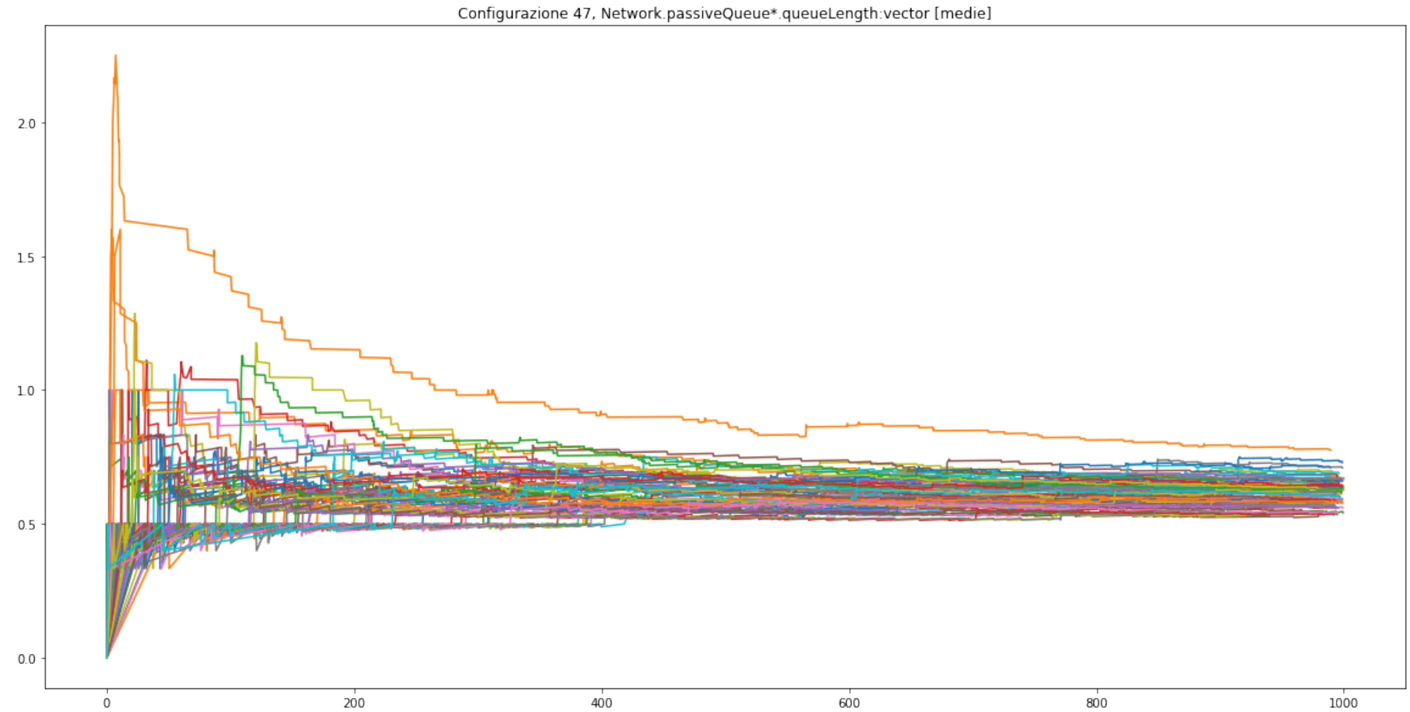

Configurazione 44, Network.passiveQueue*.queueLength:vector [medie]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
200
400
600
800
1000

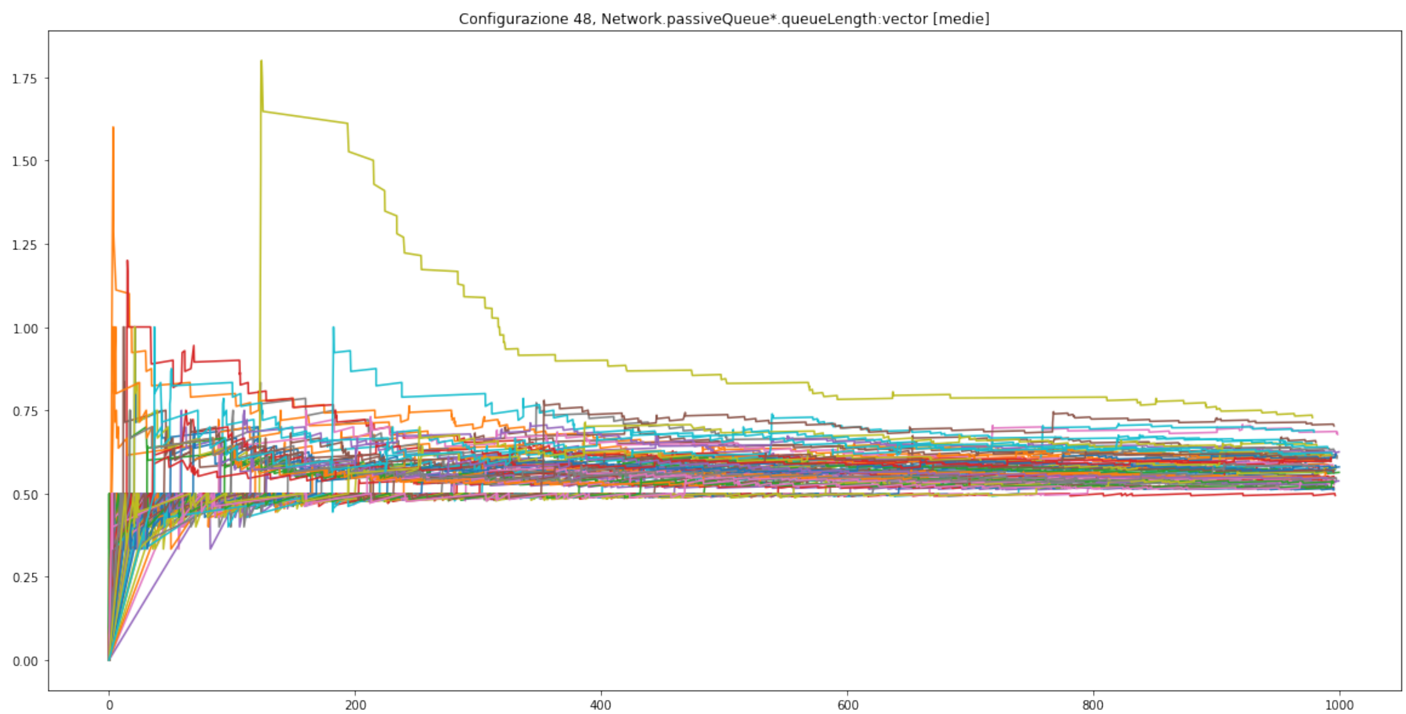
Configurazione 45, Network.passiveQueue*.queueLength:vector [medie]



Configurazione 46, Network.passiveQueue*.queueLength:vector [medie]




Configurazione 47, Network.passiveQueue*.queueLength:vector [medie]
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
1000


Configurazione 48, Network.passiveQueue*.queueLength:vector [medie]
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0
200
400
600
800
1000

References

[1] Sudipta Das, Debasis Sengupta and Lawrence Jenkins, Analysis of an M/M/1+G System Operated under the FCFS Policy with Exact Admission Control, 2012.
http://www.isical.ac.in/~asu/TR/TechRepASU201207.pdf