# Realtidssystemsschemaläggning med (m,k)-Firm garanti och återkoppling

Skribent: Oskar Östman

Institution: Åbo Akademi

8.4.2010

Handledare: Hannu Toivonen

# Abstrakt

Jag försöker I detta arbete beskriva ett återkopplat realtidssystem byggt med hjälp av FC-EDF ramverket. Utöver en beskrivning av systemet och dess komponenter så testar jag också systemet med en simulering av två olika testfall som den ursprungliga FC-EDF algoritmen också testats mot. Systemet kommer i korthet bestå av en schemaläggare: ETF, en regulator: PID, en prioritetstilldelare: DBP som arbetare med (m,k)-Firm garantin och själva exekveraren som utför jobbena i systemet.

# Innehåll

# 1. Inledning

För att förklara idén bakom en återkopplad realtidssystems schemaläggare så måste begreppen realtidssystem och realtidssystems schemaläggare(*eng. real-time system scheduler*) närmare förklaras. Ett realtidssystem är ett system vars pålitlighet baseras på dess logiska resultat, såsom alla system gör, samt på tiderna av vilka jobbena(*eng. tasks*) körs i systemet [Murthy01]. Medan en schemaläggare såsom namnet antyder gör ett schema för hur inkommande jobb skall hanteras. Det finns flertalet schemaläggare för realtidssystem såsom EDF, RMS m.fl. de vanligaste schemaläggarna har dock en gemensam svaghet: de blir opålitliga när systemet är överbelastat[Chan99].

Vid design av ett realtidssystem försöker man inkludera all beräkningskraft som behövs men det är inte alltid möjligt eller kostnadseffektivt att designa ett system med tillräckliga resurser. Tänk ett system med tillfälligt hög belastning , det skulle bli dyrt att använda extensiv hårdvara som vanligen skulle vara onödig bara för att klara sällsynta pikar i belastningen. Ibland är system belastningen bara svår att förutspå vilket kan leda till att hårdvaran inte räcker till. I dessa fall behövs en algoritm som kan hantera överbelastningar [Chen99].

Orsaken varför de två vanligaste algoritmerna EDF och RMS lämpar sig dåligt för överbelastning är att de är öppen loop (*eng. open loop*) schemaläggare, designade för förutsägbara jobb. Ett förutsägbart jobb är ett jobb vars exekveringstid kan exakt estimeras vilket ofta inte är fallet. Att algoritmerna är öppen loop innebär att de inte kommer att ändra schemat baserat på återkoppling, så ifall den nuvarande modellen av jobben är inkorrekt kommer schemaläggaren att fortsätta arbeta under felaktiga antaganden, vilket medför antagande och vägrande av fel mängd med jobb in i systemet[Murthy01].

Det är svårt att förutspå somliga systems belastning t.ex. robotstyrning och försvarssystem medan andra system kan verka med en viss överbelastning t.ex. videostreamings tjänster där ett par missade bilder i ett större sammanhang inte förstör helhetsupplevelsen. Dessa system försöker möta så många deadlines som möjligt med undantaget att livsviktiga jobb får företräde så de säkert exekveras inom utsatt tid [Chen99] [Hamd95].

Ett vanligt sätt att lösa problemet med överbelastning på är att anta den längsta möjliga exekveringstiden vid okänd riktig exekveringstid, vilket innebär att systemet kommer att

ha outnyttjad kapacitet. Jag kommer dock i detta arbete att beskriva ett system baserat på FC-EDF ramverket som beskrivs i [Chen99]. För att göra överbelastningssituationer förutsägbarare så kommer jag använda olika prioritets nivåer som skapas med hjälp av (m,k)-firm deadlines och varje prioritetsnivå ordnas med EDF schemaläggaren där antalet instanser av varje jobb kontrolleras med PID återkoppling. Enligt FC-EDF design så uppnås högt utnyttjande av kapaciteten och förutsägbarheten blir förhoppningsvis bättre med avseende på vilka jobb som vägras/nerprioriteras. Utöver beskrivning av systemet görs simuleringar där resultaten jämförs med FC-EDF enligt resultaten som Chenyang Lu m.fl. presenterade i [Chen99].

I artikeln "Design and Evaluation of a Feedback Control EDF Scheduling Algorithm" presenterar Chenyang m.fl. FC-EDF ramverket hur man implementerar återkoppling i ett realtidssystem på ett generellt sätt. I deras tillämpning sätts andelen missade deadlines som referenssignal vilket jag också kommer använda mig av, men de använder graden av systemutnyttjande som styrsignal medan jag kommer använda m:et i (m,k)-Firm deadlines som min styrsignal.

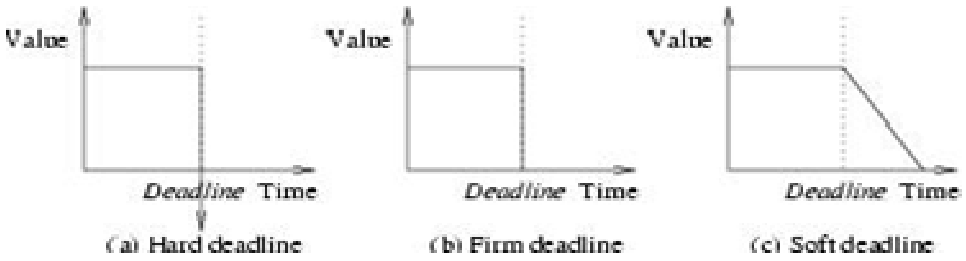
# 2. Viktiga termer och begrepp

Systemet jag vill beskriva kommer i korthet att bestå av en EDF schemaläggare baserad på (m,k)-Firm garanti där antalet obligatoriska jobb regleras med en PID regulator som tar mätvärden med en rätt så stor period för att inte använda onödigt mycket systemresurser för regleringen. Detta system kommer att vara ett förutsägbart system eftersom man kan bestämma hur stor del av processorkraften som skall tillägnas till ett visst jobb eller man kan till och med kräva att viktiga jobb alltid får exekveras. Eftersom jag vill underlätta en jämförelse med [Chen99] så har jag valt att använda samma jobbmodell som Chenyang m.fl. använde, vilken jag kommer presentera närmare i kapitel 5. Målet med kapitlet är att beskriva alla viktiga begreppen och metoder som behöver utredas för att förstå hur den återkopplade schemaläggaren jag presenterar ser ut och fungerar.

## 2.1 Belöningar i olika typer av realtidssystem

Inom realtidssystem så är deadlines och mötandet av dessa deadlines grunden för hela systemen. Av den orsaken delas olika realtidssystem in i 3 kategorier(kategorierna presenteras på engelska eftersom jag inte kunnat finna några svenska översättningar som används):

1. *Hard deadline*: straffet för en missad deadline är större än belöningen för en mött deadline.
2. *Firm deadline*: straffet och belöningen är av samma storlek.
3. *Soft deadline*: straffet ges i relation till förseningen och kommer ofta vara lägre än belöningen för exekveringen av processen.




**Figur 2.1** Diagram av exempel belöningar för olika realtidssystem [Myrthy01].

## 2.2 EDF algoritm

Earliest Deadline First (EDF) är en dynamisk schemaläggaralgoritm som används i realtidssystem. Den placerar jobbena den schemalägger i en lista på basen av jobbens respektive deadlines. Jobbet med den minsta deadlinen får den högsta prioriteten; det jobbet schemaläggs sedan som följande jobb för exekvering.

EDF är en optimal preemptive schemaläggare på enprocessor(eng. uniprocessor) system. Optimal innebär att om det är möjligt att schemalägga en mängd jobb på någon processor: ready time(ankomsttid), exekverings tid och deadline, så att jobben kan slutföras innan sin deadline då klarar en EDF schemaläggare av samma bedrift när overhead inte beaktas. Preemptive innebär att ett påbörjat jobb kan avbrytas vid varje tidssteg ifall ett jobb med lägre deadline uppenbarat sig.

Om deadlinen är likamed eller större än jobbets period, då har EDF en belastningsgräns på 100% och kan undersökas med följande belastnings test, testet är både tillräckligt och nödvändigt(eng. Sufficient and necessary).

$$\sum \frac{C_i}{P_i} \leq 1$$

Men EDF schemaläggaren kan endast garantera deadlinen när processor belastningen är m indre än 100%, jämfört med den andra jämte EDF populära schemaläggaren Rate Monotonic Scheduling (RMS) så är det dock mycket bra. Vid ett system med ett stort

antal jobb kan RMS garantera ett schema utan deadlines missar upp till 69 % processor belastning [Murthy01].

Men om EDF schemaläggaren blir överbelastad kommer den dock bli rätt så oförutsägbar, som en funktion av deadlinerna och när överbelastningen uppstår.

## 2.3 (m,k)-Firm garanti

I kapitel 2.1 introducerades Hard-deadlines och Soft-deadlines på system nivå man kan dock gå ner på jobb nivå och kategorisera jobb som livsviktiga, Hard deadline, och jobb där vissa störningar godtas, Soft deadline. Varför ett jobb kan kategoriseras på detta sätt är för att ett jobb ofta upprepar sig, t.ex. ett jobb som mäter temperaturen med 10 sekunders intervall, en sådan uppgift vilken man kan anta att inte orsakar livsfara vid en missad exekvering. Om vi antar att varannan mätning uteblir utan att resultatet försämras märkbart så kan vi bestämma att jobbet har en tolerans på 50 %. Det krävs dock ett villkor som specificerar att 2 missar på varandra följda av 2 mötta deadlines, som även blir 50 % inte tolereras, för att beskriva detta krav används (m,k)-Firm garanti. (m,k)-Firm garanti säjer att systemet skall garantera m jobb utav k exekveras inom utsatt deadline, vi kan alltså med (1,2)-Firm garanti kräva att vartannat jobb skall exekveras i tid.

Varje jobb har alltså en deadline och om jobbet är klart innan deadlinen så säjs jobbet ha mött sin deadline. Idén med (m,k)-Firm deadlines är ganska simpel ifall varje instans av ett jobb måste klara sin deadline så representeras den av villkoret (1,1)-Firm deadlines. Medan t.ex. (2,3)-Firm deadline innebär att minst 2 utav varje 3 på varandra följande jobb, av samma instans, måste klara sin deadline. (m,k)-Firm garanti är i sig själv ingen schemaläggare utan den måste kombineras med en sådan, jag kommer beskriva den i kombination med en EDF schemaläggare (se kapitel 2.2) i detta arbete.

### 2.3.1 DBP prioritetstilldelning

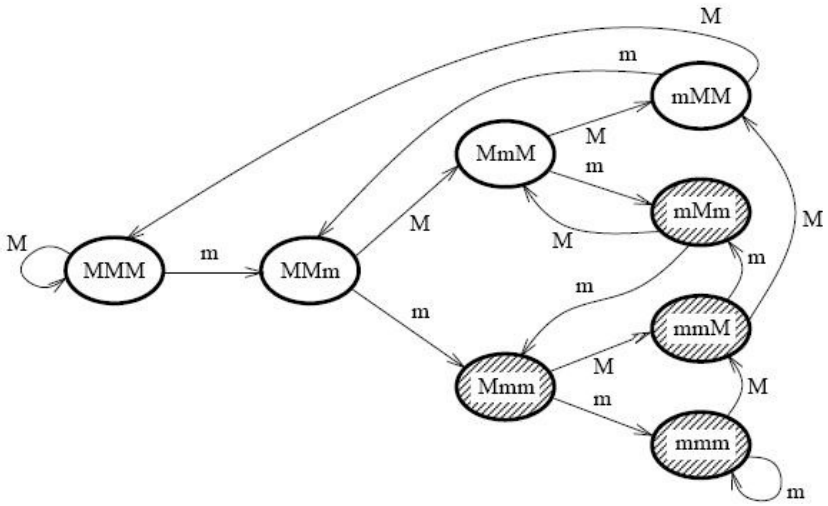
För varje jobb så upprätthåller systemet en närhistoria för att hålla reda på mötta och missade deadlines. Med hjälp av de k (i (m,k)) senaste instanserna av jobbet så kan man se ifall (m,k)-Firm kriteriet möts eller ej. Distance-Based Priority (DBP)[Hamd95] är en metod för bestämning av prioritet på basen av denna närhistoria.

Närhistorien $s$ för jobbet förvaras i ett $k$-bitars stort register där 1 representerar en mött deadline och 0 en missad deadline. När följande instans av jobbet aktiveras så skiftas

registret från höger ett steg med det nya värdet, i.e. 1 eller 0. Följande formel beräknar jobbets nuvarande prioritet:

$$prioriteten = k - l(m,s) + 1,$$

Där $l(n,s)$ är positionen från höger där den $n$:te 1:an från närhistoren $s$ av jobbet. Ifall det finns mindre än $n$ 1:or i $s$, då är $l(n,s) = k + 1$. Alltså är högsta prioriteten noll som endast ges till jobb som inte lyckats hålla (m,k)-Firm garantin, för att illustrera så skall jag visa ett exempel: om vi har ett jobb med (1,3)-Firm garanti låt då närhistorien vara MMm där litet m är missad deadline och stort M är mött deadline. $l(1, MMm) = 1$ och prioriteten blir då $prioritet = 3 - l(m,s) + 1 = 3 - 1 - 1 = 1$ [Hamd95].



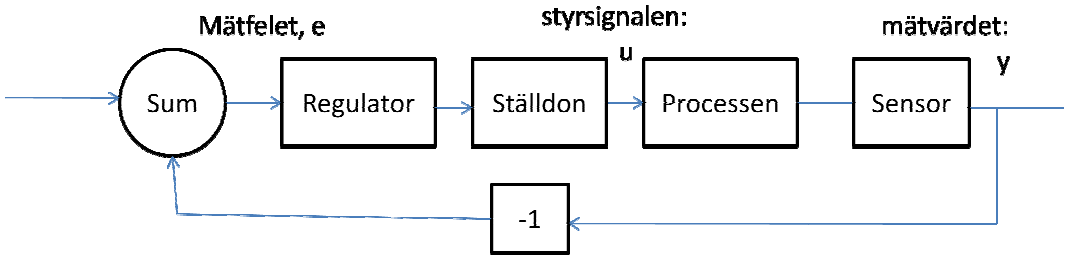

**Figur 2.2** Tillståndsdiagram exempel vid (2,3)-Firm deadline [Hamd95].

## 2.4 PID återkoppling

### 2.4.1 Introduktion till reglerteknik/återkoppling

Ett enkelt system med återkoppling illustreras i figur 2.3, som visar relationerna mellan styrsignaler och utsignaler. Systemet har en invariabel: *styrsignalen*(*eng. controll variable, manipulated variable*), *u* i figur 2.3. Styrsignalen kontrollerar systemet med hjälp av ett ställdon(*eng. actuator*), som kommer att verka som *urvalshanterare* (*eng. admission controller*) i detta arbete, men kunde typiskt vara en motor eller en ventil. Systemets utsignal kallas *mätvärdet* (*y),* värdet registreras av någon typ av sensor, i detta arbete kommer sensorn vara del av simulerings- programmet som är skrivet i C++ vilket gör det svårt att särskilja en speciell sensordel. Det önskade värdet för mätvärdet

bär namnet referenssignal(*eng. set point*), $y_{sp}$ i figur 2.3. Mätfelet $e$ är differensen mellan referenssignalen och mätsignalen som fås enligt: $e = y_{sp} - y$.




**Figur 2.3** Block diagram av ett återkopplat system.

Om vi antar att systemet är sådant att mätvärdet ökar när styrsignalen ökar och vice versa. I detta fall kan återkoppling beskrivas med följande citat av från boken[Åström06](citatet är översatt till svenska från engelska):

”Öka styrsignalen när felet är positivt och minska styrsignalen när felet är negativt”

Denna typ av återkoppling kallas negativ återkoppling eftersom styrsignalen negeras när felet kalkyleras, $e = y_{sp} - y$.

PID regulatorn är den vanligaste typen av regulator för återkopplade system. Den har blivit utvecklad under en lång tidsperiod och överlevt flera plattforms byten: från mekaniska till elektriska till nuvarande dator baserade system som verkar i diskret tid, vilket innebär att endast mätvärdena vid samplingstidpunkterna beaktas.

Om man studerar figur 2.3 kan man observera att om värdet på mätvärdet *y* är nära referenssignalen $y_{sp}$ så kommer felet att bli litet. Men för att det skall fungera bra behövs ett noggrant ställdon, en noggrann sensor och en fungerande mekanism som sköter regleringen. Några egenskaper som återkoppling har är:

- Återkoppling kan minska effekten av störningar, i fallet realtidssystem gäller: att den kan hantera en tillfällig topp i mängden inkommande jobb genom att avvisa ”överlopps” jobb så att antalet missade deadlines minimeras.
- Den kan anpassa sig till ändringar från modellen, tackvare återkopplingen som berättar åt regulatorn att någonting har inträffat och att den behöver anpassa sig.

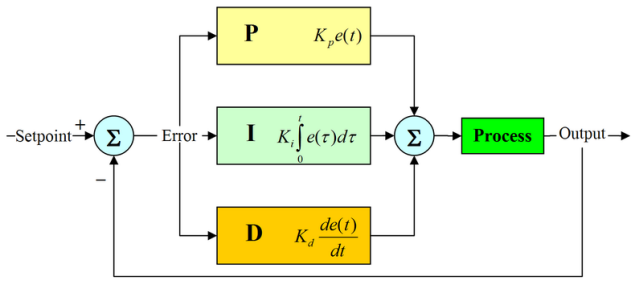
### 2.4.2 PID regulatorn

En PID regulator är en simpel form av ett återkopplat system som är tämligen enkel att implementera. De flesta återkopplade system är reglerade av algoritmen eller en variation av den. PID regulatorn är uppbyggt av 3 delar som namnet säjer P, I och D delarna. PID står för proportionell, integral och derivata.

Den proportionella delen av PID regulatorn kan även användas som en ensam regulator, inverkan som regulatorn då åstadkommer är proportionell till regulator felet, P delen av regulatorn representeras som: $u(t) = Ke(t)$.

Integral inverkan på PID regulatorn baseras på summan av de senaste felen. Integral delen minskar reglerfelet i stationärt tillstånd som proportional och derivata delen kan åsamka och är inte lika känslig mot tillfälliga toppar i mätfelet eller mot störningar. Integraldelen representeras som: $u(t) = K \int e(\tau)d\tau$.

Syftet med derivata delen är att förbättra stabiliteten på den slutna loopen. Derivata delen bestämmer hur snabbt regulatorn skall reagera på plötsliga ändringar i regulator felet och på det sättet eliminerar den långa perioder av instabilitet. Tyvärr så kan derivata delen även ha motsatt resultat ifall systemet har stora mängder störningar, då kan mindre D eller ingen D del överhuvudtaget vara förnuftigast. Derivata delens struktur är följande: $u(t) = K\frac{de(t)}{dt}$[Åström06][Häggblom07].




**Figur 2.4** Block diagram av ett PID system[Wiki10]

Genom att ställa in PID variablerna kan regulatorn erhålla den önskade regleringen för olika system. Hur regulatorn reagerar kan beskrivas enligt dess respons till mätfelet, hur nära den kommer till referenssignalens värde och enligt hur mycket systemet oscillerar.

När alla PID delarna läggs ihop så fås formeln för räknandet av PID regleringen, jag presenterar här två uppställningar av formeln föredrar dock personligen att använda den första för att den på ett intuitivt sätt förstås. Men eftersom den andra uppställningen är populär inom litteraturen och används vid riktlinjerna för hur

regulatorn skall ställas in så behöver den också visas: $u(t) = K_p e(t) + K_i \int e(\tau)d\tau + K_d \frac{de(t)}{dt}$ or $u(t) = K_c(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d \frac{de(t)}{dt})$.

Där regulatorns inställnings parametrarna är:

Proportionella $K_p$, med ett större värde får vi snabbare respons eftersom mätfelet får större genomslag. Ett för stort $K_p$ leder till ett instabilt system och oscillering.

Integral värdet $K_i$, stora värden eliminerar fel i fortfarighetstillståndet(måltillståndet) snabbare. Större $K_i$ kommer även skapa ett trögare system: som förlänger tiden innan äldre fel försvinner från regulatorn, vilket inträffar när nya värden anländer till regulatorn. Alltså minskar ett stort $K_i$ störningar effektivt men fördröjer också tiden det tar att uppnå stationär-tillstånd.

Derivata vikten $K_d$, stora värden ger mindre översläng(se exempelvis [Häggblom07]) men kommer att leda till en längre tid innan fortfarighetstillståndet uppnås p.g.a. aggressivare svängningar orsakade av derivatan. Om $K_d$ är stor kan det leda till instabilitet orsakat av att störningar förstärks [Wiki10].

### 2.4.3. PID design/inställningar

Det finns ett antal olika design metoder för PID regulatorer, de kändaste men tyvärr inte de bästa är Ziegler och Nichols två metoder. Orsaken varför metoderna är så kända är för att de är så gamla, metoderna publicerades redan 1942. En annan orsak är att eftersom de är så simpla så kommer alla process- och systemingenjörer lära sig dem som de första metoderna de använder, det är därför lätt att lockas att använda metoderna om man vill ställa in en PID regulator snabbt. Kommer även i min simulation använda en Ziegler - Nichols metod eftersom jag planerar på att använda en så lång samplingsperiod att noggrannare regulator inställningar inte skulle ändra resultatet .

Ziegler – Nichols stegsvarsbaserade rekommendationer. Denna metod baseras på informationen som fås vid ett stegförsök i ett öppet system. Stegsvaret beskrivs sedan av endast två parametrar *a* och *L*. Punkten där kurvans derivata är störst(stigningen är brantast) bestäms först, sedan ritas en tangent från punkten. Skärningspunkten med x-axeln ger *L* och skärningspunkten med y-axeln ger *a*. Tabell 2.1 används sedan för att ställa in PID variablerna.

| Regulator | $aK_c$ | $T_i/L$ | $T_d/L$ |
|---|---|---|---|
| P | 1 | - | - |
| PI | 0.9 | 3 | - |
| PID | 1.2 | 2 | L/2 |

**Tabel 2.1** Parametrar för Z-H stegs svars metod.

| Regulator | $K_c/K_u$ | $T_i/T_u$ | $T_d/T_u$ |
|---|---|---|---|
| P | 0.5 | - | - |
| PI | 0.4 | 0.8 | - |
| PID | 0.6 | 0.5 | 0.125 |

**Tabel 2.2** Parametrar för Z-H frekvens svars metod.

Ziegler – Nichols frekvenssvars metod, denna metod justerar $K_c$ värdet med endast en P regulator alltså endast proportionell reglering genom att sätta $T_i = \infty$ och $T_d = 0$. $K_c$ värdet ökas sedan gradvis ända tills systemet börjar oscillera. Värdet på $K_c$ när detta inträffar är $K_u$ och perioden av oscilleringen är $T_u$. Hur parametrarna skall användas för att ställa in PID värdena kan ses i figur 2.4[Åström06].

De två Ziegler – Nichols design metoder jag beskrivit är konstruerade för att ge bra respons vid laststörningar. Metoderna använder tämligen lösa design kriterier som kommer göra regulatorn känslig vilket kommer att leda till att jämviktsläge tar länge att uppnå, det tar alltså relativt lång tid att uppnå referenssignalen. Av tidigare nämda orsaker krävs det också ofta lite manuell design efter användandet av Z-H metoderna [Häggblom09].

# 3. Lösnings ansats

## 3.1 Reglerteknik

Ett återkopplat system är uppbyggt kring en process som skall regleras, en regulator och en sensor(se figur 2.3). Systemet har en styrsignal, som är värdet för processen vilken genererar mätvärdet, systemets utsignal. Referenssignalen är målvärdet för mätvärdet, medan differensen mellan dessa två värden är mätfelet. Mätfelet används som insignal för regulatorn som reglerar styrsignalen som i sin tur påverkar mätvärdet.

Ett återkopplat systems kretslopp ser ut som följande:

1. Sensorn bevakar mätvärdet periodiskt och jämför mätvärdet med referenssignalen för att få mätfelet.
2. Regulatorn gör sedan den åtgärden som krävs baserat på mätfelet.

3. Slutligen ändrar ställdonet styrsignalen med hjälp av regulatorns utsignal.

Jag kommer i detta arbete se ett realtidssystem som ett återkopplat system, där schemaläggaren är ställdonet. Schemaläggaren använder återkoppling för att uppnå ett system man kan förutsäga prestandan fastän systemet påverkas av en oförutsägbar belastning. Såsom tidigare diskuterats kommer jag att implementera en återkopplad schemaläggare med designen som Chenyang m.fl. presenterat.

Viktigaste komponenten i det återkopplade systemet är regulatorn av PID typ, som introducerades i föregående kapitel. Den enkla PID formeln är: $reglering = K_p e(t) + K_i \int e(\tau)d\tau + K_d \frac{de(t)}{dt}$.

Orsaken varför en PID regulator används är att den är lättanvänd och mycket stabil om den ställs någotsånär korrekt. PID reglering kräver inte någon exakt matematisk modell, en PID regulator baserad på uppskattade modeller kan ge fullt tillräckliga resultat. Eftersom de flesta realtidssystem är tämligen komplexa kan de inte bli exakt modellerade, Chenyang m.fl. [Chen99] gör en systematisk stabilitets analys men lämnar bort hur regulatorn skall ställas in, jag tycker att inställningen av regulatorn ären viktigare del och har av den orsaken satt vikten på det(se bl.a. kapitel 2).

## 3.2 Realtidsschemaläggare med återkoppling

För att en reglerteknisk tillämpning skall vara möjlig i vårt realtidssystem så måste återkopplingsvariablerna bestämmas. Variablerna som måste väljas är: styrsignalen, referenssignalen, mätsignalen, mätfelet (som alltså är differensen av referenssignalen och mätsignalen), regulator mekanismen och ställdonet/ställdonen. Efter variablerna valts kan systemet implementeras.

I realtidssystem är styrsignalen vanligen reglerad enligt andelen missade deadlines, hur många av jobbena som i procent inte blir klara innan sin deadline. Andelen missade deadlines används sedan som mätvärde. Styrsignalen måste vara en variabel som påverkar mätvärdet och kan reglera det; jag väljer m värdet i (m,k)-Firm för denna funktion medan Chenyang m.fl. vid utnyttjande av en naken EDF(istället för (m,k)-Firm deadlines) valde önskade processor belastningen. Genom reglering av m värdet kommer regulatorn välja en lämplig mängd jobb som måste accepteras medan övriga jobb behandlas ifall det finns tid. Ett högre m värde i förhållande till k värde kräver att en större andel jobb måste göras medan ett lägre m i förhållande till k ger mindre obligatoriska jobb. Denna approach leder till att systemet alltid har hög belastning, vilket

är önskvärt. Eftersom inga jobb avvisas i sig själv kommer all eventuell degradering från optimalitetsläge att bestå av brister i schemaläggaren och brister i prioritetstilldelningen.

Som en summering av återkopplade system kan man säjas att: systemet kommer att ha en starttillstånd som den opererar från vilket sedan adapteras till den verkliga situationen med hjälp av regulatorn. Regulatorn får mätfelet som insignal och kan med hjälp av vettigt inställda parametrar sedan ändra styrsignalen så att mätfelet minimeras [Chen99].

# 4. En återkopplad (m,k)-Firm schemaläggare med grund i FC-EDF

FC-EDF är ett ramverk för en återkopplad EDF schemaläggare, en mall som ger riktlinjer för hur återkoppling skall implementeras i realtidssystem. Ramverket skapades av Chenyang m.fl. [Chen99] och jag baserar min modell på den med följande ändringar: (1) Jag använder firm deadlines, som innebär att om inte deadlinen möts så är bestraffningen av samma magnitud som belöningen. (2) Jag implementerar inte olika nivåer för service eftersom jag anser att det förskönar resultatet. (3) Jag använder (m,k)-firm deadline garanti för att få förutsägbarare resultat, något som Chenyang m.fl. inte tog hänsyn till.

FC-EDF ramverket illustreras i figur 4.1 diagrammet är tämligen generellt och genom att ta bort delar eller ändra exempelvis schemaläggaren så kan man skapa det återkopplade realtidssystem som önskas. Som exempel kommer jag skippa servicenivå regleraren som tidigare nämndes. Ramverket presenterades ej som något nytt, artikeln [Chen99] är mera en sammanfattning av hur enkelt det är att implementera återkoppling i realtidssystem så att kunskapen om ämnet ökar, möjligen med aningen överoptimistiska resultat. Orsaken varför jag anser att deras resultat kan vara missvisande är p.g.a. deras servicenivå reglerare som jag tror ger en stor förbättring i resultaten speciellt när den tillämpas på ett soft-deadline realtidssystem, med okända parametrar för hur soft-deadlinen tillämpas(till försvar för artikeln så beskrivs allt annat i detalj, men denna soft-deadlines parametrar antas vara publik kännedom fastän den inte är en standard). En jämförelse av resultaten som min simulering ger mot de av Chenyang m.fl. är huvudsyftet med detta arbete samt för att faställa nyttan med ramverket som grund för uppbyggandet av en modell för ett realtidssystem.

## 4.1 Översikt av FC-EDF ramverket

Innan PID regulatorn implementeras måste systemvariablerna som motsvaras i ett återkopplat system väljas (se kapitel 3). Kraven på mitt firm-deadline system kommer vara att hålla en låg nivå av missade deadlines och givetvis en hög systembelastning, men (m,k)-Firm deadlines borde sköta om att belastningsnivån förblir hög även ifall vi skulle avvissa jobb. Därför kommer min styrsignal vara antalet missade deadlines utav alla som prioriterats i kön, närmare bestämt de som har (m,k)-krav som kräver att deadlinen möts. Referenssignalen sätts olika noll för att få så högt $\overline{m}$ värde som möjligt vilket innebär ett effektivt utnyttjande av processorn. Ifall referenssignalen sätts till 0 % så skulle systemet inte sträva till en hög processor användning. När ett system uppnår 0 % deadlines missar men har låg processor belastning, då leder det till att regulatorn med referenssignalen 0 % betraktar situationen som önskvärd. En schemaläggare med en referenssignal > 0 %, kommer dock alltid att försöka överbelasta systemet så mycket som referenssignalen ber om.

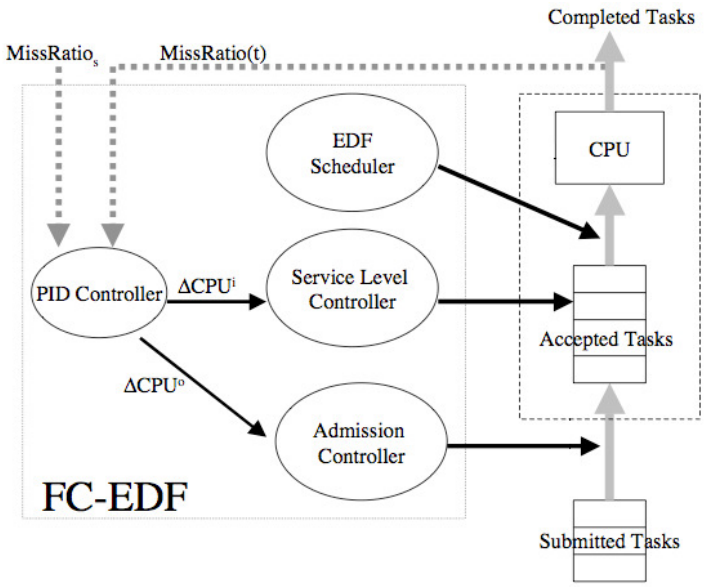
Det är omöjligt att uppnå 100 % belastning och 0 % missade deadlines i ett oberäknerligt system och det finns alltid en dragkamp mellan belastningsgrad och andelen missade jobb. Det finns två approacher för att sköta detta den pessimistiska approachen, som alltid undviker deadline missar på kostnad av processor belastningen. Detta används ofta i realtidssystem med värsta möjliga exekveringstid antaganden för schemaläggning av jobb. Sedan finns också den optimistiska approachen som Chenyang m.fl. använder och jag anser att jag måste använda den i någon mån vid skapandet av en återkopplad schemaläggare. Orsaken varför schemaläggaren måste vara optimistisk är för att implementera en referenssignal på basen av missade deadlines blir vi tvungna att acceptera en viss procent deadlines som inte blir mötta, med icke-noll kravet som förklarades tidigare. Detta är varför implementering av ett soft deadline system är att föredra framom ett hard realtidssystem (se figur 2.1)[Chen99].

Sedan väljs styrsignalen, det önskade $\overline{m}$ värdet ifrån (m,k) kriteriet används som styrsignal. $\overline{m}$ i (m,k) är minsta antalet instanser av varje k som bör mötas [Hamd95] [Chen99]. Idén är att vid ingen belastning kan EDF:n garantera 0 % missar och när belastningen ökar så kommer jobb instanserna delas in i sådana prioritetsnivåer att de kommer klara (m,k)-firm kriterierna.

Som tredje steg måste ställdonet (eng. actuators) som återfinns i FC-EDF ramverket designas som används för att reglera den önskade belastningen. Det finns en prioritets-

tilldelare (ursp. Admission controller) och en servicenivå reglerare i FC-EDF ramverket. Jag valde dock att inte implementera servicenivå regulatorn för att jag inte vill ha olika delar inom varje instans som tillåter ändrad exekveringstid. Prioritetstilldelaren reglerar arbetsmängden som kommer in i systemet, eller i mitt fall reglerar mängden som måste göras enligt (m,k) kriteriet.

FC-EDF ramverket är uppbyggt av en PID regulator, en EDF schemaläggare, (servicenivå regulator) och en prioritetstilldelare (ursp admission controller) se figur 4.1.




**Figur 4.1** FC-EDF Ramverket

Mätfelet kommer kontinuerligt att matas tillbaka till PID regulatorn. PID regulatorn räknar sedan ut storleken på styrsignalen, minskning eller ökning av $\overline{m}$ värdet; $\overline{m}$ ökar vid underbelastning och minskar vid överbelastning. Prioritetstilldelaren kallas sedan för att verkställa ändringen av m värdet. Detta utgör återkopplings loopen i schemaläggnings-systemet. Efter ändrat $\overline{m}$ värde så kommer EDF schemaläggaren att sortera jobb instanserna i de olika prioritetsnivåerna som uppstår med hjälp av EDF algoritmen [Stan99].

## 4.2 Jobb modell

Jobb modellen antar att alla jobb har firm-deadlines (se kapitel 2) och är oberoende. Till skillnad från modellen som implementeras i FC-EDF ramverket där de valt soft deadlines baserat på ”the imprecise computation model” [Liu99][Chen99]. I min modell uppstår varje jobb ett logiskt antal gånger som instans av jobbet och varje instans av jobbet har en unik exekveringstid. Dessutom har varje jobb instans en ready time och en deadline.

## 4.3 PID regulatorn

PID regulatorn är delen som tar hand om återkopplingen, vilket gör den den viktigaste delen av FC-EDF ramverket. Den tar emot mätfelet så att den kan tvinga tillbaka andelen

missar till referenssignalens värde. Regulatorn tar periodiska mätningar med mätperioden SP(*eng. Samplings Period*), räknar ut styrsignalen med följande formel som är en approximation av den som presenterades i kapitel 2.4.2.

$$\bar{m}(t) = \bar{m}(t) + \Delta\bar{m}(t),$$

$$\Delta\bar{m}(t) = C_p * error(t) + C_i(\sum_{t-IW}^{t} error(i))/IW + C_d \frac{error(t) - error(t - DW)}{DW}$$

Var felet är $error(t)$ = referenssignal – miss ratio (andel missar av alla exekverade jobb). SP, $C_p$, $C_i$, $C_d$, IW och DW är ställbara PID parametrar. SP är mätperioden, PID regulatorn kallas efter varje SP tidsenhet. $C_p$, $C_i$, $C_d$ är PID regulatorns koefficienter. IW är integraltidsintervallet under vilken vi får ett medelfel, Chenyang m.fl valde här att summera felen istället för att använda en integral vilket jag inte förstår idén med eftersom en integral är så enkel att implementera och dessutom ger bättre resultat under de första IW tidsstegen. DW är intervallet för derivata felet som innebär att ändringen för DW steg tillbaka används som derivata felet; detta värde bör väljas så litet som möjligt eftersom en riktig derivata skall ha ett DW nära noll. $\Delta\bar{m}(t)$ är PID utsignalen vilket innebär att $\Delta\bar{m}(t) > 0$ ökar belastningen genom att kräva ett högre $\bar{m}$ värde medan $\Delta\bar{m}(t) < 0$ minskar belastningen av processorn.

När $\Delta\bar{m}(t)$ kalkylerats så kallar PID regulatorn på prioritets tilldelaren för att uppdatera $\bar{m}(t)$ med formeln: $\bar{m}(t) = \bar{m}(t) + \Delta\bar{m}(t)$, om jag valt att ha flera servicenivåer så skulle även servicenivå regulatorn kallas på här [Chen99].

Inställningen av PID koefficenterna görs med Ziegler – Nichols frekvenssvars metod, som disskuterades i kapitel 2.4.3. Metoden går ut på att få mätvärdet att oscillera, man tar sedan oscilleringsperiodens värde samt $C_p$ värdet som användes för att få oscilleringen och ställer in PID regulatorn med hjälp av tabell 2.2.

## 4.4 Prioritets tilldelaren

Prioritetstilldelaren ändrar $\bar{m}$ värdet som avgör hur många instanser av ett jobb som bör fullföljas. Den bestämmer prioriteten som kontrollerar vilka jobb som skall exekveras, när antalet mötta deadlines bland de k senaste instanserna av jobbet sjunker mot $\bar{m}$ då stiger även prioriteten. Jag förklarade närmare hur $\bar{m}(t)$ juseras i föregående delkapitlet om PID regulatorn i ramverket [Stan99].

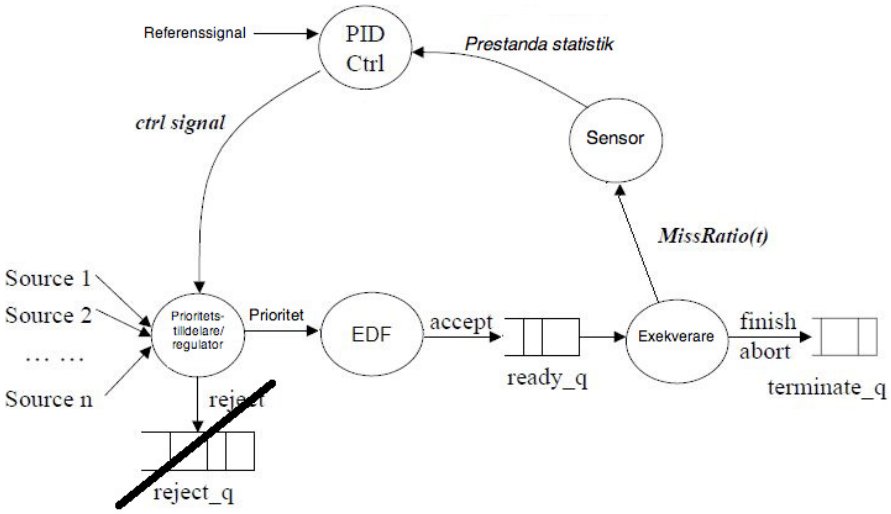
## 4.5 Kritik av FC-EDF

I [Chen99] användes en Soft deadline EDF som jag anser ger missvisande resultat eftersom de väljer att jämföra sin färdiga återkopplade EDF med soft deadline mot en soft deadline fast jag inte hittar några belägg på att soft deadline system används annat än i teorin eftersom den vid överbelastning skulle samla jobb i en stor kö som bara ökar i storlek (enl. min egen tolkning), alltså är eventuell avvisning av jobb mycket oklar i deras EDF.

# 5. Simulering och resultat

## 5.1 Simulerings modellen

Simuleringen är ett firm realtidssystem som körs på ett singel processor system, realtidssystemet är uppbyggt på en firm deadline EDF schemaläggare med återkoppling. System belastningen utgörs av periodiska oberoende jobb; jobbens deadlines är de samma som deras perioder och en ny instans av varje jobb anländer efter dess deadline. Simulatorn (figur 5.1) har sex komponenter: källorna som genererar jobb instanserna; prioritetstilldelaren som bestämmer vilken prioritet jobb skall få; verkställaren (eng. Executor) som modelerar körningen av jobben; sensorn som periodiskt plockar data ur systemet; PID regulatorn som räknar ut styrsignaler för att ändra systemets beteende; EDF schemaläggaren som schedulerar jobbena i dess respektive prioritetskö efter prioriteterna som får från prioritets tilldelaren.




**Figur 5.1** En återkopplad schemaläggare simulator [Chen99]

## 5.2 Belastnings modell

Varje källa (jobb) har en period {P}, deadlinen för jobbet är samma som perioden. Jobbet har en värsta möjliga exekveringstid (eng. worst-case execution time) {WCET}, en bästa möjliga (eng. best-case) {BCET},en uppskattad exekveringstid (eng. estimated) {EET} och en medel exekveringstid (average) {AET}. Varje samling instanser, källa, har tupeln (P, WCET, BCET, EET, AET) där EET används i firm schemaläggningen för att kolla om instansen anses ha missat sin deadline, i.e. ifall nuvarande tiden plus EET är större än deadlinen , och AET används vid skapandet av den verkliga exekveringstiden. EET och AET räknas ut såhär:

$$EET = (WCET + BCET) * 0.5$$
$$AET = EET * etf$$

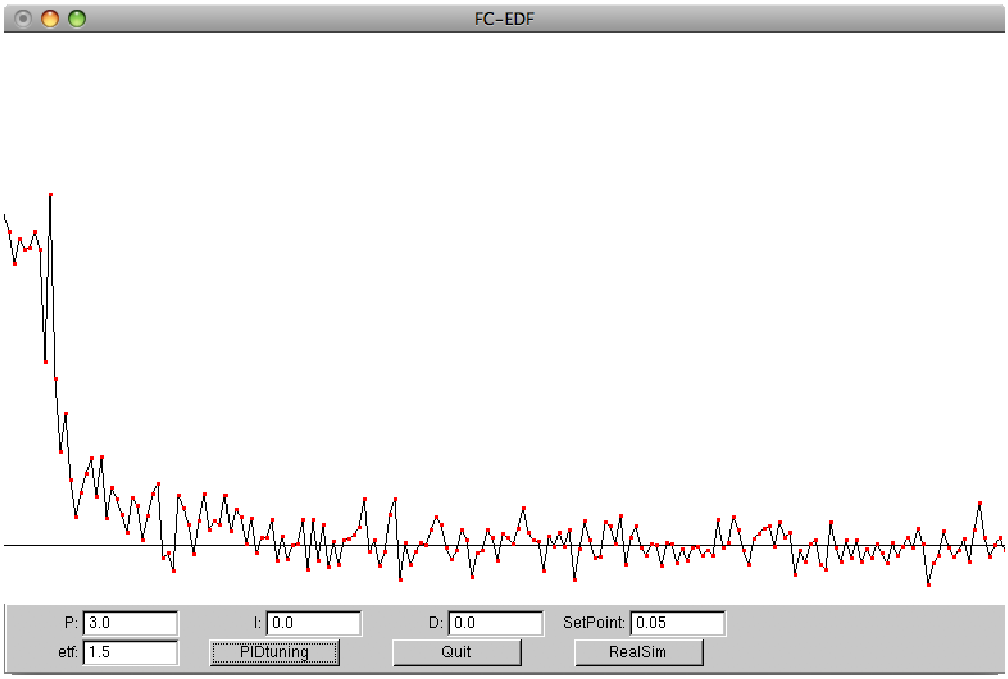
Etf värdet är exekverings tidsfaktor, som kan ställas in för att ändra riktigheten på EET värdet. Om etf värdet är likamed 1.0 då är den uppskattade tiden den samma som AET tiden,med ett större etf (etf > 1.0) så blir närmar sig exekveringstiden WCET som utan ett återkopplat system kommer att leda till överbelastning och med ett etf < 0 kommer exekveringen att bli kortare än beräknat vilket leder till lågt belastade system. Exekveringstiden, som är okänd för schemaläggaren, är beräknad som en uniform variabel i intervallet [AET, WCET] eller [BCET, AET] valt med en *Bernoulli* trial[Math10] med sannolikheten (AET-BCET)/(WCET-BCET). I experimentet utgörs källorna av 40 jobb som är periodiska, var WCET = 4 * BCET. Etf kan då endast befinna sig i intervallet [0,5:1,5]. WCET och P skapas användande jämn fördelning: WCET = uniform(5, 10), P = WCET * uniform(10, 15). P borde sedan justeras så att 31 av jobbena är harmoniska med den gemensamma nämnaren 2400 tidsenheter, detta visar sig dock förstöra förutsägbarheten enligt mina experiment så jag väljer att låta jobbena vara som de först skapas. Detta är samma källor som används i artikeln där FC-EDF ramverket presenteras och kommer att användas i experimentet . Etf värdet kan ändras under experimentets gång och mellan olika experiment. Alla jobb anländer vid tidpunkten noll för att överbelasta systemet. De jobb som blivit nedprioriterade eller vägrade kommer att försöka pånytt när ett nytt P startas.

## 5.3 Implementationen

Inställningarna för simuleringen listas i tabel 5.1. Referenssignalen är mål missration, vilken är antalet missade deadlines som andel av alla prioriterade jobb i procent, den kan ändras beroende på toleransen för missade deadlines. SP var ursprungligen tänkt

som en minsta gemensamma nämnare för majoriteten av jobbena, men eftersom jag valde att inte göra jobben harmoniska så får SP endast uppgiften som samplings-perioden för PID regulatorn. IW är mängden gamla mätningar som regulatorn behöver minnas. DW är perioden mellan två mätfel som utgör derivata delen i PID regulatorn. $\overline{m}$ och k värdena är från (m,k)-firm deadline och jag använder samma värden för alla 40 jobb.

$C_p$, $C_i$ $och$ $C_d$, är PID koefficienterna, som behöver ställas in på ett stabilt sätt för att styra schemaläggaren. Jag implementerade ett grafiskt användargränssnitt för simulatorn så jag kunde ställa in PID regulatorn(figur 5.2). Jag menade ursprungligen att använda Åström-Hägglunds metod[Häggblom07], som är en förbättring av Ziegler-Nichols frekvenssvars metod, men mätvärdesperioden SP visade sig vara för lång för att ge de fina mjuka kurvor som behövs för ett noggrant frekvenssvar så PID koefficienterna kan räknas ut. Därför valde jag Ziegler-Nichols mindre noggranna frekvenssvars metod där variationen i mina mätfel gjorde mindre skada. Jag ställde alltså in regulatorns P tills jag fick hyfsad oscillering och använde sedan tabellen i kapitel 2 för en grundinställning som jag använde som utgångspunkt för att finjusteringar. På grund av det långa SP mätvärdet så blev derivatadelen för slumpmässig därför valde jag slutligen en PI-regulator för att få ett stabilare system.




**Figur 5.2** Mitt användargränssnitt för att ställa in PID koefficienter

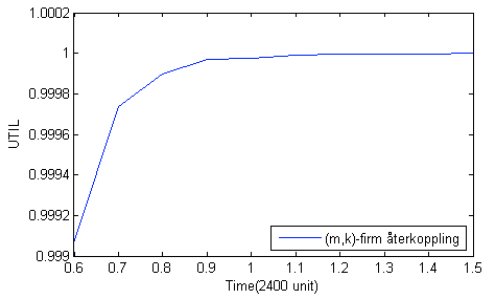
| | |
|---|---|
| Referenssignalen | 0.02 |
| SP | 2400 (tidsenheter) |
| IW | 100 (SP) |
| DW | 1 (SP) |
| $C_p$ | 1.4 |
| $C_i$ | 0.55 |
| $C_d$ | 0.0 |
| $\overline{m}$ (startvärde) | 20 |
| K | 20 |

**Tabel 5.1** Simuleringsparametrarna

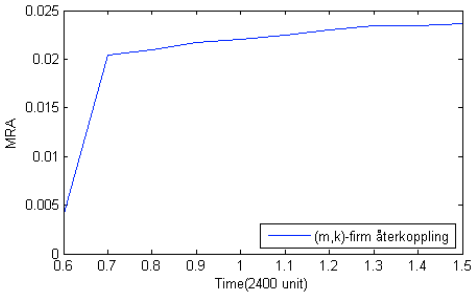
## 5.4 Prestanda mätningsparametrar

- MRA: antalet missade deadlines bland de antagna jobbena (*eng. missratio among admitted tasks*). Ett firm-deadline system försöker ha så få missade deadlines som möjligt men några missar är godtagbara eftersom straffen är i samma storleksordning som belöning. Jag byggde upp min modell kring missration så är MRA den viktigaste kriterien.
- UTIL: processor belastningen (utilization) är det andra huvudkriteriet, hur väl processorn belastas kombinerat med missration är prestanda parametrarna i denna jämförelse.

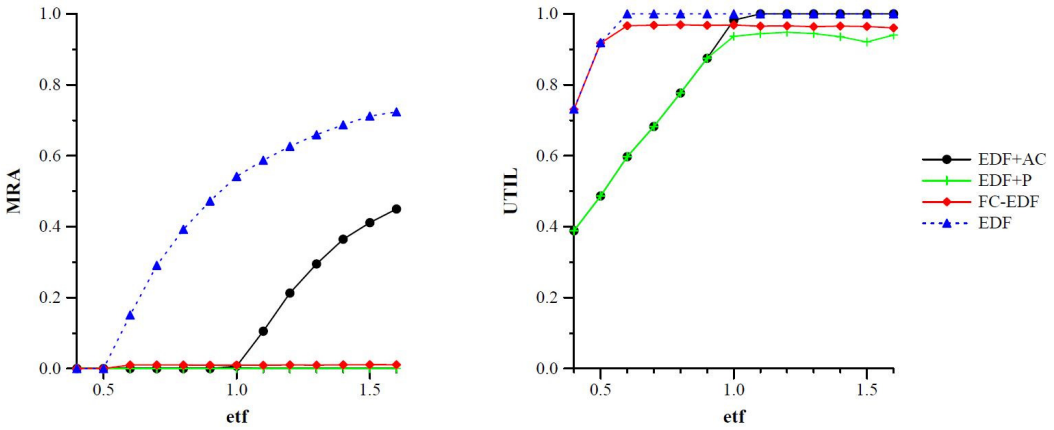
## 5.5 Första experimentet:"Steady execution time" [Chen99]




**Figur 5.3** UTIL vs etf




**Figur 5.4** MRA vs etf




**Figur 5.5** MRA vs. etf och UTIL vs. etf resultat från artikeln som presenterade FC-EDF ramverket [Chen99]

Det första experimentet visar prestandan för simuleringen med olika etf värden när EET värdet är konstant, EET är tiden schemaläggaren tror en exekvering kommer att ta. Etf värdet är konstant för varje körning som är 1200 SP enheter lång, men den riktiga exekveringstiden kommer att variera i varje jobb instans genom den slumpvisa fördelningen(se kapitel 5.2). Algoritmen kördes 10 gånger med etf värdena [0,5:1,5] med 0,1 som steglängd. Orsaken för detta experiment är att i verkliga applikationer så är den exakta exekveringstiden okänd och uppskattningar av den kan variera, vilken är orsaken att worst case (värsta fall) antaganden används i vissa realtidssystem där deadline missar inte tolereras. Worst-case antaganden är dock ganska ineffektiva ifall processor belastning slösas vid en lägre exekveringstid än väntat (etf << 1.0) om systemet är överbelastat vill säga.

Det första experimentets resultat illustreras i figurerna 5.3–5.4 medan figur 5.5 är resultaten gjorda av Chenyang m.fl. [Chen99]. Varje punkt i figurerna är från en körning av 1200 SP enheter (2880000 tidsenheter) som kördes 30 gånger för att få bättre tillförlitlighet i resultaten. Exekveringstiden i simuleringen har endast en decimals noggrannhet eftersom simuleringen gör som en loop som genomgås 2880000 * 30 * 10 = 864 000 000 vilket på testdatorn tog ungefär 40 timmar, enda skillnaden blir dock att belastningen i min simulering blir marginellt mindre än med en som innehöll flera decimaler så det krävs alltså ett högre *etf* för full belastning. Noggrannheten i den ursprungliga simuleringen är okänd, men det kan mycket väl vara att de valde heltal vilket skulle göra min noggrannare än deras.

Figur 5.3 visar tydligt belastningsfördelen med att behålla alla jobb istället för att avvisa lågprioriterade jobb, som tidigare nämnts ger min modell högre prioritet till de jobb som är nära att missa (m,k)-firm kriteriet istället för att som i FC-EDF godkänna nya jobb

endast på basen av när de anländer till systemet. Nackdelen med mitt system blir dock att när det finns många jobb i kön så krävs det ett större minne i systemet och mera processorkraft att sätta till och ta ut jobb från kön.

Andelen missar i Figur 5.4 är direkt jämförbart med resultaten i Figur 5.5 av Chenyang m.fl. dock med fördelen att (m,k)-firm deadlines baserade simulering har god förutsägbarhet i avseende på vilka jobb som exekveras. I den första körningen med *etf* på 0,6 så är belastningen av systemet såpass mild att referenssignalens 0,02 inte uppnås men i övriga belastningsfall kan man se att andelen missar matchar referenssignalen mycket bra.

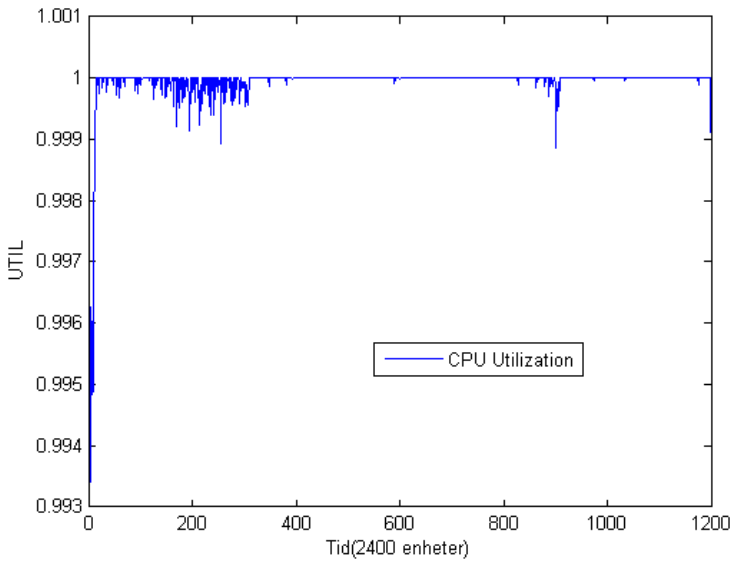
Bör nämnas att av alla figurer i 5.5 så är det endast FC-EDF jag jämför simuleringen med, men deras EDF kan också vara intressant att titta på.

## 5.6 Andra experimentet:"Dynamic execution time" [Chen99]

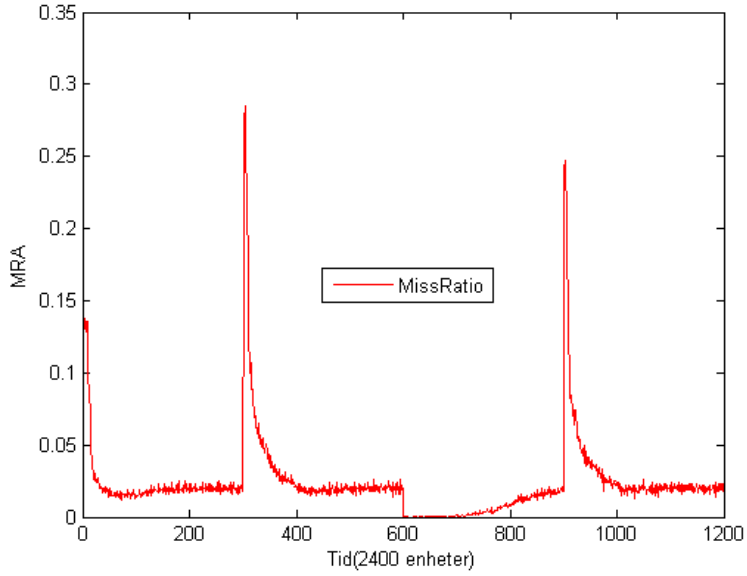
I det andra experimentet ändras *etf* värdet varje 300 SP tidsenhet för att studera vad som händer när AET (eng. Average estimated time) per jobb ändras drastiskt. De olika *etf* värdena kan avläsas från tabel 5.2.

| Interval(SP) | 0-300 | 300-600 | 600-900 | 900-1200 |
|---|---|---|---|---|
| *etf* | 0,8 | 1,3 | 0,8 | 1,2 |

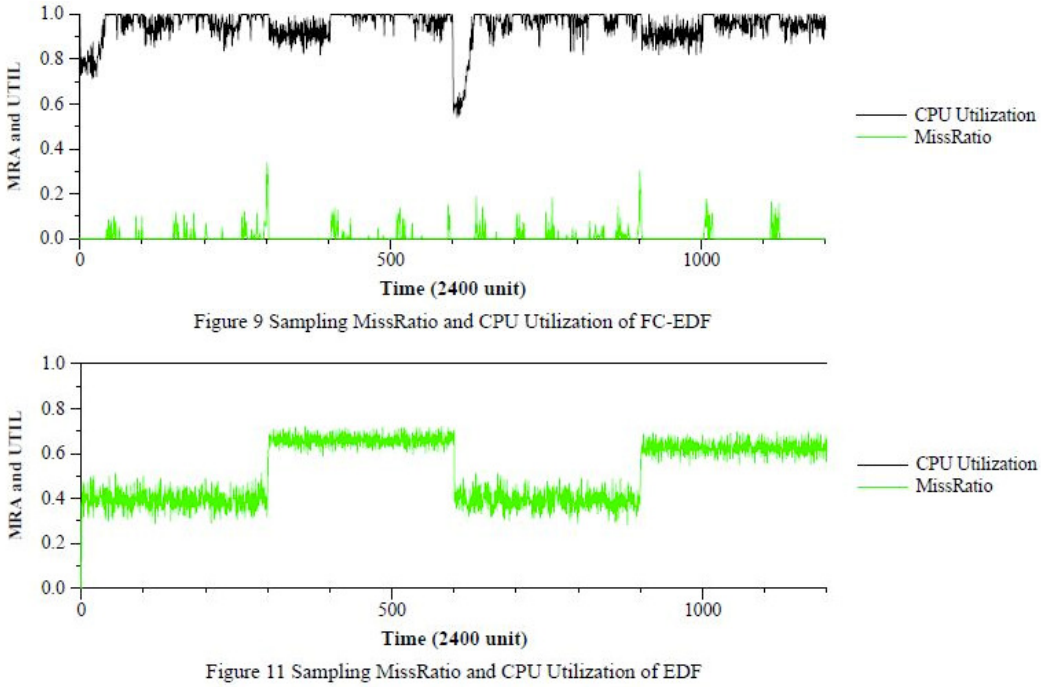
**Tabel 5.2** de olika *etf* värdena som används i det andra experimentet




**Figur 5.6** Processor belastningen i simuleringen




**Figur 5.7** Andelen missar för det andra simuleringsexperimentet




**Figur 5.8** MRA och UTIL från FC-EDF artikeln [Chen99]

Figur 5.6–5.7 illustrerar processorbelastningen och andelen missar i procent. Angående figur 5.6 kan man bara konstatera att belastningen alltid är ytterst nära 100 %. I figur 5.7 kan man tydligt se att ett *etf* värde på 0,8 (som simuleringen har under de 300 SP första enheterna) leder till en överbelastning på ungefär 14% som visar sig i missade deadlines,

vilket regulatorn reagerar snabbt på och reducerar antalet missar mot referenssignalen som är 0.02 (2% missar).

När *etf* värdet byter från 0,8 till 1,3 vid tiden 300 SP, utgörande en belastningsökning på 62 %, då kan man se en topp i kurvan som representerar tillfällig överbelastning som efter 80-90 SP återgår till fortfarighets tillstånd. Orsaken att det tar relativt länge för kurvan att återvända till måltillståndet är dels att integralvärdet är rätt så stor och dels för att jag endast implementerat 20 k (se (m,k)-firm), vilket leder till att små fel tar längre att justera än fallet skulle vara med ett större k värde eftersom k endast antar heltalsvärden.

Vid 600 SP när belastning minskar till 0,8 igen ser man att andelen missar håller sig vid noll under 150 SP enheters tid vilket innebär att regulatorns integralverkan får systemet att överreagera som i sintur leder till att förutsägbarheten under denna tidsperiod blir lidande.

De sista 300 SP tidsenheterna visar ett motsvarande beteende när systemet igen får en kraftigt större belastning, denna gång från *etf* värdet 0,8 till 1,2.

I figur 5.8 av FC-EDF ramverket kan man se mera regelbundet återkommande missar av deadlinen, dessa tror jag i alla fall delvis beror på att harmoniska jobb användes i ursprungliga FC-EDF simuleringen samt på att de använder derivata termen för sin regulator som visade sig opålitlig i min simulering när mätintervallet är såpass stort i simuleringen. Något som hjälper göra dessa toppar så korta är servicenivå regulatorn som Chenyang m.fl. implementerat vilken bara kan tillämpas i realtidssystem där varje jobb instans kan delas upp i mindre enheter.

Nedre grafen i figur 5.8 visar hur soft deadline EDFen som FC-EDFen bygger på uppför sig och man ser tydligt att antalet missade deadlines inte tyglas av någon regulator.

# 6. Slutsatser och disskussion

I detta arbete har jag velat undersöka svårigheten att implementera återkoppling med hjälp av riktlinjerna som ges från FC-EDF ramverket. Jag valde dock att som komplement till EDF schemaläggaren implementera (m,k)-firm deadlines som leder till att olika prioritetsnivåer skapas dynamiskt. Olika prioritetsnivåer leder till en stor förbättring ifrån FC-EDFs orginal implementering, FC-EDF vars syfte enligt skaparna[Chen99] är att öka förutsägbarheten för systemet, men som endast ökar förutsägbarheten

belastningmässigt fast inte med tanke på vilka jobb som utförs. Chenyang m.fl. adresserar detta med att rekommendera statiska prioritetsnivåer som ger de viktigaste jobbena förtur, detta leder dock till att de lägre prioriterade jobbena rätt så slumpmässigt exekveras. För att återgå till (m,k)-firm deadline implementeringen så ger den möjligheten att låsa högprioriterade med samma $\overline{m}$ som k värde, vilket betyder att alla instanser skall exekveras, medan mindre viktiga jobbs $\overline{m}$ regleras av regulatorn vilket leder till att systemet är förutsägbart även vid överbelastning.

Jag valde i min simulering att aldrig förkasta några jobb som ej missat sin deadline vilket kan leda till en stor overhead men eftersom realtidssystem ofta har ett relativt begränsat antal olika jobb och simuleringen endast innehåller 40 stycken jobb så anser jag att det är en bra lösning för att uppnå en ständigt hög processor belastning.

När man ser på resultaten från första experimentet så kan man se att min modell betedde sig näst intill optimalt med små avvikelser från referenssignalen. Min referenssignal på 2 % gentemot FC-EDFs referenssignal på 0,01 ledde dock till att jag fick lite högre andel missar. Orsaken att jag ändå valde en högre referenssignal var: om jag sätter den lägre så sjunker andelen missar tidvis under noll vilket medför att förutsägbarheten jag eftersträvar minskar. FC-EDF gjorde också bra ifrån sig i detta test dock med en viss fri processorkraft, men med tanke på att FC-EDF använder harmoniska jobb så tycker jag FC-EDF ger bättre resultat, eftersom resultaten faktiskt gäller i alla situationer. Jag anser dock att nyttan med dom fina resultaten kan ifrågasättas när förutsägbarheten är begränsad.

I det andra experimentet så reagerar min (m,k)-firm återkopplade EDF långsammare än FC-EDF simuleringen gör. Här har dock FC-EDF två fördelar som min modell saknar: (1) FC-EDF använder servicenivåer så att delar av ett jobb kan skalas bort utan straff, (2) den använder en soft-EDF schemaläggare som tillåter att deadlines blir något försenade utan att de missas enligt en okänd reglersamling (Chenyang m.fl. beskrev ej sin soft-deadline EDF och ej heller har jag hittat en allmän beskrivning i litteraturen). Åt andra sidan så valde jag implementera (m,k)-firm återkopplade modellen utan att jobben var harmoniska sinsemellan, vilket skapade en jämnare belastning för mitt system. Skillnaderna i resultat är främst att FC-EDF visade snabbare normalisering till referenssignalen medan (m,k)-firm systemet hade en högre belastning.

Jag byggde upp min (m,k)-firm återkopplingsmodell med målet att få ett återkopplat realtidssystem som är pålitligt även under överbelastningssituationer, detta är något jag

anser Chenyang m.fl. missade med deras eget system. Deras ramverket är dock ett välbeskrivet dokument som passar bra som stomme för en egen implementering av återkoppling i realtidssystem.

Bör dock noteras att liksom orginal FC-EDF systemet så är mitt (m,k)-firm baserade system oborstat och jag kan tänka mig att man förbättra det. Exempelvis genom att avvisa jobb som har låg prioritet då kunde man få ännu bättre resultat, detta på en viss bekostnad av processor belastningen men jag tog ej heller i beaktande den tid overhead som processorn slösade på jobb som senare missade sin deadline i simuleringen, så vad skillnaden egentligen skulle vara kunde vara föremål för framtida experiment. En annan sak som troligen skulle förbättra förutsägbarheten av systemet är att byta ut prioritets-tilldelaralgoritmen DBP mot andra möjligtvis bättre algoritmer[Lanying08][Baccouche07].

# 7. Källor


[Murthy01] G. Manimaran och C. Siva Ram Murthy, ”Resource Management in Real-Time Systems and Networks”, Massachusetts Institute of Technology 2001

[Chen99] Chenyang. Lu, J.A. Stankovic, G. Tao, och S.H. Son, "Design and Evaluation of a Feedback Control EDF Scheduling Algorithm", In Proc. Real-Time Systems Symp. pp.56-67, 1999.

[Hamd95] M. Hamdaoui och P. Ramanathan, “A Dynamic Priority Assignment Technique for Streams with (m,k)-Firm Deadlines”, IEEE Trans. Computers, vol. 44, no. 12, pp. 1443-1451, Dec. 1995.

[Åstrom06] Karl J. Åström och Tore Hägglund, ”Advanced PID control”, ISA 2006.

[Wiki10] http://en.wikipedia.org/wiki/PID_controller#Laplace_form_of_the_PID_controller

[Stan99] J.A. Stankovic, Chenyang. Lu, S.H. Son, and G. Tao, “The case for feedback control real-time scheduling”, Proceedings of the 11th Euromicro Conference on Real-Time Systems, 1999.

[Liu99] J. W. S. Liu, et. al., “Algorithms for Scheduling Imprecise Computations”, *IEEE Computer*, Vol. 24, No. 5, May 1991.

[Math10 ] http://www.math.uah.edu/stat/bernoulli/index.xhtml

[Häggblom07] Häggblom, Kurt-Erik and Böling Jari, “Reglerteknik”, https://www.abo.fi/student/tkfrtrt1

[Lanying08] LI Lanying och TAN Yu, ”Analysis and improvement of scheduling algorithms based on (m, k)-firm constraint”, International Symposium on Computer Science and Computational Technology 2008.

[Baccouche07] Leila Baccouche, Sami Liman, “(m,k) firm Scheduling of Real-Time Transactions with Concurrency Control”, Second International Conference on Systems 2007.

# 8. Appendix (C++ kod från simuleringen)