Simulació i Mètodes Numèrics
Simulació de cues

Lluís Batlle i Rossell
Miquel Pedrós Shatarski

21 de febrer de 2003

1 Introducció

L'objectiu d'aquest treball és observar el comportament de cues d'espera per sistemes amb temps de procés. Per aconseguir-ho, hem implementat les cues en Matlab[1] i n'hem observat la resposta a entrades modelades per variables aleatòries. Tenim diferents tipus de cua, al igual que diferents tipus d'entrades per cada una. També podem modificar paràmetres tan de les variables aleatòries d'entrada com de les cues.

Els resultats que observarem són la *probabilitat de pèrdues*, el *nombre de paquets sortits*, el *temps mig d'espera* i el *temps mig de procés*. El primer paràmetre ens permet veure la quantitat de paquets que la cua rebutja (comparant amb el sistema sense cua). El nombre de paquets sortits determina, per exemple en el cas de cues amb prioritat, de quina prioritat són els paquets que més s'han processat. El temps mig de procés el tenim modelat per una variable aleatòria molt concreta, i potser té menys interès; de tota manera, el temps mig d'espera (o bé temps mig en cua) que ens mostra quan s'ha hagut d'esperar el paquet a ser processat.

1.1 Entrades

Les entrades són diferents segons els tipus de cua. Les entrades a les cues síncrones estan modelades per una VA de Bernoulli (de paràmetre p) a cada instant de temps. Les de les cues asíncrones estan modelades per processos de Poisson (de paràmetre λ); això és que el temps entre entrades és una VA exponencial.

En el cas de múltiples entrades, cada una d'elles està modelada com acabem de descriure.

1.2 Tipus de cues

Hem simulat diversos tipus de cues. De tota manera, hem treballat molt més en cues asíncrones, ja que són les que tenen més anàlogs reals. Així doncs, les nostres implementacions són de:

- Cua síncrona de vàries entrades
  - `sincrona.m`
  - `test_sincrona.m`
- Cua asíncrona de vàries entrades i vàries sortides (dues implementacions), amb els paquets que escullen sortida.
  - `asincrona_dir.m`
  - `asincrona_dir2.m`
  - `test_sincrona_dir.m`
- Cua asíncrona de vàries entrades i vàries sortides, amb prioritat. Els paquets que arriben amb més prioritat són rebutjats si la cua és plena.

[1] Les pràctiques han estat provades en Matlab for Linux Version 6.1.0.450 Release 12.1, May 18 2001. Per les gràfiques, l'hem executat en un Athlon 1.2GHz sense cap altre càrrega que el Matlab.

  - `asincrona_dir_p.m`
  - `test_sincrona_dir_p.m`

- Cua asíncrona de vàries entrades i vàries sortides, amb prioritat. Els paquets que arriben amb més prioritat que altres de dins la cua quan aquesta està plena fan fora els de dins la cua, i ells entren.
  - `asincrona_dir_p2.m`
  - `test_sincrona_dir_p2.m`

Cada cua permet establir la mida de cada una de les seves cues (una per sortida).

Els `M-files` que comencen amb `test_` són els que obtenen les gràfiques de les respectives simulacions de cua. Cal dir que en el cas de la *cua asíncrona dirigida*, tot i tenir dues implementacions, al obtenir les gràfiques només utilitzem la primera, ja que és més ràpida.

2 Algoritmes

A continuació comentarem la forma en què hem implementat les cues a simular. Totes elles són independents de les petites `M-files` que creen les gràfiques amb resultats sobre la simulació. De tota manera, les entrades sí que estan modelades dins la `M-file` de la cua.

2.1 Cua síncrona

El motor és un bucle que s'executa per cada instant de temps de l'entrada. Utilitzem una matriu que representa la cua a cada instant, i simplement hi afegim o eliminem elements segons convingui. Anem comptabilitzant els elements que han entrat a la cua i els perduts (que han estat rebutjats perquè la cua estava plena). A cada instant de temps surt un element de la cua (es processa).

Les múltiples entrades es processen per cada instant de temps. És possible que en un instant de temps entrin varis elements a la cua. De tota manera, només en sortirà[2] un cada instant.

D'aquesta cua n'obtenim el temps mig d'espera (temps en cua) i la probabilitat de pèrdues. El temps mig d'espera serà petit si la cua normalment està poc plena, i serà proporcional a la mida de la cua si està molt plena. La probabilitat de pèrdues també es comportarà igual: alta com més entrades hi hagi per instant de temps.

2.2 Cua asíncrona dirigida

Ara els paquets arriben en instants de temps diferents, d'entrades diferents, i cada un volent anar a una sortida diferent (una altra cua). El primer que hem de fer és ordenar temporalment tots els paquets que entraran a les cues (siguin de l'entrada que siguin). Llavors, els podrem processar un darrera l'altre. Ja veiem que les múltiples entrades només determinen que hi hagi més entrada de paquets a les cues.

[2]En aquest treball utilitzarem sovint el terme *sortir* per referir-nos a que un element és processat

2.2.1 Primera implementació

Una vegada tenim les entrades ordenades, tenim un bucle que les va recorrent en temps. A cada entrada li calculem immediatament el temps de sortida. Només sabent l'instant de l'entrada, i els temps de sortida de les anteriors variables, podem calcular el temps que s'estarà en cua.

Si la cua és plena (podem contar els elements en cua, igual que calculem l'instant de sortida), l'element es rebutja, i compta com a perdut.

2.2.2 Segona implementació

Amb les entrades ordenades, un bucle les va recorrent en temps. Es segueix un mètode similar a la cua síncrona: s'utilitza una estructura de dades que representa la cua a cada instant de temps. Cada paquet que arriba ens determina quant de temps ha passat des de l'anterior entrada. A l'inici del bucle mirem quants paquets han sortit en aquest temps que ha passat, i llavors afegim a la cua l'entrada que acabem de tenir.

Cada cop que traiem un element de la cua en guardem l'instant de sortida. Si la cua és plena, l'element es rebutja, i compta com a perdut.

Un cop tenim els instants de sortida i el nombre de paquets perduts, podem extreure'n les dades estadístiques resultat de la simulació. En la nostra implementació, se'ns retorna el temps mig de servei, la probabilitat de pèrdues, i el temps mig en sistema.

A l'enunciat de la pràctica s'ens demanava la simulació d'una cua asíncrona sense ser dirigida; bé, això només és el cas particular amb una sola cua de sortida.

2.3 Cua asíncrona dirigida amb prioritat

Existeixen varis models que podrien donar a diferents implementacions d'una cua amb prioritat[3]... Nosaltres hem considerat dos casos: que els elements de més prioritat puguin fer fora a altres de menys prioritat de la cua, i que no puguin. La diferència de codi és molt petita, i per això no les comentarem apart.

Podem dir que l'algoritme amb què simulem aquesta cua és molt similar al que no té prioritat (la segona implementació, que té una estructura de dades que representa la cua). Nosaltres simplement mantenim una estructura de dades on tenim a cada instant on toca anar cada paquet segons la seva prioritat. Per exemple, per una cua plena, els elements de més prioritat aniran més endavant (prop de la sortida), i els de menys aniran cap al final de la cua. Aquests índex dins la cua per cada prioritat venen determinats per totes les entrades en cua que hi han hagut abans. Per tant, els anem determinant dinàmicament cada cop que fem una entrada (o una sortida) de la cua.

3 Resultats

Totes les simulacions s'han fet amb dues entrades.

[3]En aquest treball entenem el nivell de probabilitat com a números naturals. Els més petits són els de més prioritat, i viceversa

3.1 Cua síncrona

Hem executat la simulació de la cua amb els següents paràmetres:

- Probabilitat d'entrada p: de 0.01 a 1, 20 mostres
- Mides de cua: de 1 a 21, de 2 en 2
- Instants de temps: 1000

Així, la crida era:

```
test_sincrona(0.01,1, 1:2:21, 1000);
```

Respecte a la probabilitat de pèrdues (Figura 1) veiem que com més llarga és la cua, menys probabilitat de pèrdues hi ha. De tota manera, el paràmetre que més la determina és la freqüència cada quan hi ha entrades. Podem dir que per $p > 0.5$ la cua s'emplenarà ràpidament, i hi haurà molta pèrdua de paquets.

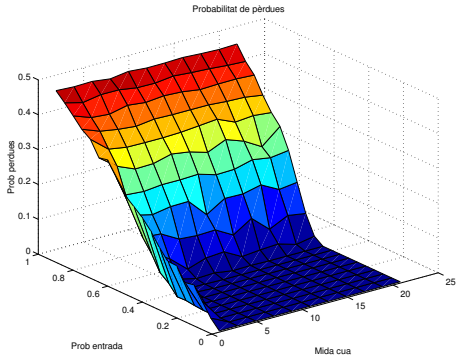

Figura 1: Probabilitat de pèrdues en cua síncrona. CPU : 14.64 s, junt amb el Temps mig en cua.

Respecte el temps mig d'espera, sabrem que serà més gran com més plena estigui la cua de mitjana. I això és proporcional a la mida de la cua (per $p > 0.5$), i serà més o menys constant per probabilitats d'entrada $p < 0.5$ (la cua estarà normalment buida). Ho podem veure a la Figura 2

3.2 Cua asíncrona

Per a fer aquesta simulació hem utilitzat el cas particular d'una sortida de la *cua dirigida*. Hem executat la simulació de la cua amb els següents paràmetres:

- Nombre de sortides: 1
- Probabilitat d'entrada λ: de 0.1 a 4, de 0.2 en 0.2
- Mides de cua: de 1 a 20, de 2 en 2
- Instants de temps: 1500

Així, la crida era:

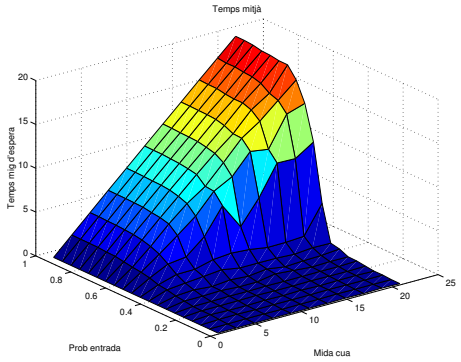

Figura 2: Temps mig en cua síncrona. CPU : 14.64 s, junt amb la Probabilitat de pèrdues.

```
test_asincrona_dir(0.1:0.2:4, [1:2:20], 1500, 1);
```

Veiem que el temps mig de servei està al voltant de la unitat (VA generada per nosaltres); també veiem que quan λ és gran, el sistema té un comportament molt ben definit: El temps mig en sistema creix proporcionalment a la mida de la cua, i la probabilitat de pèrdues és molt alta sigui quina sigui la mida de la cua.

En el cas de que $\lambda < 0.5$ tindrem que la cua estarà buida normalment; això passa perquè hi ha dues entrades. Podriem dir que dues entrades amb $\lambda = 0.5$ equivalen a una amb $\lambda = 1$ (que és la que determina el temps de processat – VA exponencial). La probabilitat de pèrdues és baixa en aquest cas, i el temps mig de sistema deixa de dependre de la mida de la cua (ja que mai arriba a estar plena). Tot això ho veiem a la Figura 3.

3.3 Cua asíncrona dirigida

Hem executat la simulació de la cua amb els següents paràmetres:

- Nombre de sortides: 2
- Probabilitat d'entrada λ: de 0.1 a 4, de 0.2 en 0.2
- Mides de cua 1: de 1 a 20 de 2 en 2
- Mides de cua 2: de 1 a 10 de 1 en 1
- Instants de temps: 1500

Així, la crida era:

```
test_asincrona_dir(0.1:0.2:4, [1:2:20; 1:1:10], 1500, 2);
```

Els resultats són els mateixos que en el cas de la *cua asíncrona*, però amb les entrades repartides entre dues cues (ja que la probabilitat de que una entrada esculli una o altra sortida la determina una VA uniforme). Podem veure els resultats a les Figures 4 i 5, per la primera i segona cua respectivament.

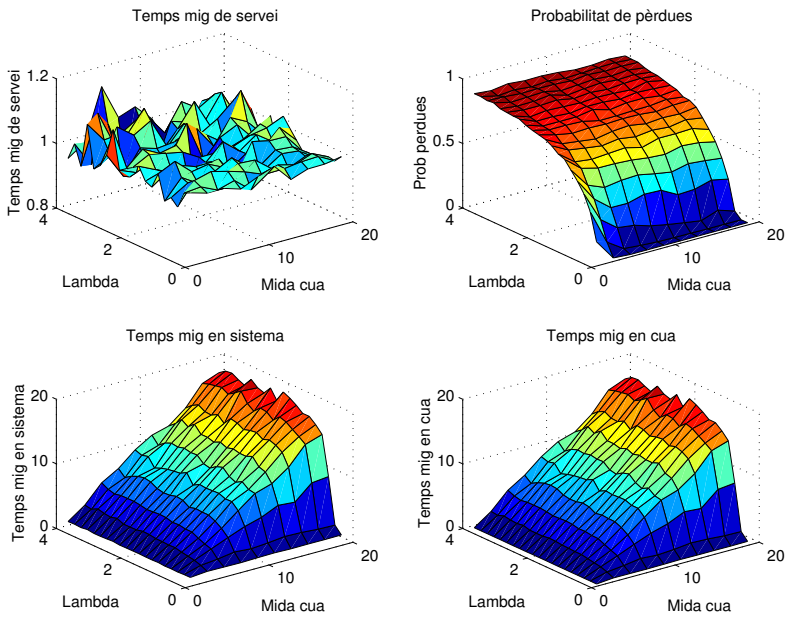

Figura 3: Cua asíncrona. CPU : 84.56 s

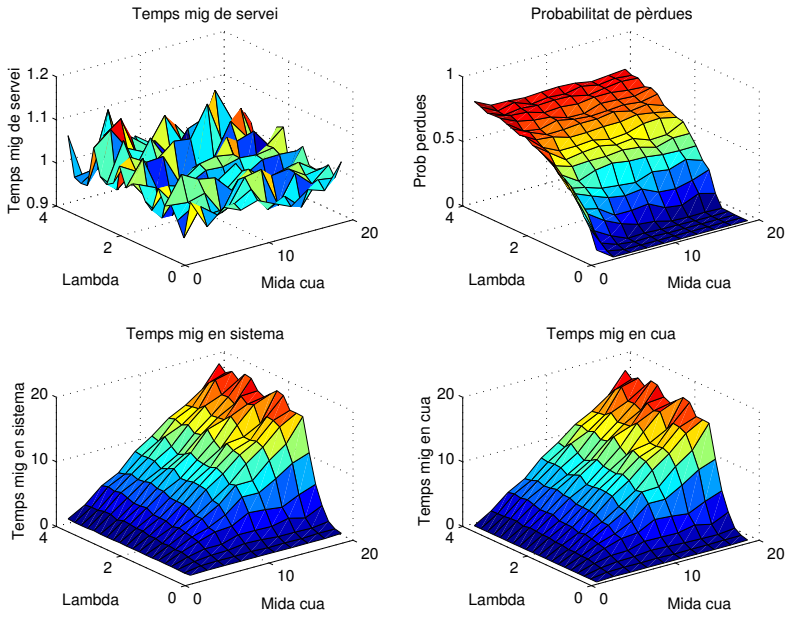

Figura 4: Cua asíncrona dirigida, cua 1. CPU : 88.64 s, junt amb cua 2

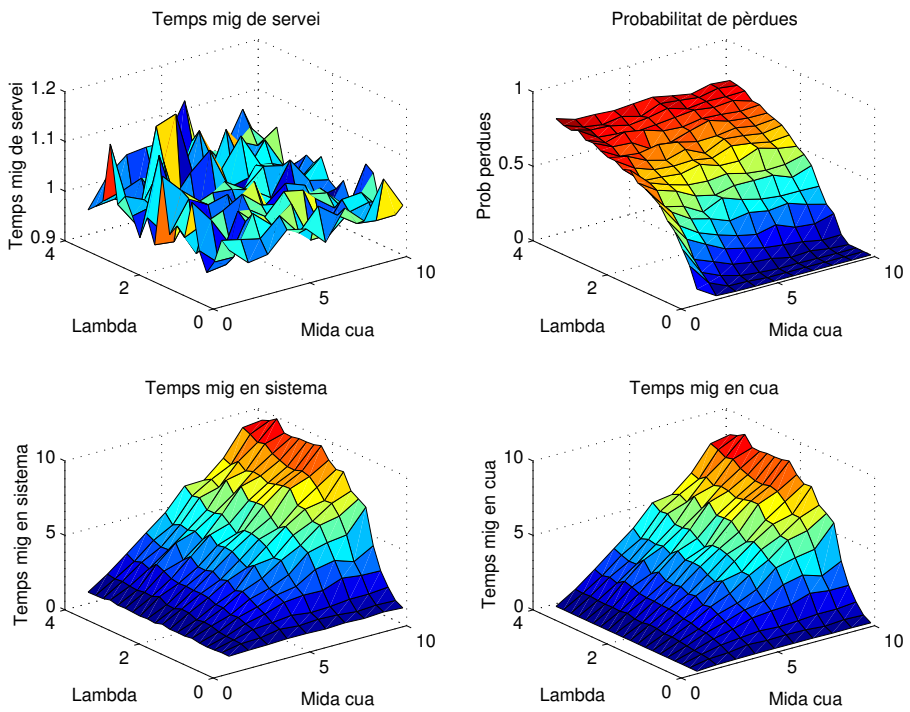

Figura 5: Cua asíncrona dirigida, cua 2. CPU : 88.64 s, junt amb cua 1

3.4 Cua asíncrona dirigida amb prioritat

Les dues implementacions de la cua són prou diferents com perquè mereixin ser comentades apart. Està clar que representen diferents models de la realitat. El Tipus 1 es refereix a la cua que quan està plena no hi pot entrar cap altre element fins que es buidi, i el Tipus 2 a la que els elements de més prioritat poden fer fora de la cua a altres de menys.

A més, les cues tenen respostes molt diferents a diferents fluxes d'entrada. En comentarem de ràpids ($\lambda = 5$), normals ($\lambda = 1$) i lents ($\lambda = 0.2$).

Els paràmetres que hem utilitzat per les simulacions són:

- Probabilitat d'entrada λ: 5, 1 i 2
- Mida de cua: 20
- Instants de temps: 5000
- Número de sortides: 1

Així, la crida era pel *Tipus 1*:

```
test_asincrona_dir_p(lambda, [20], 5000, 1, 20);
```

i pel *Tipus 2*:

```
test_asincrona_dir_p2(lambda, [20], 5000, 1, 20);
```

3.4.1 Tipus 1

La resposta a un fluxe ràpid d'aquesta cua és que el temps d'espera dels elements de menys prioritat és molt més baix que el temps dels de més prioritat. No podem obtenir resultats de la probabilitat de pèrdues o del número d'elements que surten, ja que mentre la cua no estigui plena no es rebutja cap element en concret. Ho observem a la Figura 6.

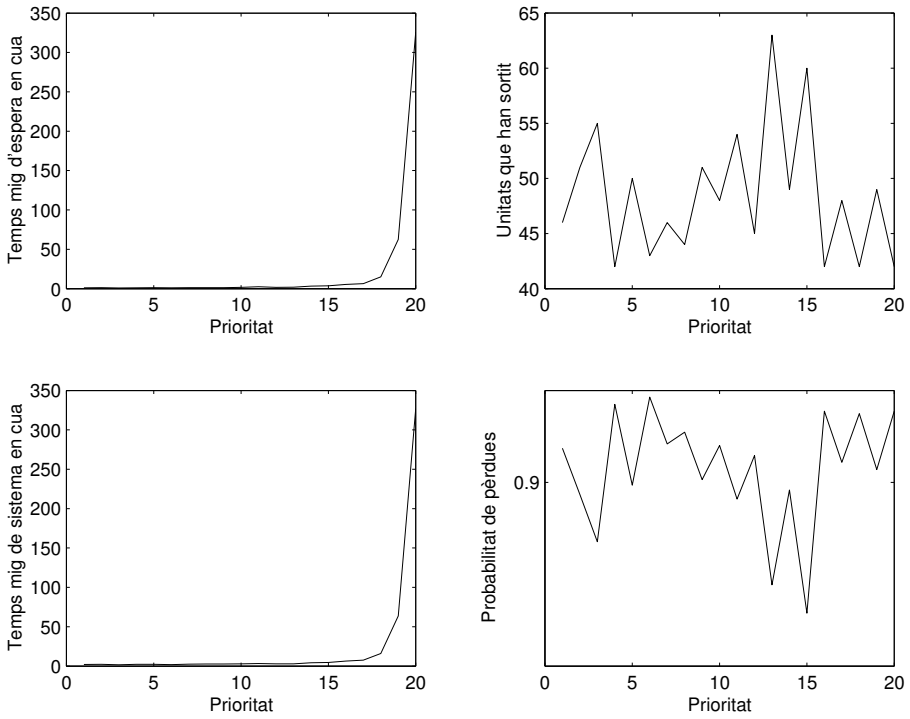

Figura 6: Cua asíncrona dirigida amb prioritat, Tipus 1, entrades ràpides $\lambda = 5$. $CPU : 1.6\,\mathrm{s}$

A un fluxe normal veiem una resposta similar a un fluxe ràpid. La gràfica del temps mig d'espera en cua és una mica menys brusca, simplement. Ho veiem a la Figura 7.

Amb un fluxe lent podem veure clarament la jerarquia de prioritats en la gràfica del temps mig d'espera. En aquest cas veiem també que la probabilitat de pèrdues és 0. Ho podem veure a la Figura 8.

A simple vista, el temps mig d'espera té tendència a ser una gràfica similar a $f(x = (0, 1)) = e^{ax}$, on a és proporcional al fluxe d'entrada, i x és la probabilitat.

3.4.2 Tipus 2

La resposta a un fluxe ràpid d'aquesta cua ens demostra perquè serveix aquest model de cua: els elements de poca prioritat són rebutjats. La gràfica que ens crida més l'atenció és la del nombre d'unitats que han sortit; igualment (en relació amb aquesta) tenim la probabilitat de pèrdues, que és més gran com de més prioritat són els elements d'entrada. No tenim punts evaluats en les

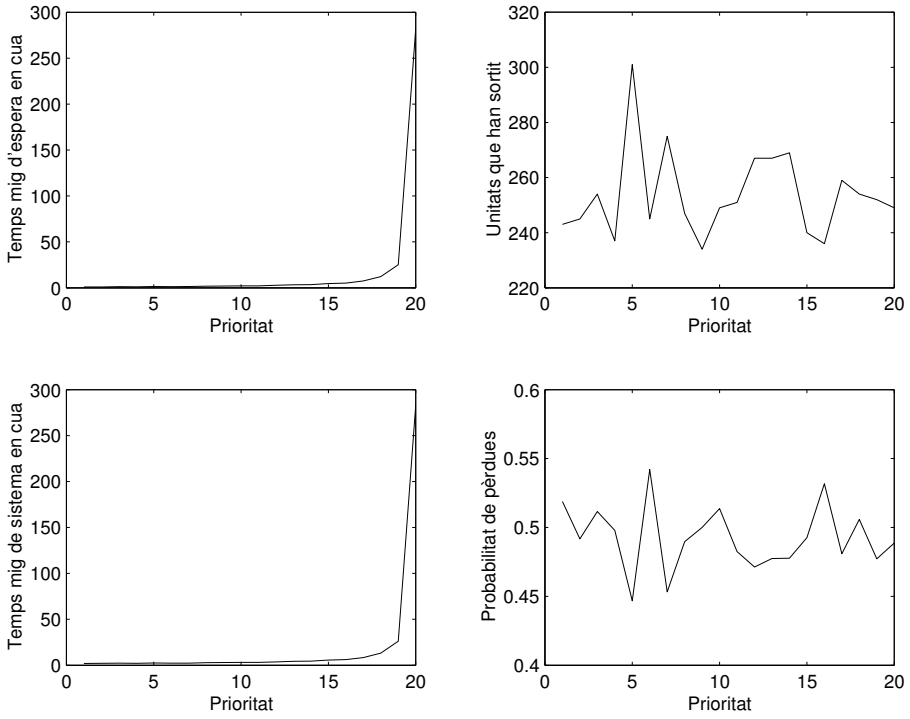

Figura 7: Cua asíncrona dirigida amb prioritat, Tipus 1, entrades normals $\lambda = 1$.
CPU : 2.99 s

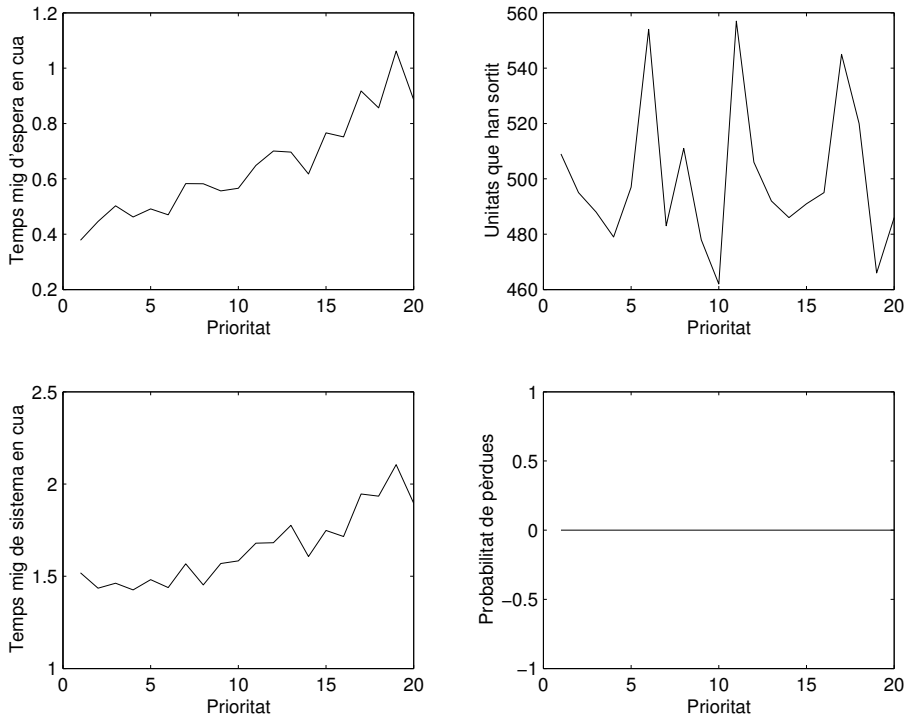

Figura 8: Cua asíncrona dirigida amb prioritat, Tipus 1, entrades lentes $\lambda = 0.2$.
CPU : 3.92 s

prioritats baixes del temps mig d'espera, ja que no hi ha elements que hagin sortit de la cua per poder-los-hi mesurar. Ho veiem a la Figura 9.

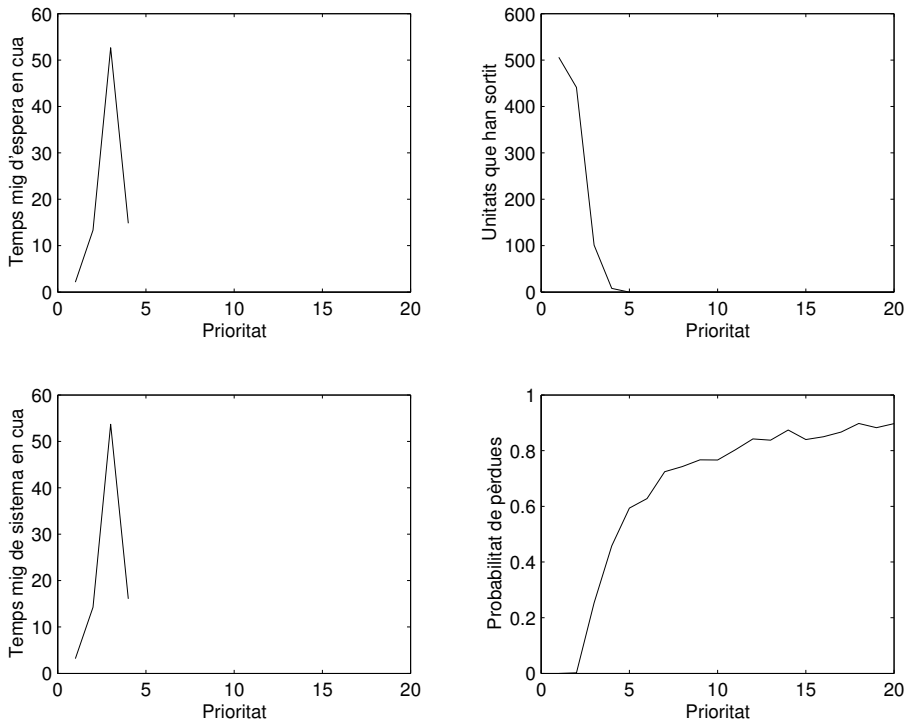

Figura 9: Cua asíncrona dirigida amb prioritat, Tipus 2, entrades ràpides $\lambda = 5$. CPU : 1.93 s

La resposta a un fluxe normal és encara més definitòria. A partir d'una certa prioritat (que és funció del fluxe d'entrada) els elements comencen a ser descartats. No tenen oportunitat de ser processats perquè els elements de més prioritat fan fora als que en tenen més. De tota manera, els elements de prioritat alta es comporten com si estéssin en una cua del *Tipus 1*. Hi continua havent una probabilitat de pèrdues baixa pels elements d'alta prioritat. Ho veiem a la Figura 10.

En el cas d'una cua lenta, es comporta exactament igual que una cua de *Tipus 1*. La diferència entre sengles cues rau en com actua quan la cua està plena; en aquest cas, els dos tipus responen sense que la cua s'empleni. Ho veiem a la Figura 11.

4 Possibles millores

La implementació de les cues que hem fet pot ser millorada en varis aspectes, i alguns d'ells mereixen ser comentats. Per exemple, no hem provat com responen les cues a altres models d'entrada. Podiem haver provat ràfagues, que és el que passa a hores punta o en events especials en alguns models reals. També podiem haver estudiat més a fonts una cua síncrona, o simplement haver modelat una entrada síncrona a cada una de les cues asíncrones. També podiem haver

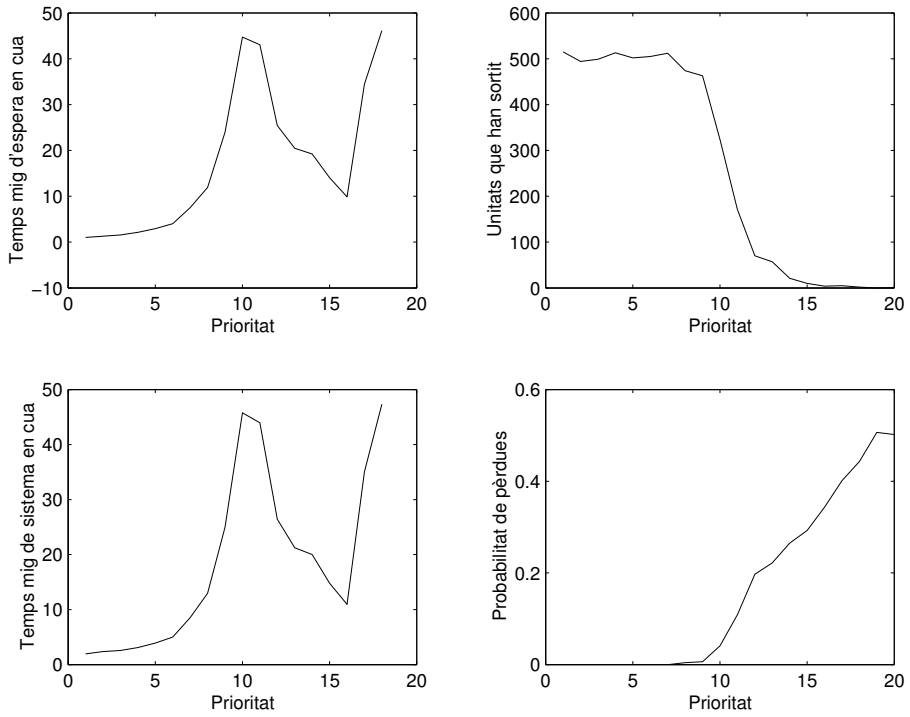

Figura 10: Cua asíncrona dirigida amb prioritat, Tipus 2, entrades normals $\lambda = 1$. $CPU : 3.67\,\mathrm{s}$

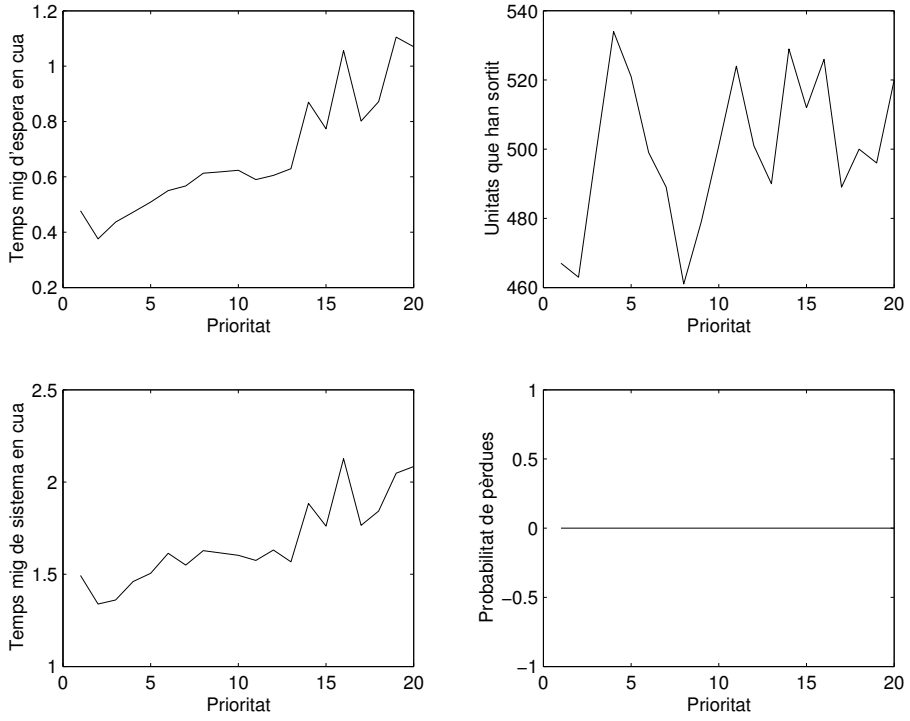

Figura 11: Cua asíncrona dirigida amb prioritat, Tipus 2, entrades lentes $\lambda = 0.2$. $CPU : 3.85\,\mathrm{s}$

estudiat el temps de procés dels paquests (per exemple, variar-lo segons un tamany de paquet). I com no, també podiem haver modelat altres tipus de cues.

Val a dir que hem fet dues implementacions de la cua amb prioritat, però en podiem haver fet dues més: depenent de si la prioritat pot fer fora un element que s'estigués processant o no. Nosaltres hem fet que l'element processat no es pot tocar fins que surti del sistema.

De tota manera, els resultats que hem obtingut ens ofereixen una bona visió de com es comporten aquestes cues, així com comprendre millor quina és la millor elecció alhora de necessitar una cua en una aplicació real. Cada una té una resposta molt diferent segons els fluxes d'entrada, i això ens permet obtenir una bona relació entre els recursos que pot dedicar un sistema a cues, i l'eficiència que tindrà.

5 Codi font

El codi font de les `M-files` l'adjuntem a continuació, de cada fitxer. Per una còpia electrònica dels fitxers, es poden trobar les últimes versions (les que han generat les gràfiques d'aquest treball) a:

- *http://vicerveza.homeunix.net/viewcvs/viewcvs.cgi/practiques/metodes/projecte/?cvsroot=Viric*