## MAC usage decay/windows usage decay

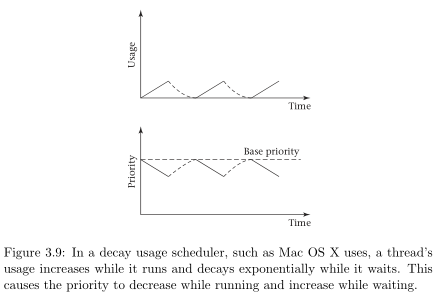
**Decay usage scheduling**

* Brugere vil gerne have god responsiveness med fokus på brugerinput og effektivitet – brugere tænker ikke så ofte i deadlines.
* Idéen er at forhøje prioriteten af ventende tråde og nedprioritere running tråde.
* Bruger round-robin tidsfordeling

**Schedulering på Windows – strengt set ikke usage decay scheduling**

* Prioriteter ændres dynamisk
* Anderledes (modsat) fremgangsmåde ift. OS X Usage Decay Scheduling men ca. samme resultat:
  + En tråd der har ventet får en højere prioritet end en der har været kørende
* Når en tråd bliver runnable:
  + Prioriteten hæves initielt så tråden foretrækkes.
  + Prioriteten falder (lineært) mod basisniveauet mens tråden kører.
* Forskel fra OS X:
  + Bruger ikke eksponentiel decay til at booste ventede tråde, men i stedet får en tråd der har ventet et prioritetsboost alt efter hvad tråden ventede på.
    - Et lille boost efter at have ventet på en disk
    - Stort boost efter at have ventet på brugerinput på keyboard etc.
    - Interaktive tråde hæves mere end simpel I/O
* Brugeren kan angive vigtighed af jobs.
* Man får altså et system der er responsivt fra brugerens synspunkt

**Usage decay scheduling (OS X)**

* Jo længere en tråd har kørt, des lavere prioritet får den (til en vis specifik grænse)
  + Når tråden ikke kører falder dens usage med tiden til nul.
* En tråd der ikke har kørt i et stykke tid, får sin prioritet opjusteret til basisniveauet.
* Tråde der venter på input får opjusteret deres prioritet så de hurtigt får processortid når deres input bliver klart
* Sænker kørende trådes prioritet under basisniveauet og sikrer på den måde at selv lav prioritetstråde bliver kørt fra tid til anden.
* Ventende trådes usage nedscales med 5/8, 8 gange i sekundet og deres prioritet nærmer sig så basisniveauet.  
  Kørende trådes usage scales op ift. hvor stort load der generelt er på CPU’en, dvs. deres prioritet sænkes under basisniveauet.
* Sikrer en høj grad af interaktivitet

**Fordele/Ulemper ved dynamisk prioritet**

* Denne type schedulering er velegnet til at sikre et responsivt system for brugere
* Men præcis styring er vanskeligere
  + Det er ikke det primære mål (der er typisk nok kraft til at jobs bliver udført alligevel)
  + Havde man en server kunne der være tale om at en vigtigere kunde fik større prioritet end de mindre vigtigere kunder. Vi kan ikke fordele ulige processer-opmærksomhed til tråde, med mindre vi piller ved basisprioriteten af trådene (ikke en god idé).