Unstructured P2P networks

- Hvad er et ustruktureret netværk?
    - Forskelle fra struktureret netværk
- **Topologi**
    - Power law distribution - set i Gnutella og GIA
        - Altid tæt på en well connected peer
- **Søgning**
    - Flooding - giver meget traffik
    - Expanding ring - bedre, men stadig skidt
    - Random walkers - skalerer rigtig godt, ikke sikkert man finder svar
        - Høj TTL - mange hops kan give store delays
        - K-walkers - større sandsynlighed for at finde svar, men mere trafik
        - Check back
        - Send ikke forespørgsel til samme peer to gange
        - Biased walking - Send forespørgsel til well connected peers
- **Adaptive probabilistic search**
    - Husk hvor hits kommer fra så fremtidige søgninger kan sendes samme vej
    - Større sandsynlighed for at sende i en retning hvor man har fået mange hits
    - Walkers skal backtrackes for at levere denne information til peers
    - Optimistisk eller pessimistisk approach
        - Optimistisk - alle walkers der ikke får svar backtrackes
        - Pessimistisk - kun den walker der får svar backtrackes
    - Switching-APS - optimistisk ved populære søgninger pessimistisk ellers
    - Weighted-APS - sandsynligheder er omvendt proportionale med distancen til ressourcen
    - Virker bedst med lav churn rate
- **Churn**
    - Ustrukturerede netværk er som regel gode til at håndtere churn
- **GIA**
    - One-hop replication - hold et index over naboer
    - Active flow control - token baseret system (giver incitament til at dele)

Structured P2P networks

- Struktureret vs ustruktureret netværk
    - Skalerer langt bedre
    - Routing er deterministisk og søgning kan gøres i O(log n) tid
    - Større problemer med håndtering af churn grundet mere state
    - Mere sårbar overfor targeted angreb da routing er deterministisk
    - Keyword search er svært da der arbejdes med hashfunktioner - Cubit
- **Distributed hash tables**
    - Hver peer har et tilfældigt ID
    - Ressourcer får også ID og bliver gemt hos den peer med det nærmeste ID
    - Udfordringer
        - Routing skal distribueres
        - Ved join og leave skal ressourcer flyttes og state opdateres
- **Chord** - prototypisk eksempel på DHT netværk
    - En operation → lookup(ID)
    - Tegn ring med peers
        - Forklar resource assignment
        - Forklar join, leave og failure
    - Forklar lookup med finger tables - O(log n) time & space
    - Håndtering af failure
        - Nodes holder en liste af successors
        - Resource replication
        - Lille risiko for at nodes med tætte ID dør samtidigt da ID tildeles tilfældigt
    - For simpelt → ignorerer lokalitet og styrke af peers
- **Pastry**
    - Forbedret ved mere komplex routing table som tager højde for lokalitet
    - Routing table består af
        - Leaf set → indeholder L tætteste peers i ID space
        - Routing table → indeholder udvidet finger table
        - Neighbourhood set → indeholder M tætteste peers i IP space
    - **Routing**
        - Tegn routing table (b=2) og forklar levels og b parameteren
        - Check om ID er i leaf set → one hop
        - Find en indgang i routing tabellen med længere shared prefix og forward beskeden → O(log n) hop
        - Hvis en sådan ikke findes, send til en peer med ligeså langt prefix, men tættere på numerisk → sjældent tilfælde (døde nodes)
    - Joining
        - Find peer A vha. multicasting og benyt dennes neighbourhood set
        - A sender join besked på X's ID som ender hos Z
        - X bruger Z's leaf set da de ligger tæt på hinanden i ID space
        - Alle nodes på join-beskedens vej sender info til X som bruges til routing table
    - **Location awareness**
        - Sikres i opbygning af routing tabel
- Cubit - keyword search (brugere kan ikke stave)
    - ID space er keywords og afstand måles i Levenshtein distance

Applications for P2P

- **YouServ** → Deling af dokumenter i større virksomhed (IBM)
    - Deling er svært
        - Email, Central server, Standard P2P sharing, Shared file systems
    - Webserver til deling af data, webbrowser til at tilgå data
    - Centralt dynDNS til at connecte til peers (kræver centraliseret authentifikation)
    - Coordinator → Tjekke authentifikation, Tjekke tilgængelighed af peers, Registre proxy og replicas
    - **Replication**
        - Tegn kald strukturen
        - Manuelt valg af hvem der skal replikere data og hvad der skal replikeres
        - dynDNS redirecter til replicas når den primære peer ikke er online
    - **Proxying**
        - Tegn kald strukturen
        - Opret permanent socket til en proxy peer som ikke er bag firewall
        - Alt ingående trafik routes gennem proxy og socket
- **YouSearch** → Effektiv søgning i YouServ
    - Nodes indekserer deres egne dokumenter og laver et bloom filter
    - Bloom filter sendes til centralt register
    - Registeret beregner et samlet bloomfilter for effektiv forwarding af queries
    - YouSearch benytter *k* hash funktioner og registeret holder *k* bit vectors
        - Dette muliggør *k* parallele registre
    - Søgning
        - En query sendes til registeret som returnerer en liste af peers der måske har filen
        - Den søgende peer sender queries til disse peers
        - Hvis filen findes opdateres registerets cache med query og IP på søgende peer. Den søgende peer cacher resultatet. Dette øger performance markant.
- **PAST** - distribueret backup system
    - **Replica diversion**
        - Hvis en peer ikke har plads til en fil lokalt spørger den en peer i sit leaf set og gemmer en pointer til filen.
    - **File diversion**
        - Hvis der ikke er plads til filen i nærheden af den peer der skal gemme filen, genereres et nyt fileID med en ny random salt. Dette sker sjældent.
    - Caching
        - Peers besøgt på vejen, i en lookup eller insert operation, cacher den efterspurgte fil hvis der er plads
- **SCRIBE** - P2P based multicast system
    - Gruppe chat, medie streaming, multiplayer spil
    - Message passing
        - Top-down → Peer A beder Root om at forwarde en besked. Beskeden bliver derefter sendt ned gennem træet. Der er højt load på Root. Gør det lettere at kontrollere sekvensen af beskeder.
        - Crawl → Alle peers er lige. Peer A forwarder selv sin beskedtil sine naboer. Gør det svært at kontrollere sekvensen af beskeder.
    - Sikkerhed
        - Peers kan floode netwærket, eller nægte at forwarde beskeder

Security and reputation in P2P

- Angreb
  - Angreb på P2P netværk er anderledes end angreb på centraliserede systemer
  - DDoS angreb
  - Ondsindede peers
- **Sybil / eclipse**
  - Opret en masse falske peers → har større effekt end en ondsindet peer
  - Defence: lave det dyrt at joine eller check subnets
  - Eclipse attack
    - Omring en eller flere peers og kontroller alt traffik fra disse
    - Defence: lad ikke peers vælge hvor de vil indsættes i netværket
- **Anonymitet og censur**
  - Anonymitet → mix networks, afsender og modtager kan ikke begge kendes
  - Enhver mixer har et nøglepar (asymetrisk kryptering)
  - Beskeder sendt på netværket bliver krypteret et antal gange og routet gennem netværket
    - Hver peer kan kun dekryptere et lag og sende beskeden videre
- **Tarzan**
  - P2P mix netværk med cover traffic (mimics) der giver fuld anonymitet
  - **Edge analysis** → umuligt da der ikke er nogen edges, alle peers er ens
  - **Traffic analysis** → umuligt pga. mimics. Det er umuligt at vide hvornår en peer sender støj og hvornår de sender information.
  - Netværket opdateres vha. gossipping
  - Taler kun til sine naboer
  - **Sybil angreb** → En message path er altid konstrueret så den krydser subnets
  - Jo flere peers der deltager jo mere mere anonymt er det
- **Freenet**
  - Distributed information storage system
  - Struktureret P2P netværk, meget skalerbart
  - Hver peer tilbyder storage, men ved ikke hvad der bliver gemt (er krypteret)
    - **Plausible deniability** → man står ikke til ansvar for opbevaret data
  - Filer caches langs routing path
    - **DDoS angreb** → efterspurgte ressourcer bliver replikeret over hele netværket så det er svært at lægge ned.
  - **Traffic analysis** → svært da de benytter en slags mixing før normal routing
  - **Sybil attacks** → insertion ID bliver genereret af peers i fællesskab så man kan ikke indsætte sig selv taktisk smart
- **ARA** - sikrer god opførsel af peers
  - Hvordan kan vi få peers til at opføre sig ordentligt?
  - Benytter credit system → svarer til share ratio
  - Hver peer har et sæt af *interested peers* der holder øje med pågældene peer
    - Interested peer → har kommunikeret med pågældende peer indenfor et tidsrum
  - Da mængden af credit i systemet er 0, kan man kun snyde sig bedre på andres bekostning
  - Ved hver transaktion genereres en rapport
  - Rapporten benyttes til at kontrollere på en peer (undersøge om den har snydt)
  - Dette gøres ved at sammenholde peerens rapporter med rapporter fra interested peers
  - Hvis peeren har snydt vil de interested peers sige det, da de ellers taber credit (selfish)
  - Kræver meget bogholderi og traffik, men løser opgaven uden central server

Mobile ad hoc networks

- Egenskaber ved MANETs
  - Batteri
  - Ingen central
  - Kommunikerer trådløst
- Routing på Internettet
  - **Link State**
    - Kender hele netværket og kan derfor levere optimale ruter
    - Kræver meget traffik at lave routing tables
    - Virker bedst på små og stabile netværk
  - **Distance Vector**
    - Kommunikerer kun med naboer
    - Routing tabeller bliver gradvist opbygget
    - Mindre overhead, og skalerer derfor bedre
    - God til at håndtere ustabile netværk
- Routing i MANETs
  - Skal tage højde for MANETs egenskaber
  - **Proaktive protokoller**
    - Routing information samles til at starte med og holdes up-to-date
    - Stort overhead hvis der ikke er meget traffik
  - **Reaktive protokoller**
    - En rute findes når den skal bruges
    - Benyt flooding for at finde en rute til destinationen
    - Giver høj latency
    - Effektiv når der ikke er meget traffik
  - Udnyttelse af wireless kommunikation → Passive acknowledgements og route shortening
- **Energi effektiv MANET routing**
  - Udnytter non-lineær relation mellem transmission distance og energien brugt herpå
  - Nedsæt signalstyrke
    - Nedsætter støj på netværket (sparer også energi hos modtagere)
  - Meget strøm bliver brugt i IDLE state
  - Benyt “sleep” intervaller og retransmit tabte pakker
    - Øger latency
  - Span → Hav en “vågen” routing backbone af coordinators
    - Peers skiftes til at være coordinators for at fordele strømforbrug

BitTorrent

- Terminologi
    - Tracker → central komponent til peer discovery
    - Seeds → peers der har den fulde fil
    - Leechers → peers der aktivt downloader filen
    - Swarm → foreningen af seeds og leechers
    - .torrent file → meta data fil som indeholder information omkring torrenten
- Trackeren vælger peers helt tilfældigt (de er ikke vægtet)
- **Protocol**
    - Efter åbning af .torrent filen kontaktes trackeren
    - Trackeren returnerer en liste af ca. 50 tilfældige peers (fra swarm)
        - Trackeren kontaktes hver halve time for at vise aktivitet
        - Trackeren kontaktes hvis en peer skal have nye peers fra swarm
        - Trackeren kontaktes når en peer forlader swarm
    - Der oprettes TCP forbindelser til nogle af de returnerde peers (laver neighbourhood)
    - Data omkring pieces bliver udvekslet vha. bitmaps (opdateres når et piece downloades)
    - Download starter, men da peeren ikke har noget at dele kan den heller ikke downloade
    - Problemet løses vha. **choking**
        - Hver peer i nabolisten har to bits ((un)interested og (un)choked)
        - Interested → om pågældende peer er interesseret i de pieces vi har
        - Choked → vi tillader ikke pågældende peer at downloade fra os
        - Når vi downloader fra en peer unchokes den (den kan også downloade fra os)
            - Derfor er det bedst at vælge interesserede peers at downloade fra
        - Optimistic unchoke → Sørger for at peers uden pieces kan komme i gang
            - Kan exploites (kommer tilbage til)
        - Alle peers skal til enhver tid have et antal unchoked naboer
    - Seeders uploader til de peers hvor den kan dele mest muligt
    - Valg af peice til download
        - Random → peices bliver distribueret tilfældigt så alle har noget at dele
        - Rarest first → Download de sjældneste peices først, mindre risiko for at torrenten går i stykker da ingen peer vil have "den eneste kopi" særlig længe
- **Angreb**
    - Ødelægge the swarm → Gøre det svært at downloade en fil
        - **Piece lying** (sybil) - Udnytter "rarest first" ved at sige de har pågældende pieces som dermed ikke er sjældne mere. Hvis en peer requester dem checkes denne.
        - **Eclipse attack** - Peers vil forsøge at connecte til korrekte peers så de omringes. Dette kan kraftigt øge download tiden.
    - Udnytte the swarm → downloade uden at dele
- **BitTyrant**
    - Split en high capacity peer op i low capacity peers (får mere ud af optimistic unchokes)
    - De viser også at andre peers kan snydes ved kun at uploade lidt, men downloade mere. Active set vælges så til de peers der har størst ratio og udvides indtil peeren ikke har mere upload capacity.
    - Kan øge effektiviteten med 70%
    - Et system udelukkende med BitTyrant klienter vil være mere fair
    - Tit-for-tat er ikke det der giver BitTorrent dens effektivitet, men i stedet at peers er villige til at dele uden at få lige så meget igen. Det er dette BitTyrant udnytter.