[21/22] 38092-MOD2 - RETI DI TELECOMUNICAZIONE

Maffeis Isaac 1041473

Relazione di progetto: Sim. C++ C6

1. Introduzione

L'elaborato proposto riguarda il progetto C6 del simulatore di un sistema a coda C++ :

- Si consideri un sistema a coda con un servente e K posti in coda. I tempi di interarrivo siano una variabile casuale uniforme tra a e b e ogni arrivo sia composto da un numero x di utenti, dove x è una v.c. Binomiale con valor medio C. I tempi di servizio siano esponenziali negativi con valor medio T. Si determini la probabilità di rifiuto e il ritardo medio di attraversamento. Il software accetta in input i parametri K, a, b, T e C.

Il progetto è stato realizzato tramite l' IDE Microsoft Visual studio 2022 in linguaggio C++ e caricato su GitHub all'indirizzo: https://github.com/Isaacmaffo96/simqueue_2022 .

È presente anche un file excel nel quale sono collezionati i dati relativi ai vari test effettuati.

2. Sviluppo

2.1 Modifiche

Il progetto si discosta dal codice del simulatore originale in quanto si richiede una coda limitata K (K parametro input), una distribuzione per i tempi di interarrivo continua tra a e b (a, b parametri input) e l'arrivo a lotti composti da x utenti con x v.c. Binomiale con valor medio C (C parametro input), inoltre viene richiesto di determinare la probabilità di rifiuto.

Per attuare le modifiche richieste sono stati modificati i seguenti file:

- buffer.cpp
- buffer.h
- easyio.cpp
- easyio.h
- event.cpp
- event.h
- queue.cpp
- queue.h
- rand.h

Con ordine vengono commentate di seguito le modifiche.

2.2 Buffer a coda limitata K

Per implementare un buffer limitato da un parametro K viene inizialmente chiesto all'utente questo parametro, tramite la funzione read_int della classe easyio.h (Valore di default inf. ,valore minimo 0, valore massimo inf.), il risultato viene salvato come parametro intero Kqueue della classe buffer.h. Inoltre vengono definite le variabili interi count e tot_lost, sempre nella classe buffer.h che contano

rispettivamente il numero di pacchetti nel buffer e il totale dei pacchetti persi dal buffer. Nell'implementazione della classe buffer.cpp viene definito il modo in cui il buffer scarta i pacchetti, in particolare ogni volta che avviene un inserimento nel buffer viene incrementata la variabile count, mentre ogni volta che viene rimosso un pacchetto viene decrementata. Quando il valore di questa variabile supera il valore di Kqueue il pacchetto verrà scartato senza essere fatto entrare nel sistema.

2.3 Distribuzione continua uniforme

La distribuzione continua uniforme tra due parametri a e b è implementata grazie all'apposito costrutto GEN_UNIF nella classe rand.h , i parametri a e b (min e max) vengono chiesti all'utente e di default sono impostati sui valori interi 1 e 2.

Questa nuova distribuzione non sostituisce quella già esistente, ma ne offre un'alternativa, l'utente può selezionare quale distribuzione desidera per i tempi di interarrivo. Per fare ciò si è aggiunto al codice delle classi queue.cpp e event.cpp uno switch case con le 2 distribuzioni

2.4 Arrivo a lotti (batch)

L'arrivo a lotti degli utenti è stato implementato nella classe event.cpp, in particolare nella funzione arrival::body(), che gestisce il corpo di un arrivo. Nello specifico, dopo aver determinato la lunghezza del lotto e assegnata alla variabile batch_size, questa variabile verrà decrementata ogni volta che arriva la richiesta di generazione di un evento di arrivo, fino ad arrivare ad 1, tutti gli arrivi generati fino a quel momento verranno creati con lo stesso tempo di arrivo, così da simulare un arrivo a gruppo di eventi. Una volta che batch_size viene decrementata a 1 viene generato un nuovo lotto si ripete la procedura.

2.5 Distribuzione Binomiale

Per prima cosa viene chiesto all'utente il valor medio C della v.c. x Binomiale, poi la distribuzione viene generata nella classe event.cpp grazie alla classe binomial_distribution del namespace std, che riceve come parametri il valore massimo della distribuzione e la probabilità di successo.

3. Analisi dei risultati

3.1 Dati di Test

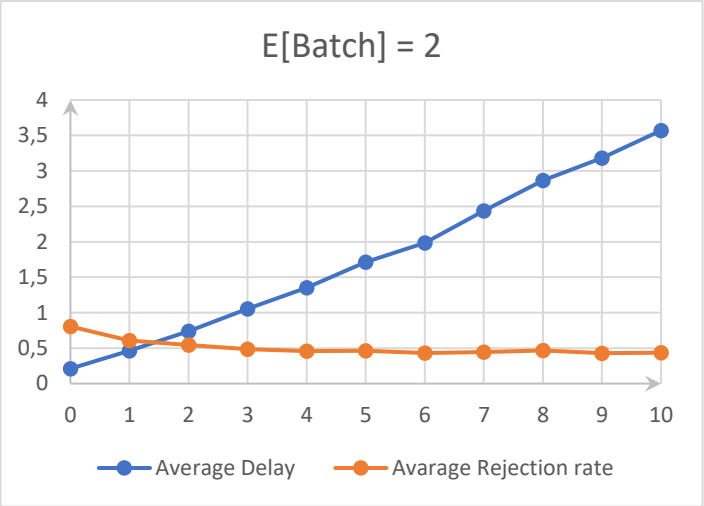
Sono state svolte 55 simulazioni, in particolare si son tenuti tutti i valori del simulatore di default, così come i parametri a,b (1,2) e il tempo di servizio T (0,4). Si sono modificati invece i valori della coda K e del valor medio del lotto C.

I risultati di queste simulazioni sono stati conservati in un file excel, dove sono presenti anche grafici per facilitarne la comprensione.

3.2 Analisi coda buffer K

Per prima cosa si analizza il comportamento del buffer a coda limitata al variare del parametro K, considerando fisso il valor medio dei lotti pari a 2.
Si effettuano 11 misurazioni del ritardo medio sperimentato e del tasso medio di rifiuto, con K che varia da 0 a 10.

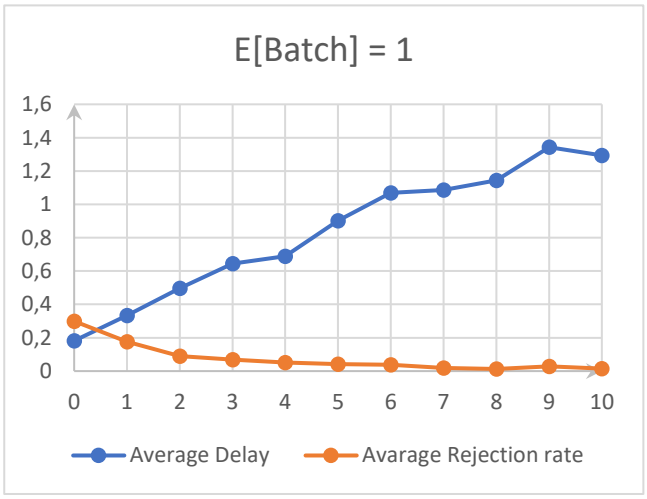
K	Average Delay	Avarage Rejection rate
0	0,211051	0,806148
1	0,462819	0,606484
2	0,740776	0,544302
3	1,055093	0,48702
4	1,351708	0,458807
5	1,713199	0,462639
6	1,983983	0,432066
7	2,437889	0,444439
8	2,866506	0,466625
9	3,182409	0,429268
10	3,569791	0,435234

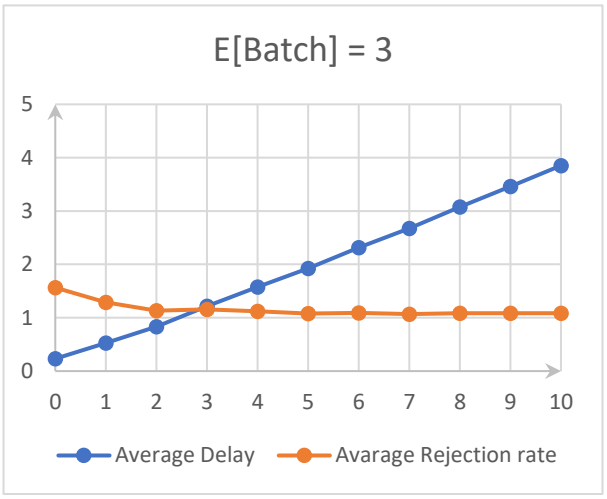


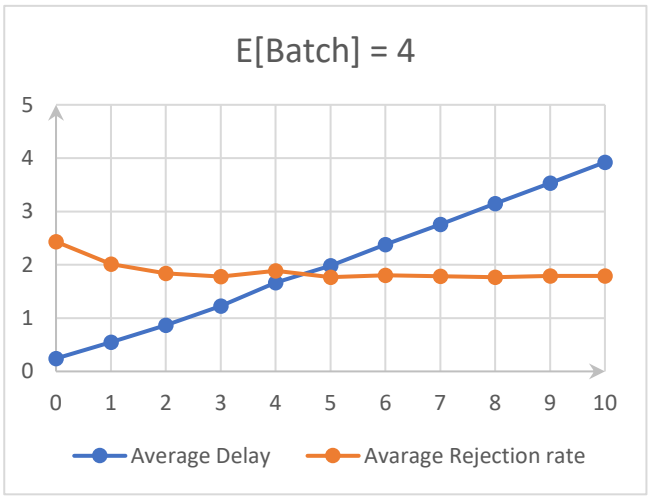
I dati mostrati in tabella o graficamente dimostrano come all'aumentare della lunghezza della coda il ritardo medio sperimentato aumenta in modo lineare, mentre il tasso medio di rifiuto diminuisce all'aumentare della coda fino a convergere attorno a un certo valore.

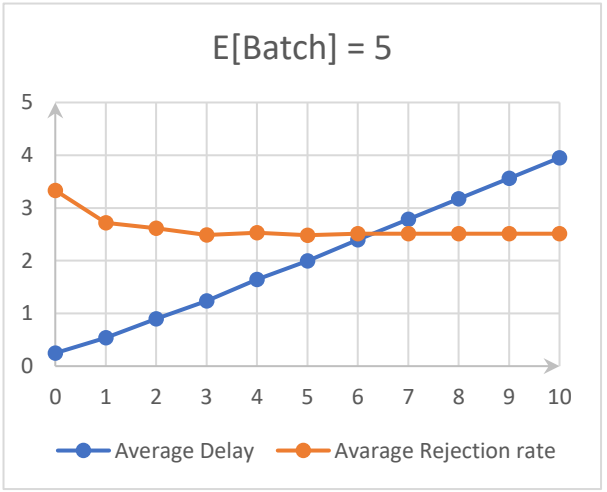
3.3 Analisi del valor medio C

L'esperimento appena mostrato viene ripetuto con il parametro C del valor medio della lunghezza dei lotti che varia da 1 a 5.





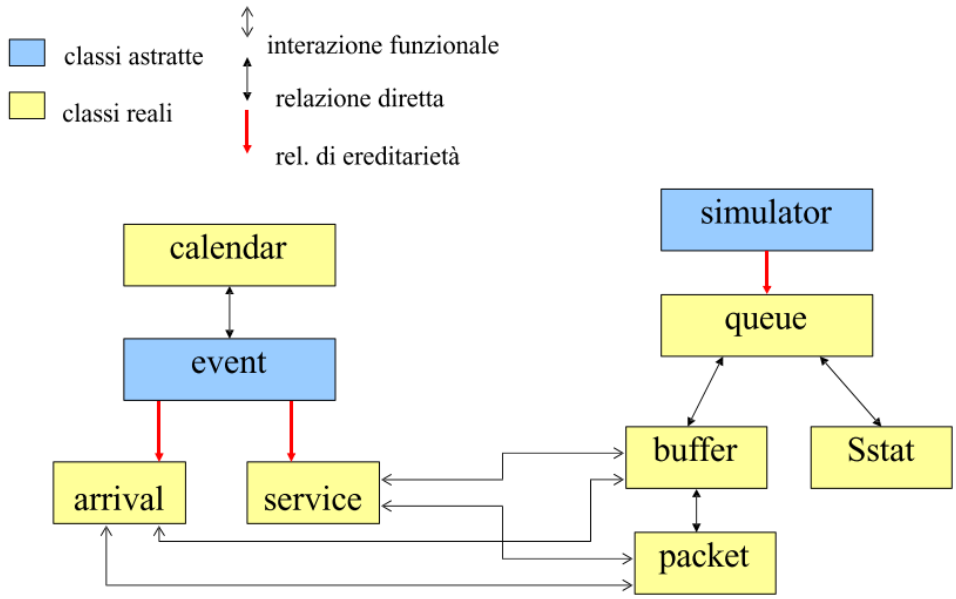




Tramite l'analisi grafica è evidente come all'aumentare del parametro C le prestazioni generali del sistema diminuiscano, il ritardo medio sperimentato peggiora e aumenta in modo rapido, per poi converge con le prove 3,4 e 5. Anche il tasso medio di rifiuto aumenta con l'aumentare di C, anche se poi segue sempre la sua curva di decrescita e convergenza all'aumentare della coda.

4. Struttura

La struttura delle classi non è stata alterata, mantenendo le relazioni originarie, non sono state aggiunte nuove classi, ma implementato codice all'interno di quelle già esistenti.



5. Esempio di funzionamento

Tornando al caso proposto in dettaglio precedentemente, ossia l'analisi della coda limitata da K, con valor medio dei lotti C fissato a 2, viene mostrato un esempio di simulazione.

5.1 Fase di input

```
MODEL PARAMETERS:
Buffer queue limit: K =  [2147483647] > 2
Arrivals model:
1 - Poisson:>
2 - Continuous Probability Distributions:>
 [2] > 2
Batch Arrivals?>:
0 - No:
1 - Yes:
 [1] > 1
Mean batch size value:>
C =  [2] > 2
min value a: [1] > 1
max value b: [2] > 2
 Service model:
1 - Exponential:>
 [1] > 1
Average service duration (s) [0.4000] > 0.4
SIMULATION PARAMETERS:
Simulation transient len (s) [100.0000] > 100
Simulation RUN len (s) [100.0000] > 100
Simulation number of RUNs [5] > 5
```

In questa fase vengono chiesti all'utente i parametri di input, per il progetto proposto sono stati aggiunti i parametri K, a, b, C, è stata aggiunta la scelta tra la distribuzione per i tempi di interarrivo e la scelta di avere gli arrivi a lotti.

Per quanto riguarda invece i tempi di servizio e i parametri della simulazione è rimasto tutto invariato e si usano i valori proposti a default.

Il valore proposto a K di default è il valore massimo che può avere un intero, per simulare una coda infinita.

5.2 File di trace

```
*******************************************
          TRACE RUN 1
*******************************************
Average Delay              0.866254  +/- inf  p:inf
Average Rejection rate     0.726852  +/- inf  p:inf
*******************************************
          TRACE RUN 2
*******************************************
Average Delay              0.865512  +/- 9.42e-03  p:1.09
Average Rejection rate     0.677646  +/- 6.25e-01  p:92.26
*******************************************
          TRACE RUN 3
*******************************************
Average Delay              0.812937  +/- 2.26e-01  p:27.83
Average Rejection rate     0.625375  +/- 2.56e-01  p:40.93
*******************************************
          TRACE RUN 4
*******************************************
Average Delay              0.783619  +/- 1.52e-01  p:19.36
Average Rejection rate     0.592031  +/- 1.72e-01  p:29.05
*******************************************
          TRACE RUN 5
*******************************************
Average Delay              0.740776  +/- 1.57e-01  p:21.22
Average Rejection rate     0.544302  +/- 1.76e-01  p:32.41
```

Una volta inserito i dati di input il simulatore viene avviati e mostra i dati raccolti dopo ogni run, oltre al ritardo medio sperimentato è stato aggiunto il tasso medio di rifiuto.

5.3 File di output

```
*******************************************
       SIMULATION RESULTS
*******************************************

Input parameters:
Buffer Queue limit:        2
Mean batch size:           2
Transient length (s)       100.000
Run length (s)             100.000
Number of runs             5
Traffic load               1.537
Average service duration   0.400
Results:
Average Delay              0.740776  +/- 1.57e-01  p:21.22
Average Rejection rate     0.544302  +/- 1.76e-01  p:32.41
D  1.537376   0.740776   1.57e-01 -1.144357
```

Alla fine di tutti i run richiesti vengono mostrati i dati di output, risultati della simulazione.

Sono state aggiunte le voci riguardi alla scelta della grandezza del buffer K, il valor medio dei lotti C e il tasso medio di rifiuto.