Peer-to-peer systemer

INF 5040 høst 2005

foreleser: Frank Eliassen

Motivasjon for peer-to-peer

- Iboende begrensninger til standard klient/tjener model
  - Sentralisert design, mangel på skalerbarhet og feil-toleranse
    - Prosessering
    - Nettverks-båndbredde
- P2P systemer handler om å distribuere prosesseringslast og nettverks-båndbredde mellom alle noder som deltar i et distribuert informasjonsystem
  - Løser flaskehalsproblemet, men må ”betales for” i form av betydelig mer komplekse algoritmer og ytterlige sikkerhetsproblemer

Hva er P2P?

- I et P2P system opptrer enhver deltagende node som både klient og tjener ("servent"), og "betaler" for deltagelsen ved å tilby aksess til noen av sine ressurser (typisk prosesserings, og lagringsressurser, men kan og være logiske ressurser (tjenester)).
- Et applikasjonsnivå-Internett på toppen av Internett (overlay network)

Frank Eliassen, SRL & Ifi/UiO 3

Fokus for forelesningen

- Presentere generelle teknikker som forenkler konstruksjon av peer-to-peer applikasjoner og øker deres grad av skalerbarhet, pålitelighet, og sikkkerhet.

Frank Eliassen, SRL & Ifi/UiO 4

Underliggende karakteriska for P2P systemer

- Hver deltager bidrar med ressurser til systemet
- Alle noder har samme funksjonelle kapabilitet og ansvar
- Ingen avhengighet av en sentral entitet for administrasjon av systemet (selv-organiserende)
- Effektiviteten avhenger kritisk av algoritmer for plassering av data over mange noder og for seinere aksess til disse
- Uforutsigbar tilgjengelighet til prosesser og noder

Utviklingen av P2P systemer og applikasjoner

- Første generasjon
  - Napster
    - Deling/utveksling av musikk
    - Hybrid C/S og P2P (sentral indeks-server)
- Andre generasjon
  - Gnutella, Freenet, Kazaa, ...
    - Desentraliserte fildelingssystem
- Tredje generasjon
  - P2P mellomvare
    - Applikasjonsuavhengig mellomvarelag for forvaltning av distribuerte ressurser i global skala
    - Pastry, Tapestry, CAN, Chord, ...

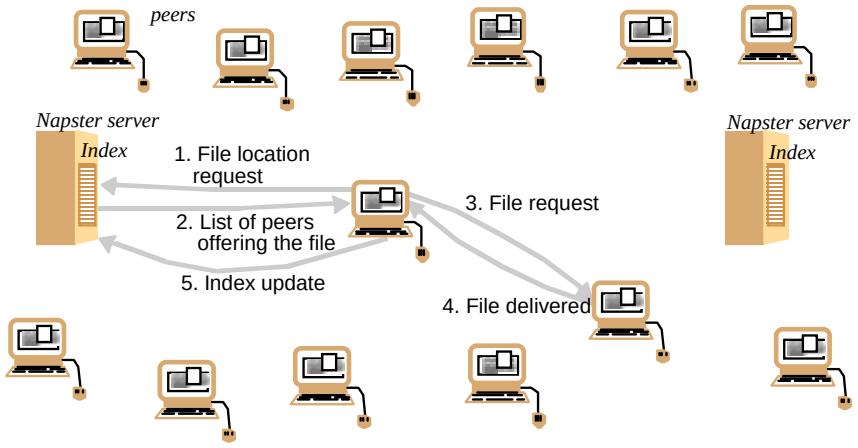
P2P mellomvare karakteristika

- Konstruert for å
  - utplassere ressurser (dataobjekter og filer) på en mengde datamaskiner som er vidt spredt utover Internet
  - rute meldinger til dem på vegne av klienter
  - skjule for klientene lokasjonen til ressursene (tranparens)
- Gir effektivitetsgarantier (antall hopp)
- Replikerer ressurser til noder på en strukturert måte for å tilfredsstille krav om tilgjengelighet, tillit, lastbalansering og lokalitet.
- Ressurser identifiseres ved GUIDs (avledet fra "sikker enveis-hash" – se læreboka kap 7.4.3).
  - Randomisert fordeling av objektreplika til noder i forskjellige organisasjoner i hele verden reduserer sannsynligheten for samtidig utilgjengelighet

Forskjell mellom IP og overlay routing for P2P applikasjoner (fra boka)

	IP	***Application-level routing overlay***
Scale	IPv4 is limited to 2^{32} addressable nodes. The IPv6 name space is much more generous (2^{128}), but addresses in both versions are hierarchically structured and much of the space is pre-allocated according to administrative requirements.	Peer-to-peer systems can address more objects. The GUID name space is very large and flat (>2^{128}), allowing it to be much more fully occupied.
Load balancing	Loads on routers are determined by network topology and associated traffic patterns.	Object locations can be randomized and hence traffic patterns are divorced from the network topology.
Network dynamics (addition/deletion of objects/nodes)	IP routing tables are updated asynchronously on a best-efforts basis with time constants on the order of 1 hour.	Routing tables can be updated synchronously or asynchronously with fractions of a second delays.
Fault tolerance	Redundancy is designed into the IP network by its managers, ensuring tolerance of a single router or network connectivity failure. *n*-fold replication is costly.	Routes and object references can be replicated *n*-fold, ensuring tolerance of *n* failures of nodes or connections.
Target identification	Each IP address maps to exactly one target node.	Messages can be routed to the nearest replica of a target object.
Security and anonymity	Addressing is only secure when all nodes are trusted. Anonymity for the owners of addresses is not achievable.	Security can be achieved even in environments with limited trust. A limited degree of anonymity can be provided.

Napster



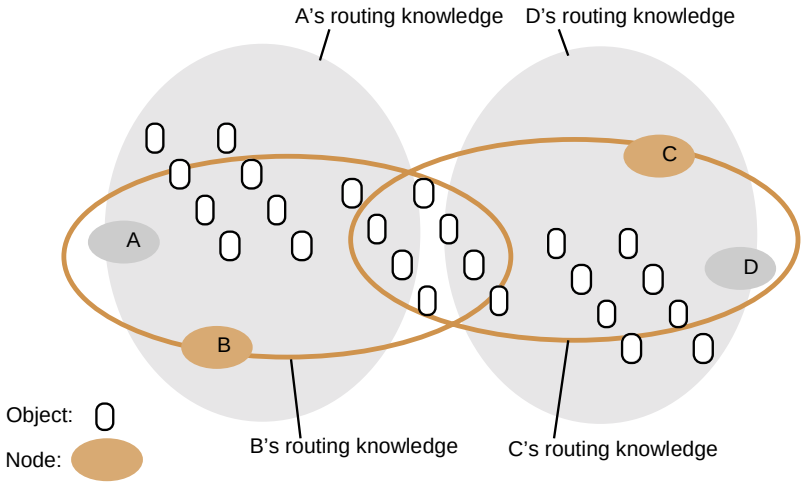
P2P mellomvare (1 av 2)

- Utfordring: tilby en mekanisme som gir rask og pålitelig aksess til data ressurser på en lokasjonstransparent måte.
- Funksjonelle krav
  - Forenkle konstruksjon av tjenester som implementeres over mange noder i et utstrakt distribuert nettverk
    - Muliggjøre lokalisering og kommunikasjon med enhver ressurs som er gjort tilgjengelig
    - Mulig å legge til nye ressurser og fjerne gamle
    - Mulig å legge til nye noder og fjerne gamle
    - Enkel applikasjonsuavhengig API

P2P mellomvare (2 av 2)

- Ikke-funksjonelle krav
  - Global skalerbarhet
  - Lastbalansering
  - Optimalisering for lokal interaksjon mellom nabo-”peers”
  - Kunne handtere svært dynamisk node-tilgjengelighet
  - Sikkerhet til data i en omgivelse med ulike tillitsrelasjoner
  - Anonymitet, “deniablity”, og resistens mot sensur

Distribusjon av informasjon i et “routing overlay”



Routing overlay

- Applikasjonslagsalgoritme for å lokalisere noder og replikerte objekter (uavhengig av nettverksruting)
- Gjerne implementert som et mellomvarelag
- Sikrer at enhver node kan aksessere etthvert objekt ved å rute hver forespørsel gjennom en sekvens av noder, og utnytte kunnskapen til hver av dem for å lokalisere objektet (eller nærmeste replika)
- Ansvarlig for livsykelforvaltning av objekter og noder

Grunnleggende API for en Distribuert Hash-Tabell (Pastry)

- Et objekts GUID beregnes fra hele eller deler av dets tilstand vha en enveis hash-funksjon (som SHA-1).
- GUIDene benyttes til å plassere objekter og til å gjenfinne dem (derav: distribuert hash-tabell)

put(GUID, data)
The *data* is stored in replicas at all nodes responsible for the object identified by *GUID*.
remove(GUID)
Deletes all references to *GUID* and the associated data.
value = get(GUID)
The data associated with *GUID* is retrieved from one of the nodes responsible it.

Grunnleggende API for Distribuert ObjektLokalisering og Ruting (DOLR) i Tapestry

publish(GUID)
GUID can be computed from the object (or some part of it, e.g. its name). This function makes the node performing a *publish* operation the host for the object corresponding to *GUID*.
unpublish(GUID)
Makes the object corresponding to *GUID* inaccessible.
sendToObj(msg, GUID, [n])
Following the object-oriented paradigm, an invocation message is sent to an object in order to access it. This might be a request to open a TCP connection for data transfer or to return a message containing all or part of the object's state. The final optional parameter *[n]*, if present, requests the delivery of the same message to n replicas of the object.

Studie av et kasus: Pastry

- Har en enkel men effektiv design.
- Noder og objekter tilordes en 128-bit GUID
  - Benytter sikker hash-funksjon på nodens "public key" og på objektets navn eller del av tilstand
- I et nettverk med N noder, vil Pastry rutingalgoritme levere en melding adressert til en vilkårlig GUID, i O(log(N)) steg
  - Hvis GUIDen identifiserer en aktiv node, leveres meldingen til den. Ellers leveres meldingen til noden med GUID numerisk nærmest til den.
- Fullt selv-organiserende
  - O(log(N)) meldinger når deltager slutter seg til resp. når en deltaker forlater tjenesten

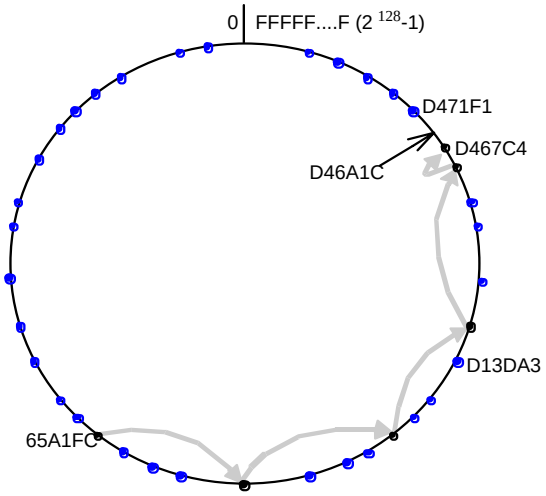
Rutingalgoritme i Pastry

- Beskrives i to stadier i læreboka:
  - En forenklet algoritme uten bruk av rutingtabell (men som bruker "nabo-informasjon") som ruter korrekt men lite effektivt
  - Den komplette algoritmen som ruter en forespørsel til en vilkårlig node ved O(log(N)) meldinger

Rutingalgoritme i Pastry: Stadium 1

- Hver aktive node vedlikeholder en vektor L ("the leaf set") av lengde 2l, som inneholder GUID og IP adresse til de nodene med GUID som er numerisk nærmest på hver side (l over og l under)
- Vektoren vedlikeholdes av Pastry (når noder slutter seg til og forlater tjenesten)

Sirkulær ruting: Korrekt, men ineffektiv



The dots depict live nodes. The space is considered as circular: node 0 is adjacent to node (2^{128}-1). The diagram illustrates the routing of a message from node 65A1FC to D46A1C using leaf set information alone, assuming leaf sets of size 8 (l = 4). This is a degenerate type of routing that would scale very poorly; it is not used in practice.

Rutingalgoritme i Pastry: Stadium 2

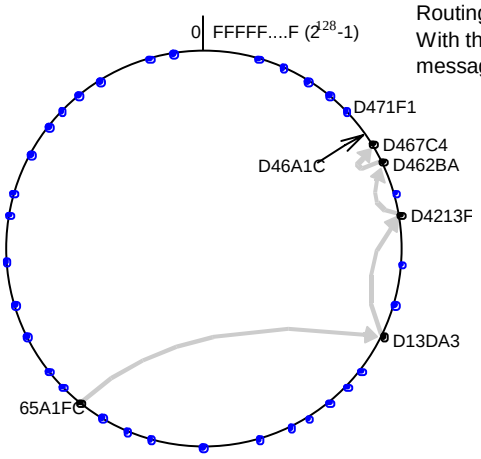
- Effektivisere algoritmen fra stadium 1 vha rutingtabeller
- Enhver Pastry node vedlikeholder en tre-strukturert rutingtabell som gir GUID og IP-adresse for en mengde noder spredt utover hele adresserommet av GUID-verdier.
- Tabellens “dekningstetthet” er størst for GUIDer som er numerisk nærmest til nodens egen GUID.

Første fire rader i en Pastry ruting tabell

p =	GUID prefixes and corresponding nodehandles n														
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	n	n	n	n	n	n		n	n	n	n	n	n	n	n	n
1	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
	n	n	n	n	n		n	n	n	n	n	n	n	n	n	n
2	650	651	652	653	654	655	656	657	658	659	65A	65B	65C	65D	65E	65F
	n	n	n	n	n	n	n	n	n	n		n	n	n	n	n
3	65A0	65A1	65A2	65A3	65A4	65A5	65A6	65A7	65A8	65A9	65AA	65AB	65AC	65AD	65AE	65AF
	n		n	n	n	n	n	n	n	n	n	n	n	n	n	n

The routing table is located at a node whose GUID begins 65A1. Digits are in hexadecimal. Then n represent [GUID, IP address] pairs specifying the next hop to be taken by messages addressed to GUIDs that match each given prefix. Grey-shaded entries indicate that the prefix matches the current GUID up to the given value of p: the next row down or the leaf set should be examined to find a route. Although there are a maximum of 128 rows in the table, only $\log_{16} N$ rows will be populated on average in a network with N active nodes.

Pastry ruting eksempel



Routing a message from node 65A1FC to D46A1C. With the aid of a well-populated routing table the message can be delivered in ~ $\log_{16}(N)$ hops.

Pastry ruting algoritme

To handle a message M addressed to a node D (where $R[p,i]$ is the element at column i, row p of the routing table):

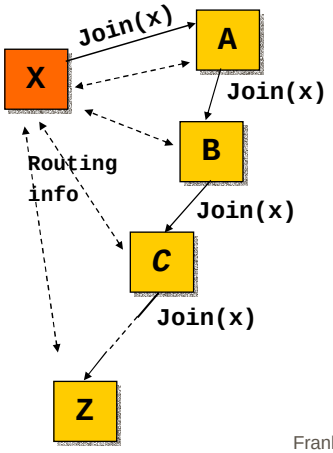
1. If ($L_{-l} < D < L_l$) { // the destination is within the leaf set or is the current node.
2. Forward M to the element L_i of the leaf set with GUID closest to D or the current node A.
3. } else { // use the routing table to despatch M to a node with a closer GUID
4. find p, the length of the longest common prefix of D and A. and i, the $(p+1)^{th}$ hexadecimal digit of D.
5. If ($R[p,i] \neq null$) forward M to $R[p,i]$ // route M to a node with a longer common prefix.
6. else { // there is no entry in the routing table
7. Forward M to any node in L or R with a common prefix of length i, but a GUID that is numerically closer.

}

}

Pastry: Integrasjon av ny host

- Join protokoll for å etablere rutingtabell og “leaf set”



X: new node to join

A: closest neighbour

Z: GUID closest numerically to X (routed to the normal way)

B, C, ...: nodes the join message is routed via

A, Z, B, C, ... transmits relevants parts of their routing tables and leaf sets to X. X uses this info to build its initial routing table and leaf set.

X then transmits its routing table and leaf set to A, Z, B, C, ...

Pastry: Host forlater P2P systemet eller feiler

- En Pastry node anses som å ha forlatt P2P systemet når dens umiddelbare naboer (i GUID rommet) ikke kan kommunisere med den lenger
  - Alle noder sender 'heartbeat' meldinger til nabonoder (i flg sitt leaf set)
- Når dette skjer er det nødvendig å reparere alle leaf set som inneholder GUID til noden som forsvant
  - En node reparerer sitt "leaf set" L ved å spørre en node nær til den som feilet om å sende sitt "leaf set" L'. En node fra L' velges for å erstatte noden i L som feilet.
- Rutingtabeller repareres "ved behov" (når ruting-forespørsler feiler)

Pastry: Feiltoleranse og pålitelighet

- "At least once" mekanisme for meldinger
  - Gjenta rutingforespørsel ved fravær av respons
  - I mellomtiden kan muligens rutingfeilen være reparert
- Randomsiering av ruting valg (steg 5 i rutingalgoritmen)
  - I noen tilfeller velges en node med mindre felles prefiks enn det maksimale (gir ruting valg som avviker fra standardalgoritmen på en tilfeldig måte)
  - Dersom en ondsinnet node blokkerer en rute vil det før eller siden bli valgt en annen ved bruk av "at least once" mekanismen
- Flere utvidelser av Pastry (MSPastry) for å bidra til pålitelighet (dependability)
  - Ack etter hvert hopp i rutingalgoritmen og valg av alternativ rute ved timeout
  - "heartbeat"-meldinger
  - m.m.

Evaluering (MSPastry)

- Basert på simuleringer [Castro et al 2004]
  - Overodnet resultat: god ytelse og høy pålitelighet med tusener av noder
  - Smidig degradering ved økende feilrate i nettverket
- Pålitelighet
  - Med 0% tapsrate på IP-meldinger, MSPastry tapte 1,5 av 100.000 forespørsler
  - Med 5% tapsrate på IP-meldinger, MSPastry tapte 3,3 av 100.000 forespørsler, og 1,5 av 100.000 forespørsler ble levert til gal node
- Ytelse
  - Målt relativ ekstra kostnad å sende meldinger via MSPastry i forhold til mellom to noder via UDP/IP.
  - Relativ ekstra kostand varierte fra ca 1,8 (0% medlingstap av IP-meldinger) til ca 2,2 (5% meldingstap av IP-meldinger)
- Ovehead
  - Kontroll-trafikk utgir ca 2 meldinger per minutt per node i det lange løp (initiell kostnad ved “setup” er imidlertid relativt stor)

Eksempel på Pastry-basert applikasjon: Squirrel

- Web-caching system som utnytter lagrings- og beregningsressurser som allerede er tilgjengelig på desktop-maskiner i et lokalnett
- GUID: anvender SHA-1 på URLen. Gir en 128 bits Pastry-GUID.
- Noden med GUID numerisk nærmest til web-sidens GUID, blir objektets hjemmenode (“home node”)
- Hjemmenoden er ansvarlig for å lagre cache-kopier av objektet (opptrer som proxy-server for dette objektet)
- Klientnoder benytter Squirrel til å rute GET eller cGET forespørsler til web-objektets hjemmenode (dvs web-objektets proxy-server)
- Evaluering viser at ytelsen ved bruk av Squirrel web cache er sammenlignbar med ytelsen ved bruk av sentralisert cache (ytelse målt som (1) reduksjon i bruk av ekstern båndbredde, (2) latenstid erfart av brukeren, (3) lagrings- og beregningsmessig last på klientnoder)

Oppsummering

- P2P systemer distribuerer prosesseringslast og nettverks-båndbredde mellom alle noder som deltar i et distribuert informasjonsystem
- P2P systemer er ikke avhengige av en sentral entitet for administrasjon av systemet (er selv-organiserende)
- Effektiviteten avhenger kritisk av algoritmer for plassering av data over mange noder og for seinere aksess til disse
- P2P mellomvare er et applikasjonsuavhengig programvarelag som realiserer et ”routing overlay”
- Studie og evaluering av et kasus: Pastry
- En Pastry-basert applikasjon: Squirrel web-cache