4. Naming

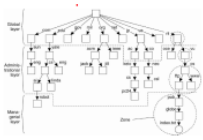
Introduktion: Maskiner har access points. Tilgåes via addresse. Findesvha names, identifiers.

Flat naming: Identifier → Addresse.

- **Forwarding pointers:** tidligere addresse peger på nye addresse. Kæde af pointers → lang tid + stor risiko for at kæden brydes. Multiple points of failure + lang svartid + knuder skal opretholde links.
- **Home-based:** Hjemme-lokation holder addresen. Addressen på enheden opdateres løbende. Single point of failure + lang svartid på første besked.
- **DHT:** Et chord-netværk. Identifiers udgør keys. **Finger-tabel** for hurtigere LOOKUP. **[TRANSPARENT]**
  $FT_p[i] = succ(\boldsymbol{p} + \mathbf{2^{i-1}})$
  Request på k giver:
  $q = FT_p[j] \leq k < FT_p[j+1]$
  
  - **Joining/leaving:** Joiner ved at finde succ(p+1) og leaver ved at sige det til naboer.
  - **Opdater FT:** Vigtigst er FT[1]. Tjek jævnligt næste knude, for at se om q = pred(succ(q+1)). Tjekker jævnligt hele FT samt pred.

Structured naming: Navn → Adddresse. **[TRANSPARENT]**

- Navne organiseres i **name space** → labeled directed graf. Mindst én rod + leaf i træet har et **path name** der er defineret ud fra roden.



Et name space for store distribuerede systemer (fx internettet) organiseres ofte hirakisk i logiske lag.

1. **Globalt:** Rod-serveren mf. Top level domæner på internettet. Må ikke gå ned. Oppe >>langsom. God udnyttelse af cache, da child servere sjældent ændrer sig.
2. **Administrativt:** Directory-servere der tilhører organisationer. Der kan være flere lag servere pr organisation. Disse servere skal gerne være hurtige til at svære og udføre opgaver.
3. **Managerial:** De resurser der tæt vil blive ændret. Routere og lign på lokale netværk, filer, libraries etc. Da arbejdsopgaverne for servere her er begrænsede, behøver man ofte ikke mere end én.

Name resolver i en klient står for at skaffe resursen. Denne kontaker rod-serveren, med fx en url.

- **Iterativt** Rod-serveren kontaktes med url→ Returnerer addressen på den næste server samt den uresolvede den af url'en. Næste server kontaktes på sammme måde. Fortsætter indtil resursen returneres. **[TRANSPARENT] – se tidligere.**
- **Rekursivt**: Klienten kontaker kun rod serveren. Denne står for at resolve. Resursen returneres via rod-serveren. Bruges ikke, pga den ekstra arbejdsbyrde på rod-serveren. Udnytter dog caching på server siden meget. **[TRANSPARENT] - se tidligere.**