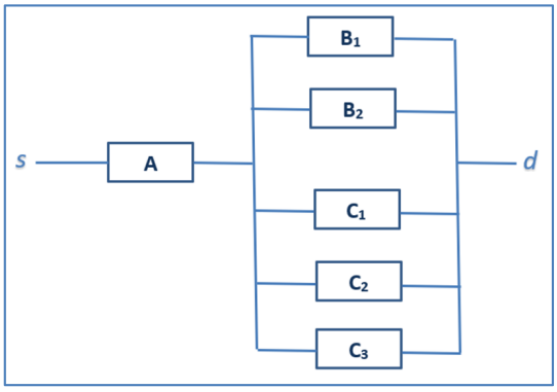
RELAZIONE PROGETTO AMA-CPS (II PARTE, MODEL AND ANALYSIS)

Autori:

- Marco Agatensi 7172615
- Lorenzo Bartolini 7172579

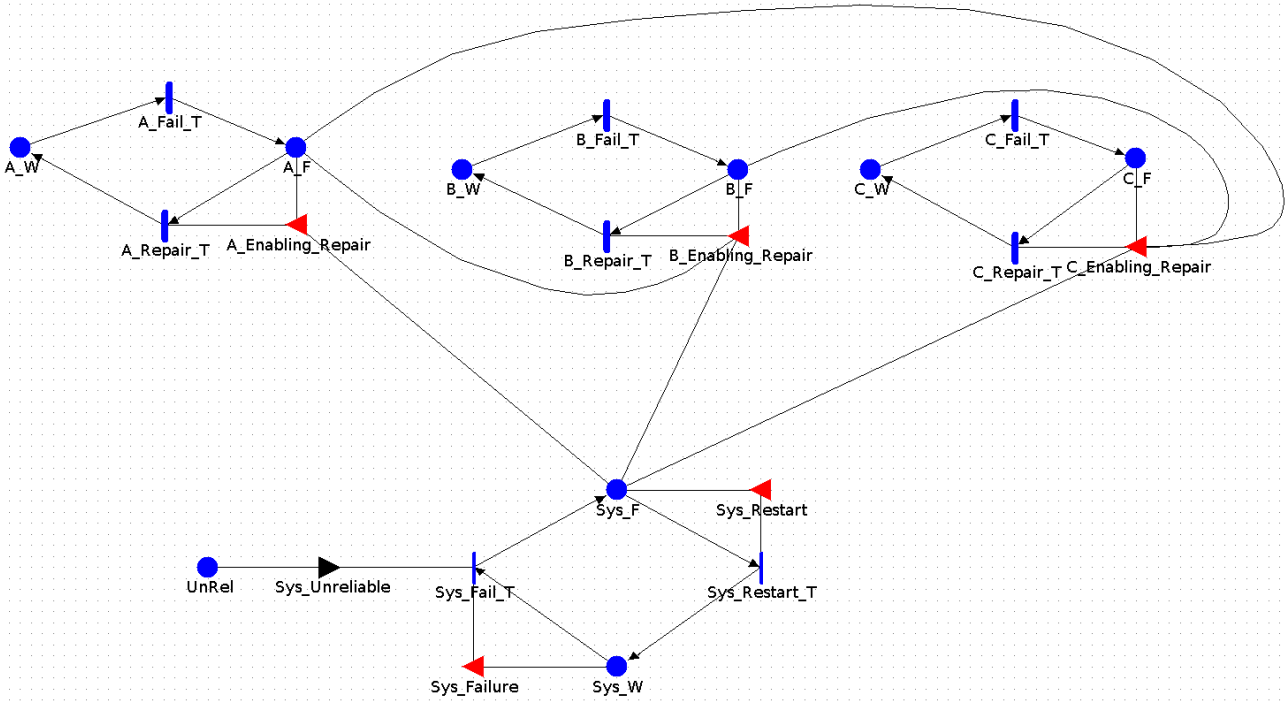


DESCRIZIONE DEL SISTEMA (Opzioni R1, O1, W1)

Il sistema fallisce appena non esiste più un path tra *s* e *d*, altrimenti funziona correttamente. Dopo il fallimento del sistema, i componenti che non erano già falliti, non possono fallire.

Nel nostro sistema, le unità (A, B_i, C_i) possono essere riparate solo quando l'intero sistema è fallito. Inoltre, i componenti devono essere riparati sequenzialmente nel seguente ordine: prima A, poi B_i ed infine C_i. Non appena sono state riparate le unità sufficienti a riottenere un path tra *s* e *d*, il sistema ricomincia a funzionare correttamente e pertanto le riparazioni dei rimanenti componenti falliti verranno interrotte fino al prossimo fallimento del sistema.

STOCHASTIC ACTIVITY NETWORK



DESCRIZIONE DEL COMPORTAMENTO DEL SISTEMA

I componenti delle classi *A*, *B* e *C* sono stati rappresentati ciascuno da due place, uno che rappresenta quelli che stanno funzionando correttamente (<component_class>_W) e l'altro che rappresenta quelli che sono falliti (<component_class>_F). I token nei due place rappresentano il numero di componenti attualmente funzionanti oppure falliti.

Il nostro scenario prevede che la riparazione dei componenti avvenga solo quando l'intero sistema sia fallito; inoltre è necessario imporre un ordine alle riparazioni, prima *A* poi *B* poi *C*.

Per imporre queste condizioni abbiamo utilizzato degli Input gate chiamati *Enabling_Repair*. Questi gate controllano che il sistema sia fallito osservando la presenza di un token nel place *Sys_F*. Gli

input gate di *B* e *C* inoltre controllano che non ci siano componenti falliti in *A_F* per la riparazione dei componenti *B* e in *A_F* e *B_F* per la riparazione dei componenti *C*.
Abbiamo deciso di modellare l'intero sistema, nella porzione inferiore della SAN, utilizzando due place per indicare che l'intero sistema sia funzionante oppure fallito. La transizione da uno stato all'altro è effettuata con delle transizioni istantanee abilitate da input gate. Questi input gate verificano la presenza o assenza di un path da *s* a *d* come da specifica del sistema.
Di particolare interesse è il place *UnRel* che rappresenta che il sistema sia fallito almeno una volta. Il token viene inserito nel place tramite un Output Gate attivato dalla transizione *Sys_Fail_T*.

DEFINIZIONE DELLE REWARD VARIABLES

Reliability:
Metrica che rappresenta la probabilità che il sistema funzioni in maniera continuativa, ovvero senza mai fallire, nell'intervallo [0, t].
Per ottenere queste probabilità abbiamo definito la reward variable come Instant of Time specificando il Rate Reward analizzando la presenza di token nel place UnRel descritto in precedenza.
Funzione: return (1 - CriticalSystemAgatensiBartolini->UnRel->Mark());
Possiamo osservare che se al tempo t non c'è nessun token in UnRel, la reward variable vale 1 significando che il sistema è Reliable perché nell'intervallo [0, t] il sistema non è mai fallito.
Availability (steady state):
Metrica che rappresenta la probabilità che il sistema sia in funzione facendo tendere t all'infinito.
Per ottenere questa probabilità abbiamo utilizzato una reward variable Steady State andando a specificare il Rate Reward come il marking del place Sys_W.
Funzione: return CriticalSystemAgatensiBartolini->Sys_W->Mark();

Possiamo osservare come queste reward variables siano molto semplici nella definizione grazie al lavoro di modellazione fatto nella SAN creando una sezione del modello il cui compito è quello di rappresentare l'intero sistema. In questo modo la definizione delle reward variable è semplice e immediata.

SOLVER UTILIZZATI

Transient Solver:
Risolutore analitico utilizzato per il calcolo della Reliability.
I parametri che abbiamo inserito nel solver sono: Accuracy=9; Verbosity=0;
Direct Steady State Solver:
Risolutore analitico utilizzato per il calcolo della Availability a Steady State.
I parametri che abbiamo utilizzato sono: Stopping Criterion=9; Rows=0; Stability=0; Tolerance=0.001; Verbosity=0;

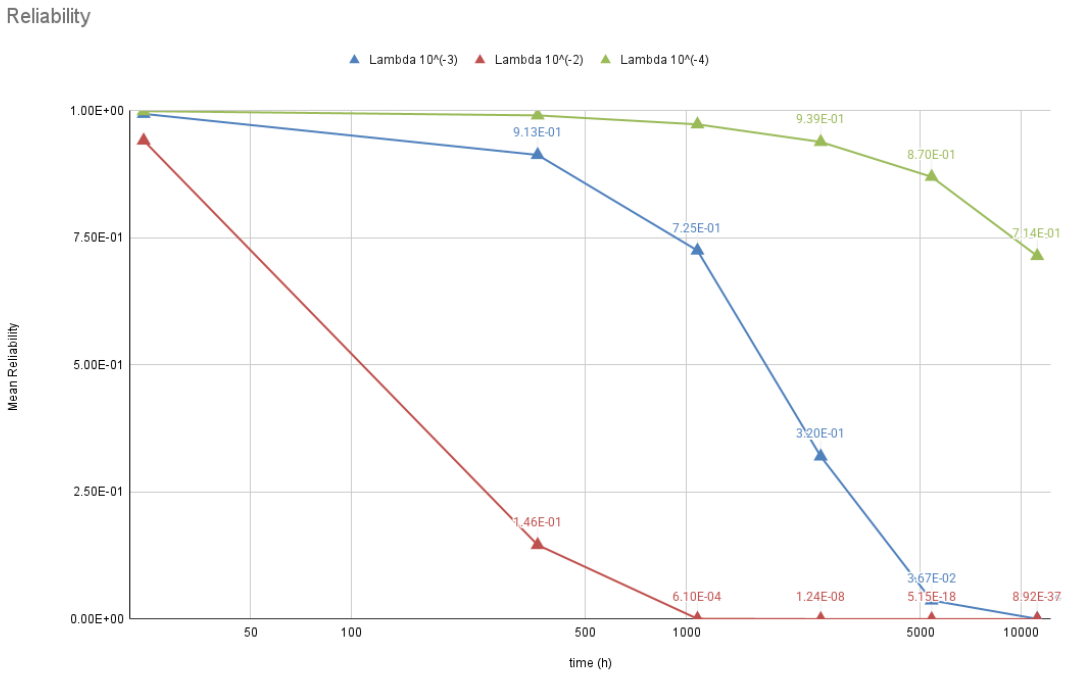
RISULTATI E CONCLUSIONI

Parametri utilizzati

-> lambda = 10^(-2), 10^(-3), 10^(-4) (fallimenti all'ora)
-> mu=10^(-2), 10^(-3), 10^(-4) (riparazioni all'ora)
-> t=1, 15, 45, 105, 225, 465 (giorni)

NB: Abbiamo scelto di utilizzare le ore come sistema temporale di riferimento, nelle reward variables abbiamo quindi convertito *t* giorni in *24*t* ore.

Reliability:



lambda1 = 10^(-2)			
time (h)	mu1 = 10^(-2)	mu2 = 10^(-3)	mu3 = 10^(-4)
24	9.42E-01	9.42E-01	9.42E-01
360	1.46E-01	1.46E-01	1.46E-01
1080	6.10E-04	6.10E-04	6.10E-04
2520	1.24E-08	1.24E-08	1.24E-08
5400	5.15E-18	5.15E-18	5.15E-18
11160	8.92E-37	8.92E-37	8.92E-37

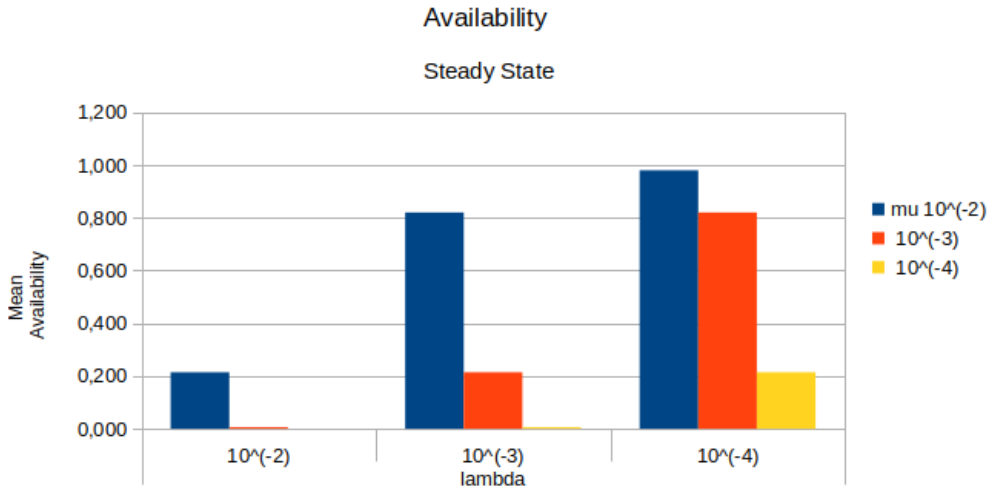
lambda2 = 10^(-3)			
time (h)	mu1 = 10^(-2)	mu2 = 10^(-3)	mu3 = 10^(-4)
24	9.94E-01	9.94E-01	9.94E-01
360	9.13E-01	9.13E-01	9.13E-01
1080	7.25E-01	7.25E-01	7.25E-01
2520	3.20E-01	3.20E-01	3.20E-01
5400	3.67E-02	3.67E-02	3.67E-02
11160	4.65E-04	4.65E-04	4.65E-04

lambda3 = 10^(-4)			
time (h)	mu1 = 10^(-2)	mu2 = 10^(-3)	mu3 = 10^(-4)
24	9.99E-01	9.99E-01	9.99E-01
360	9.91E-01	9.91E-01	9.91E-01
1080	9.73E-01	9.73E-01	9.73E-01
2520	9.39E-01	9.39E-01	9.39E-01
5400	8.70E-01	8.70E-01	8.70E-01
11160	7.14E-01	7.14E-01	7.14E-01

Come è possibile vedere dalle tabelle, il parametro mu legato alle riparazioni dei componenti non influisce nella reliability in quanto il nostro sistema può iniziare a riparare solo quando il sistema è già fallito. Di conseguenza mu entra in gioco quando ormai il sistema è fallito e quindi non ne modifica la reliability.
Dal grafico possiamo osservare come, al crescere del tempo, la reliability diminuisca. Inoltre le modifiche di lambda sono perfettamente rispecchiate nell'andamento della reliability, con lambda piccolo la reliability è più alta rispetto a lambda grandi.

Per migliorare la reliability del sistema è necessario lavorare profondamente sul tasso di fallimento dei componenti e meno sul migliorare il rate di riparazioni in quanto non influenza la reliability. Possiamo osservare come ci sia un calo drastico della reliability passando da lambda=10^(-4) a lambda=10^(-3). Questo può essere un indizio per i progettisti del sistema nell'indirizzare maggiori risorse per abbassare lambda ottenendo un sistema più reliable.

Availability (Steady State):



		mu		
	x	10^(-2)	10^(-3)	10^(-4)
lambda	10^(-2)	2.143E-01	5.768E-03	6.568E-05
	10^(-3)	8.202E-01	2.143E-01	5.768E-03
	10^(-4)	9.802E-01	8.202E-01	2.143E-01

Dal grafico osserviamo l'Availability a steady state, il colore rappresenta il parametro mu e l'asse delle X rappresenta il parametro lambda.
Osserviamo che aumentando il rate di riparazione dei componenti l'availability migliora sensibilmente. Anche lambda ha un effetto molto importante, infatti per lambda molto piccoli si verificano meno fallimenti e l'availability migliora molto soprattutto in caso di riparazioni meno frequenti.

Per migliorare l'availability del sistema è sicuramente molto più importante lavorare sul rate di riparazione dei componenti pur mantenendo un rate di fallimenti contenuto. Ad esempio con lambda=10^(-4) possiamo accettare verosimilmente anche mu=10^(-3); invece per mu=10^(-2) possiamo accettare lambda=10^(-3). Questo può essere d'aiuto ai progettisti del sistema per capire dove è meglio indirizzare le risorse con l'obiettivo di ottenere un sistema migliore.