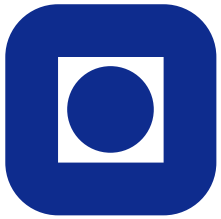


NTNU
Norwegian University of
Science and Technology

Kjernen til operativsystemet

Donn Morrison
Department of Computer Science

With slides contributed by Geir Maribu

Outline

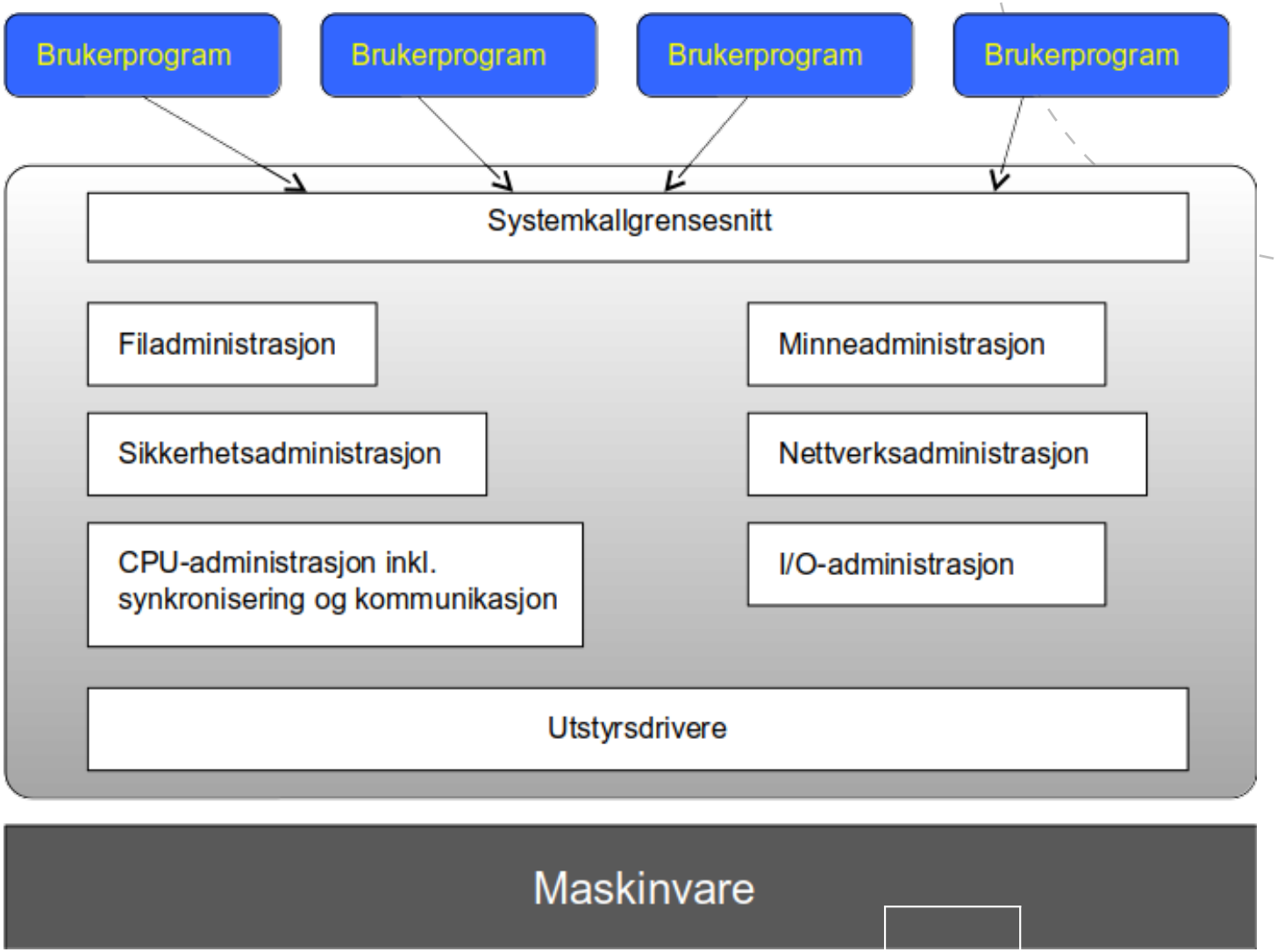
Review

Kernen

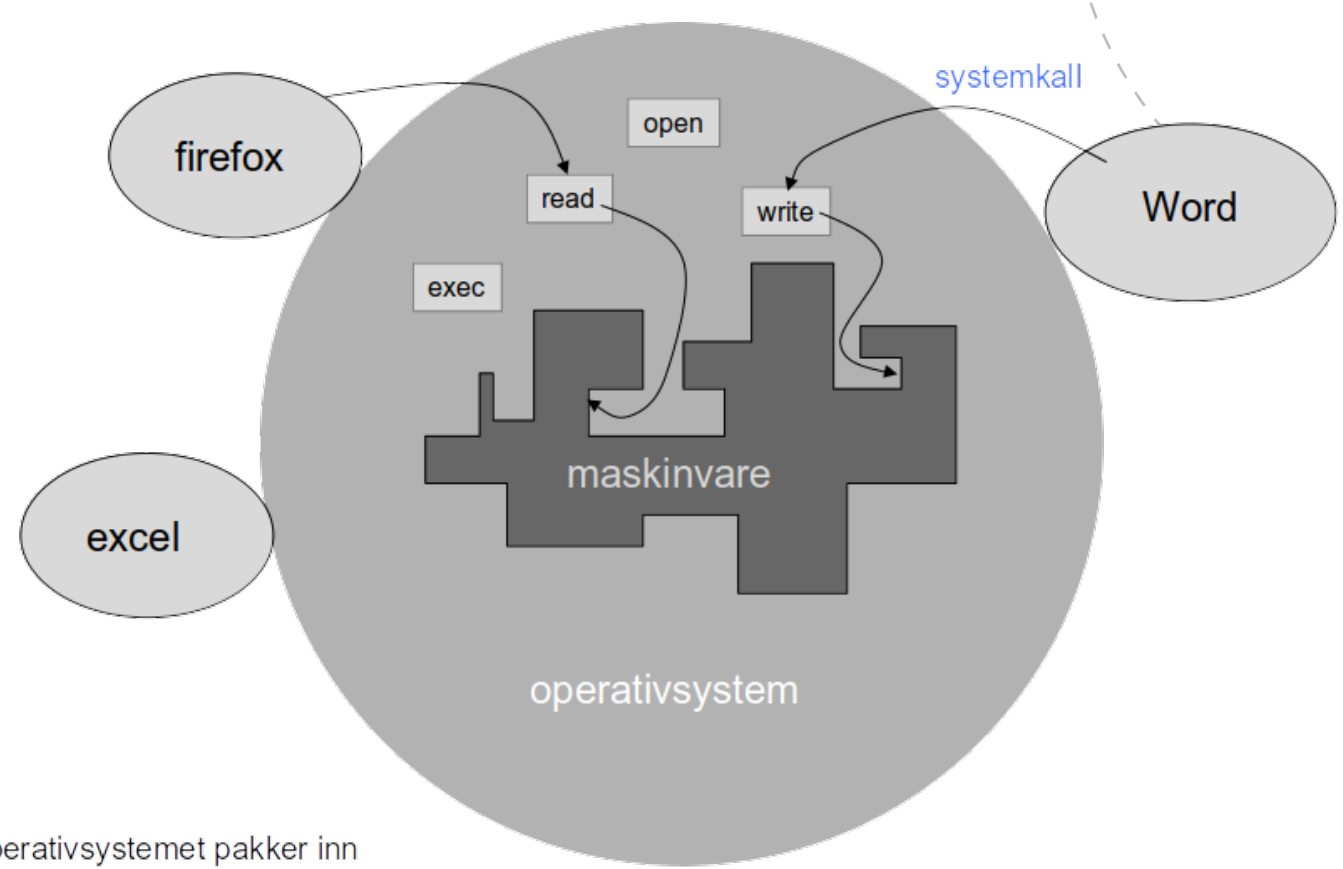
Hva holder opsys på med?

- Administrasjon av I/U-enheter
- Administrasjon av CPU(-er)
  - CPU-kø
  - Synkronisering og kommunikasjon
- Administrasjon av minne
  - Virtuelt minne
- Administrasjon av filer
- Nettverksadministrasjon
- Sikkerhetsadministrasjon

Oversikt


Brukerprogram
Brukerprogram
Brukerprogram
Brukerprogram
Systemkallgrensesnitt
Filadministrasjon
Minneadministrasjon
Sikkerhetsadministrasjon
Nettverksadministrasjon
CPU-administrasjon inkl. synkronisering og kommunikasjon
I/O-administrasjon
Utstyrsdrivere
Maskinvare

Systemkall

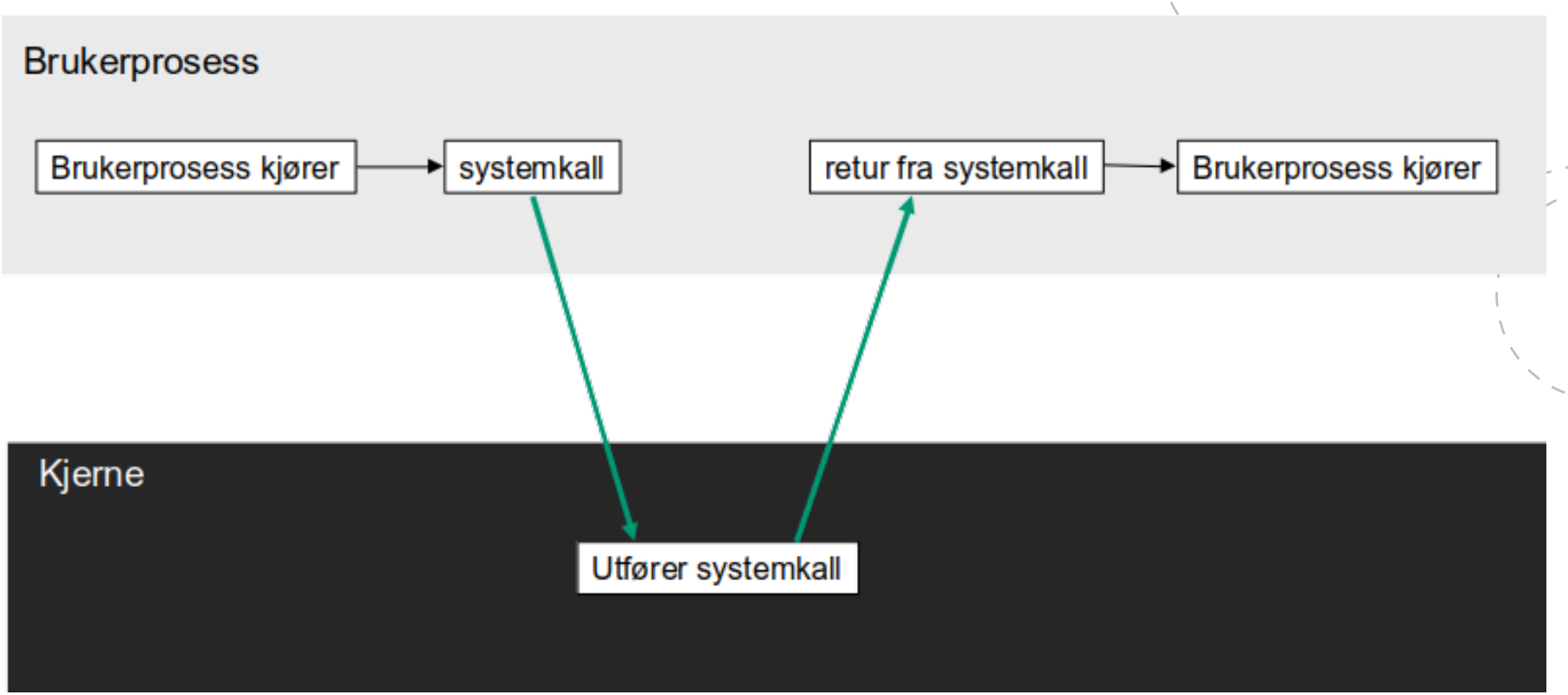


Operativsystemet pakker inn maskinvaren, og vi får et enklere grensesnitt mot maskinen

Systemkall

- Systemkall handler om hvordan et program ber om en tjeneste fra operativsystemet
  - F.eks tilgang til maskinvare, opprette og kjøre proseser, kommunisere med kjernen
- Hvorfor systemkall?
  - Sikkerhet, dvs brukerprogrammer skal ikke på egen hånd få adgang til maskinvare på operativsystemet
  - Systemkallet gir en veldefinert og sikker tilgang til operativsystemets tjenester
- Hvor finner systemkallene?
  - Programmeringsspråkets biblioteker (API-er)

Systemkall

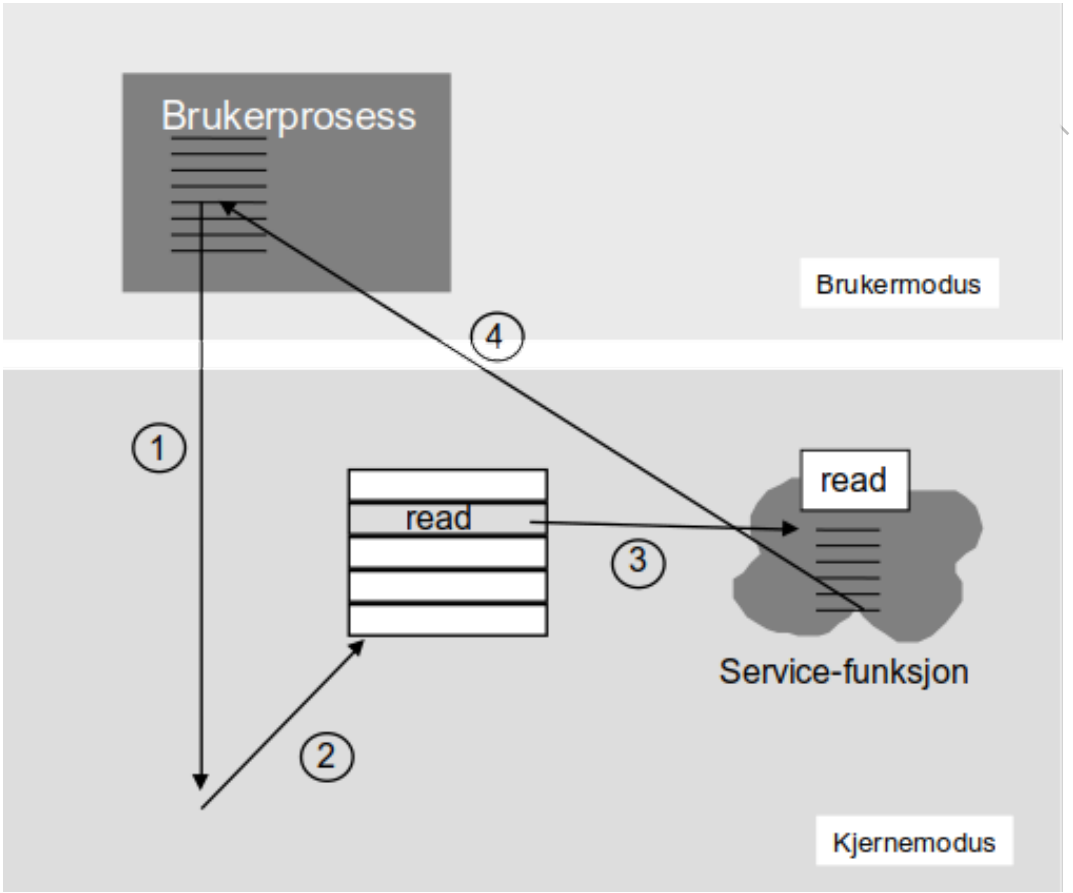

Brukerprosess
Brukerprosess kjører
systemkall
retur fra systemkall
Brukerprosess kjører
Kjerne
Utfører systemkall

Outline

Review

Kernen

Kerne- versus brukermodus

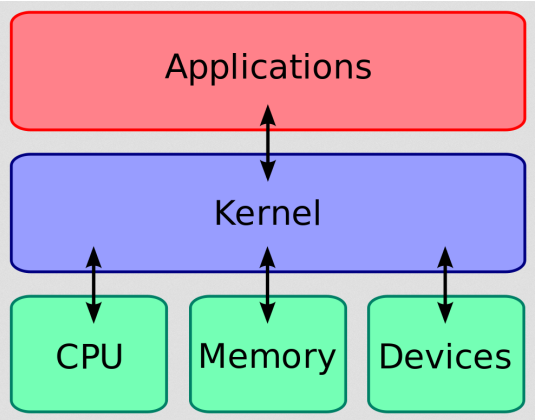


Kerne- versus brukermodus

- Brukermodus
  - Lavt privilegert
    - dvs kan ikke direkte aksessere maskinvare/operativsystemets ressurser
    - Kan bare aksessere egne ressurser (variabler, filer, osv.)
  - Feil krasjer aktuell prosess
  - Privat virtuelt adresseområde
    - Alle prosessene har hver sine adresseområder
- Kjernemodus
  - Høyt privilegert
    - Kan aksessere alle ressurser
  - Feil krasjer hele operativsystemet
  - Ett stort virtuelt adresseområde som omfatter alt
  - Programmeringsspråkets biblioteker (API-er)
- Brukerprogrammene
  - Svitsjer hele tiden mellom brukermodus og kjernemodus
  - Hvordan? Systemkallene

Kjernen

- Når et dataprogram ber om en tjeneste fra kjernen kalles det et systemkall
  - f.eks en I/U-operasjon
- Flere typer kjerne
  - Monolitisk: Operativsystemets egne instruksjoner kjører i sin helhet i samme adresserom, dvs i kjernemodus
  - Mikrokjerne: Operativsystemet er delt opp i prosesser som kjører i vanlig brukermodus. Mer modulært og lettere å vedlikeholde



Hvorfor kjernen?

- Hovedfunksjonen til kjernen er å administrere datamaskinens maskinvare og ressurser og tillate at andre programmer kan bruke disse ressursene
- Typiske ressurser er
  - CPU: Kjernen bestemmer hvilket program som skal få kjøre
  - Minne (RAM): Bestemmer hvor i minnet et program ligger, hvor mye minne, og hva som skjer når minnet er fullt
  - I/U-enheter: tastatur, mus, disker, usb-disker, skjerm, nettverkskort osv: Kjernen tar i mot forespørsler til å bruke utstyret, administrere I/U-køer og tilbyr gode og sikre metoder for å bruke utstyret
- Kjernen tilbyr også synkronisering og kommunikasjon mellom prosesser

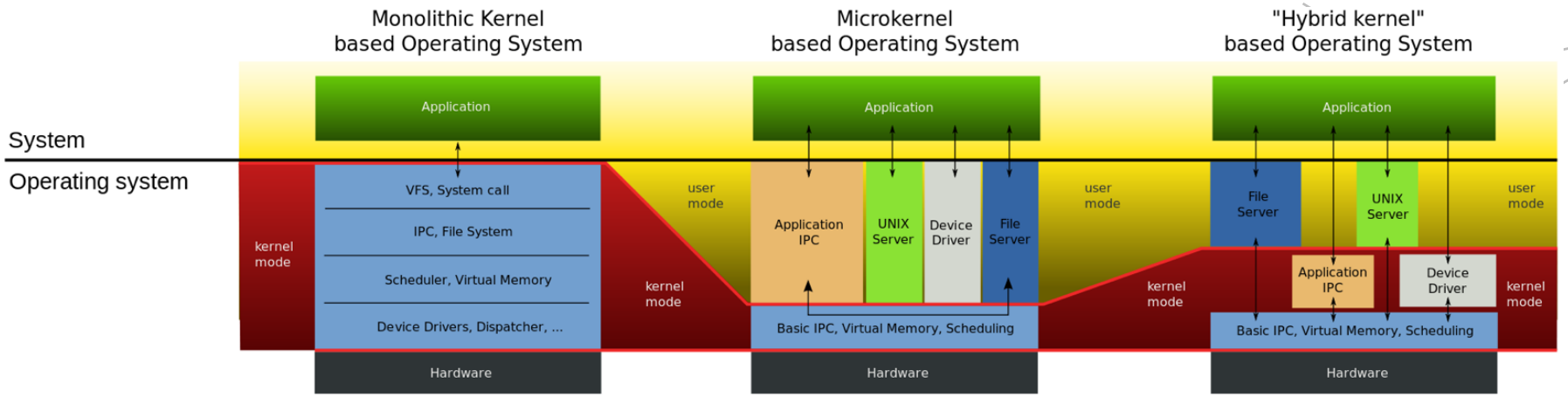
Typer av kjerner

— Monolittisk
- Hele operativsystemet kjører i det samme adresserommet, dvs ett stort program som inneholder hele opsys (inkludert driverne)
- Lettere å implementere sammenlignet med mikrokjerne
- Ulempe: feil i en av komponentene, f.eks en driver vil krasje hele systemet
- Fordel: Færre kodelinjer enn mikrokjerne og derfor raskere
- Eks: Windows 9x (DOS), Unix, Linux, FreeBSD. De to siste kan laste moduler under kjøring

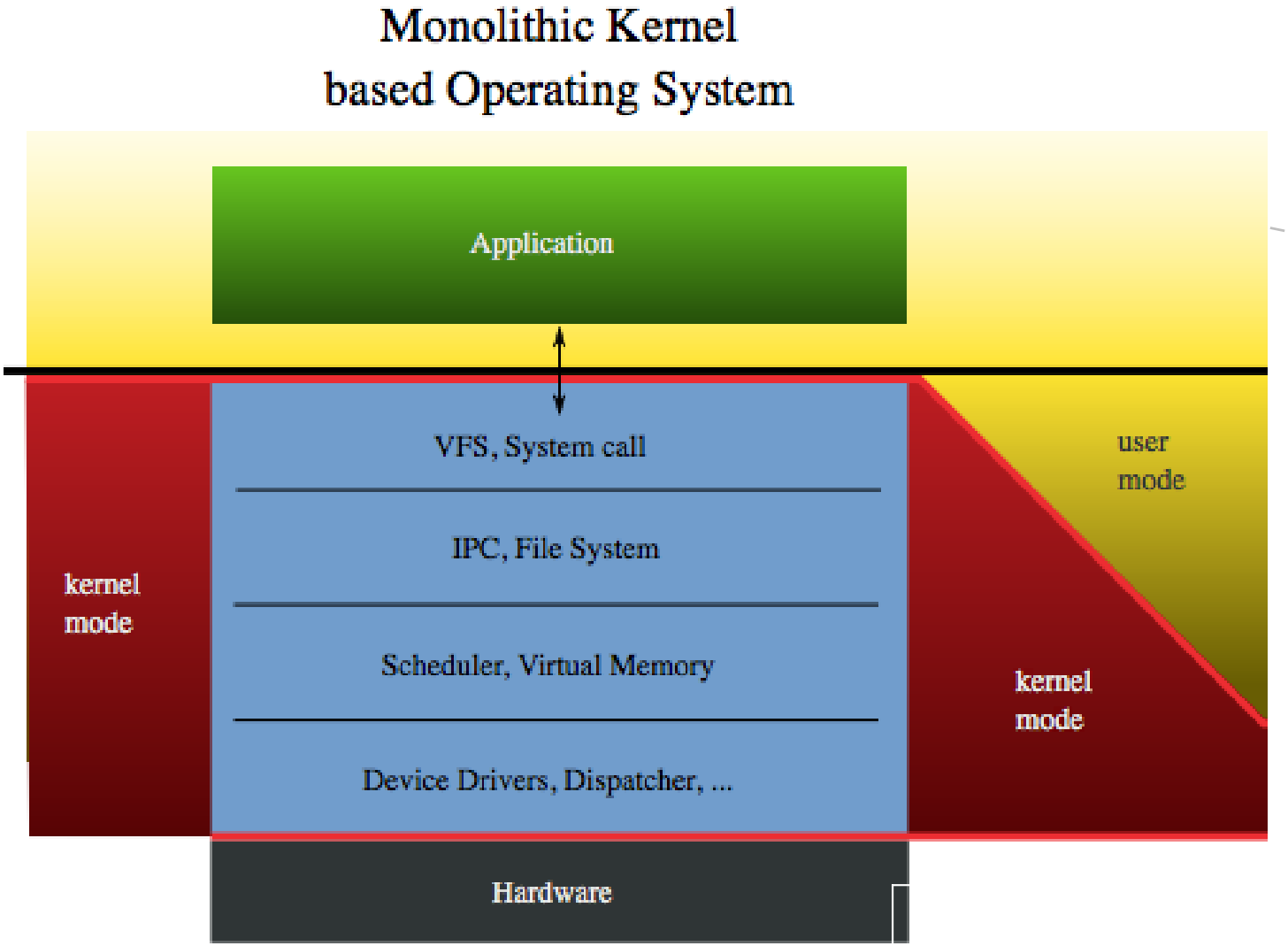
Typer av kjerner

- Mikrokjerne
  - Flere av opsys-tjenestene er flyttet ut av kjernen og til et sett av tjenere som kommuniserer via en minimal kjerne, en såkalt mikrokjerne
  - Minst mulig i kjernemodus og mest mulig i brukermodus
  - Mikrokjerne er enklere å vedlikeholde, men et stort antall systemkall og kontekstskifter kan senke hastigheten til systemet
- Hybridkjerne
  - Ligner på mikrokjerne, men har mer programkode i kjernemodus for økt hastighet
  - Eks: Windows NT, Mac OSX

Typer av kjerner



Monolitisk kjerne 1



Monolitisk kjerne 2

- Alle OS-tjenester kjøres
  - ... som en stk hovedtråd i samme minneområde (adresseområde)
  - Den monolittiske kjernen er ett program som inneholder all programkoden for å utføre alt som har med kjerneoperasjoner å gjøre.
    - Utstyrsdrivere, scheduler, minneadministrasjon, filsystem, nettverksaksess
- Fordeler:
  - Rask og efektiv tilgang til maskinvare
  - Enklere å realisere enn mikrokjerne

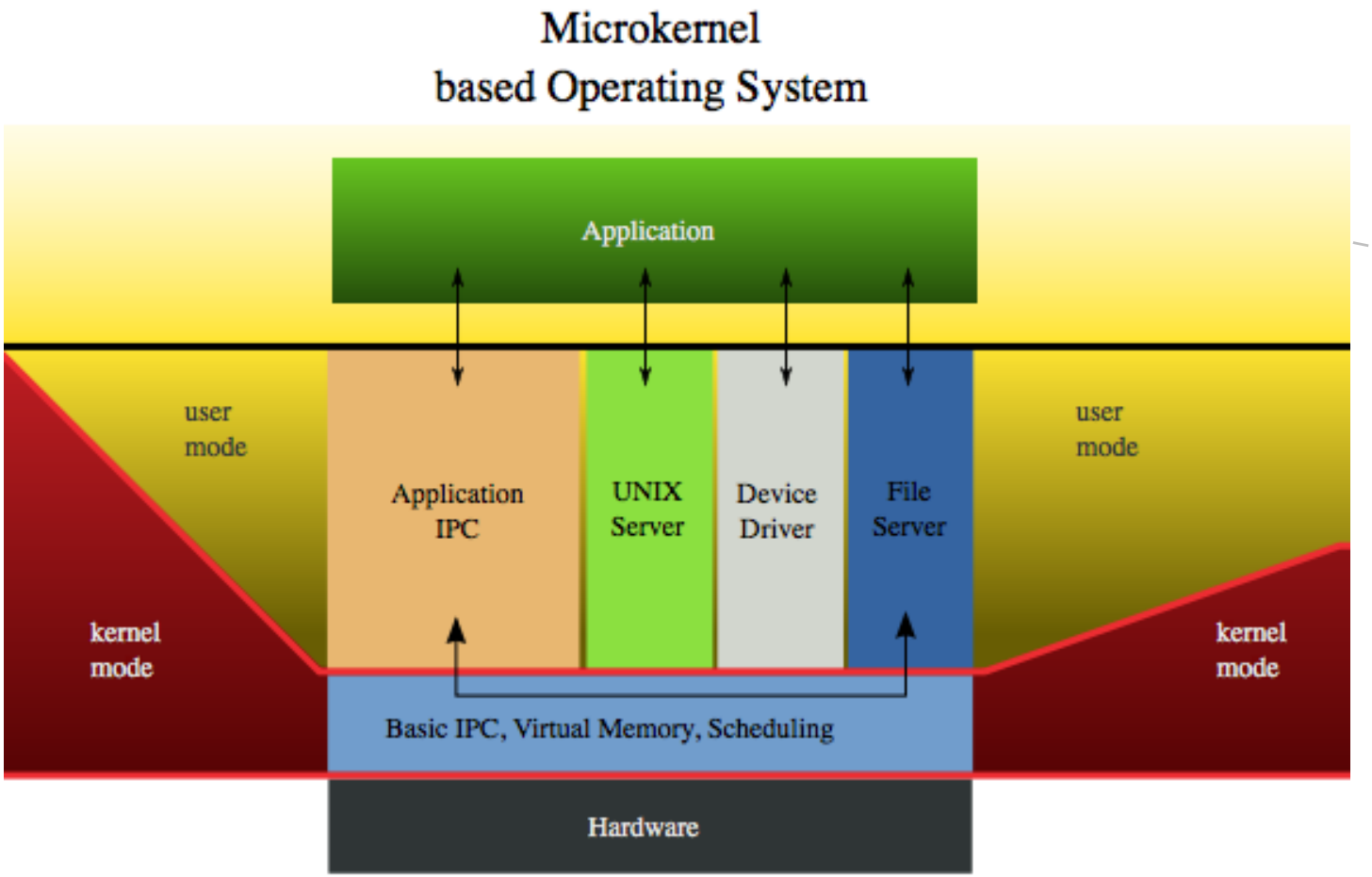
Monolitisk kjerne 3

- Ulemper:
  - Avhengigheten mellom de forskjellige systemkomponentene: Feil i en driver kan krasje hele systemet
  - Store kjerner er vanskelige å vedlikeholde
  - Ikke portable: Må skrives på nytt for hver ny arkitektur
- Hvor brukes monolittiske kjerner?
  - Unix-lignende operativsystemer
  - Nyere monolittiske systemer, som f.eks Linux, kan laste kjernemoduler etter behov.

Monolitisk kjerne 4

- Monolittisk Linux-kjerne kan lages ekstremt liten fordi
  - Moduler kan lastes dynamisk
  - Enkel å konfigurerer etter spesifikk behov
- De minste Linux-ene kan kjøres på maskin med 4M RAM og 20M harddisk
- Linux egner seg derfor godt i såkalte ”embedded systems”, dvs innebygd i små enheter (web-kameraer, fjernkontroller, husholdningsapparater etc)

Mikrokjerne 1



Mikrokjerne 2

... an approach to Operating System design by which the functionality of the system is moved out of the traditional "kernel", into a set of “servers” that communicate through a “minimal” kernel, leaving as little as possible in “system space” and as much as possible in “user space”

Kilde: `https://en.wikipedia.org/wiki/Kernel_(operating_system)`

Funksjonaliteten til systemet er flyttet ut av den tradisjonelle kjernen og inn i et antall tjenere (servere) for så å kommunisere gjennom en minimal kjerne med minst mulig programkode i kjernemodus og mest mulig i brukermodus.

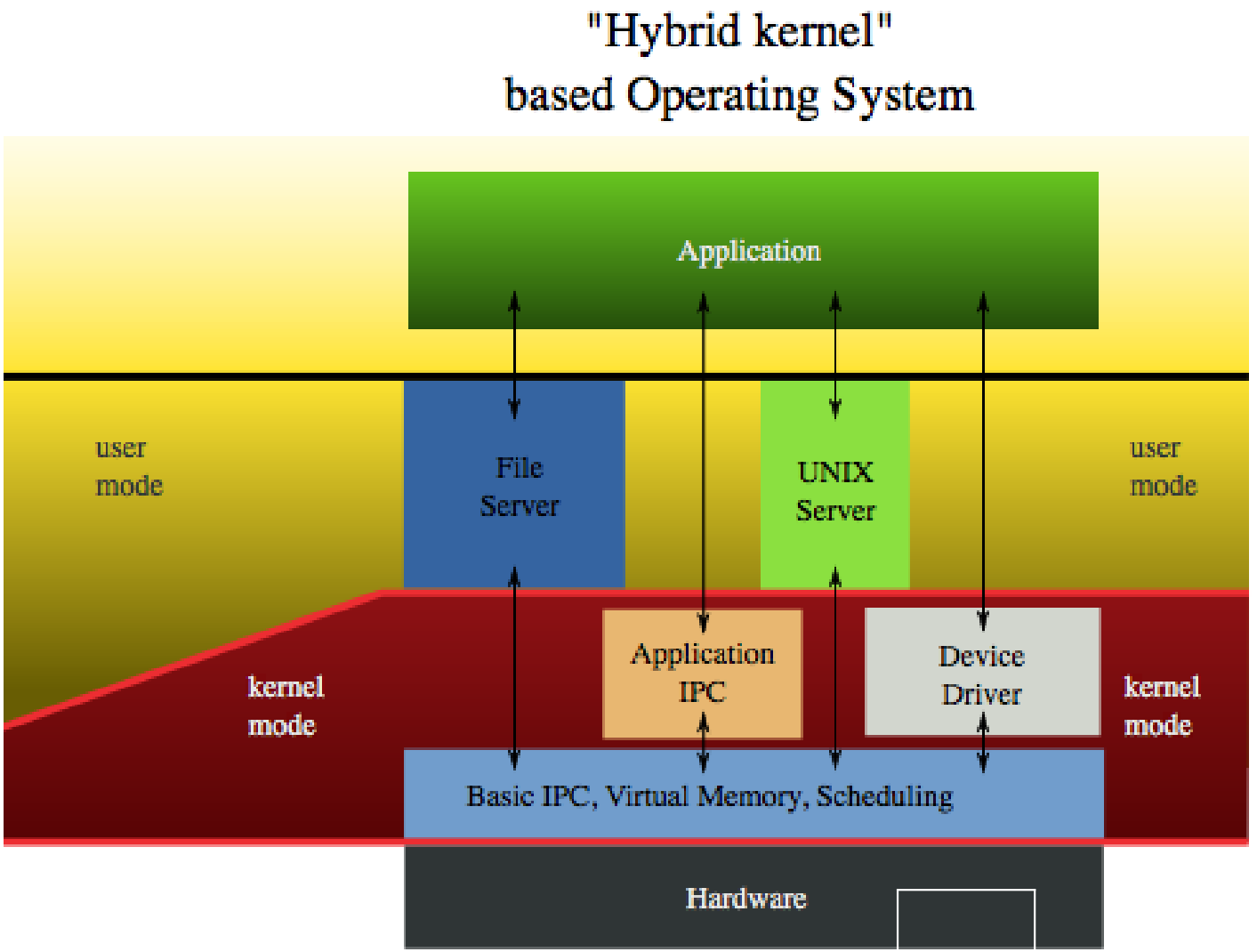
Mikrokjerne 3

- Kjernen (av type mikro) et sett av funksjoner og systemkall for å realisere et minimalt OS med tjenester som:
  - Grunnleggende minneadministrasjon
  - Grunnleggende multitasking (prosessadministrasjon)
  - Grunnleggende kommunikasjon mellom prosesser
  - Grunnleggende I/O
- Andre OS-tjenester som normalt ligger i kjernen er nå flyttet ut i brukermodus og kalles tjenere:
  - Komplett scheduler
  - Komplett minneadministrasjon
  - Filsystem
  - Nettverk

Mikrokjerne 4

— Mikrokjerne sammenlignet med monolittisk:
  - Mikrokjerne er enklere å vedlikeholde pga modularisert oppbygging
  - Systemkallene og kontekstskiftene går saktere enn hos monolittisk sidene kommunikasjonen nå går mellom tjenere og en mikrokjerne
  - Mikrokjerne bruker mer minne enn monolittisk

Hybridkjerne 1



Hybridkjerne 2

- Hybridkjerne er en kombinasjon av monolittisk og mikrokjerne design.
- Noen tjenester (slik som nettverksstakk og filsystem) er lagt tilbake i kjernemodus for å øke ytelsen sammenlignet med tradisjonell mikrokjerne.
- Men fortsatt kjøres en god del kjerneprogramkode som egne tjenere i brukermodus, f.eks utstyrsdrivere.

Hybridkjernen kombinerer ytelse og enkel design fra monolittisk kjerne med modularitet og kjørestabilitet fra mikrokjerne (f.eks krasj hos en driver tar ikke ned hele systemet)

Hvem bruker hva?

- Monolittisk
  - Unix, Linux, FreeBSD
- Mikrokjerne
  - AIX, BeOS, Hurd, Mach, Minix, QNX
- Hybridkjerne brukes i de kommersielle OS-ene:
  - Microsoft NT, XP, Vista, 7, 8, 10
  - Apple Mac OS X basert på Carnegie Mellons Mach-kjerne og FreeBSDs monolittiske kjerne