Retele de calculatoare
Paradigma *peer-to-peer*

Sabin-Corneliu Buraga

http://www.infoiasi.ro/~busaco

"Reality is merely an illusion,
albeit a very persistent one."

Albert Einstein

Cuprins

- Paradigma *peer-to-peer* (P2P)
  - Preliminarii
  - Definitii
  - Caracterizare
  - Tipuri de aplicatii
  - Infrastructuri
  - Aspecte tehnice & aplicatii

Preliminarii

- Uzual, privim clientul ca fiind o componenta:
  - lipsita de capacitati computationale (*dumb terminal*): modelul *master/slave*
  - avind capacitati reduse (PC, dispozitiv fara fir,...): modelul client/server
- Probleme ale arhitecturii client/server:
  - Lipsa robustetei
  - Lipsa rezilientei
  - Lipsa scalabilitatii
  - Incapacitatea oferirii de servicii cind cererea e mare
  - Vulnerabilitate la atac

Definitii

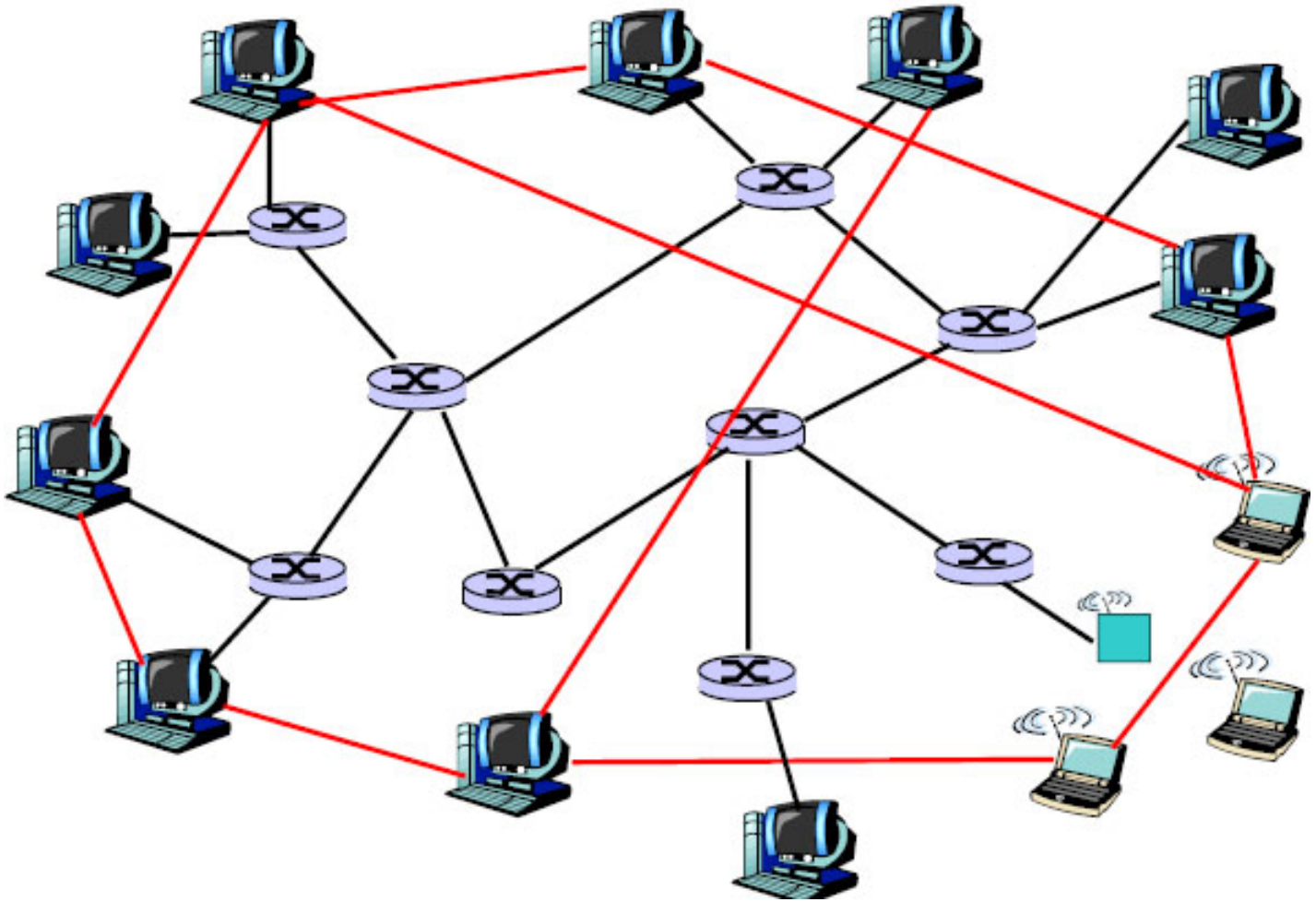
- *Peer = one that is of equal standing with another* (conform Webster)
- *Peer-to-peer* (P2P) ≡ arhitectura de retea in care nodurile sunt relativ egale
  - In sensul ca fiecare nod este in principiu capabil sa realizeze functii specifice ale retelei
  - In practica, multe dintre nodurile retelei pot realiza asemenea functii
- P2P vizeaza partajarea resurselor (servicii de procesare, obiecte digitale,...), folosindu-se de tehnologiile Internet actuale

Definitii

- Sistemele P2P, in sens strict, sunt sisteme complet distribuite
  - Toate nodurile sunt total echivalente, in termeni de functionalitate si a activitatilor pe care le pot desfasura
- *Peer-to-peer* (P2P) ≡ clasa de aplicatii care se bazeaza pe resursele – de stocare, de procesare, continut, prezente umane – disponibile la marginile (*edges*) Internet-ului

Definitii

Edges of the Internet (*overlay networks*)



Caracterizare

- Precursori:
  - Serviciile ARPANET (’69)
    - o serie de noduri sunt tratate ca fiind egale (*peers*)
  - Agentii de transfer al mesajelor de *e-mail*
  - USENET (acum, Netnews) – retea de acces la stiri (’79), independenta initial de Internet
  - DNS (’84) – se bazeaza pe o serie de elemente centralizate, dar ofera o viziune descentralizata
    - DDNS (*Distributed DNS*)

Caracterizare

- Caracteristici definitorii:
  - Partajarea resurselor computationale prin interschimb direct si mai putin prin intermedieri oferite de o autoritate centralizata (server)
    - Serverele centralizate pot fi folosite insa pentru a realiza activitati specifice (initializarea retelei P2P, adaugarea de noi noduri in retea,...)
    - Ideal, nodurile participa activ si unilateral la realizarea de operatii ca localizarea & *caching*-ul nodurilor/continutului, dirijarea informatiilor, managementul resurselor transferate etc.

Caracterizare

- Caracteristici definitorii:
  - Abilitatea de a trata instabilitatea si variatiile conectivitatii retelei, adaptindu-se automat la erorile survenite sau la dinamicitatea nodurilor
    - Topologia retelei P2P e adaptiva si toleranta la defecte, nodurile auto-organizindu-se in vederea mentinerii conectivitatii si performantei retelei

Caracterizare

- Reteaua P2P este una suprapusa (*overlay*) peste cea fizica
  - Se situeaza la nivel de aplicatie ⇒ flexibilitate
  - Muchiile virtuale sunt conexiuni TCP sau pointeri la adrese IP
  - Mentinerea retelei P2P se face prin verificarea periodica a conectivitatii (*ping*) ori a existentei (mesaje “mai traiesti?”)
  - Cind un nod pica, sistemul P2P ar putea stabili noi muchii
  - Proximitatea (fizica) a nodurilor nu e importanta
  - Reteaua P2P poate fi structurata sau nu

Tipuri de aplicatii

- Comunicare & colaborare
  - Sisteme ce ofera o infrastructura pentru facilitarea comunicarii & colaborarii directe, in timp real deseori, intre noduri
    Sisteme conversationale (*chat*, mesagerie instantanee): IRC, ICQ, YM!, Jabber, Skype

Tipuri de aplicatii

- Calcul distribuit
  - Sisteme ce folosesc puterea computationala a nodurilor disponibile (cicli de procesor)
    Rezolvarea unor probleme prin *divide-et-impera*:
    SETI@home, genome@home
    Reteaua P2P poate oferi servicii specifice, fara a se cunoaste exact masinile care le deservesc

Tipuri de aplicatii

- Suport pentru serviciile Internet
  - Sisteme multicast P2P
  - Infrastructuri de indirectare
  - Aplicatii de securitate
    (impotriva atacurilor DoS sau a virusilor)
  - etc.

Tipuri de aplicatii

- Sisteme de stocare (eficienta)
  - Proiectarea de sisteme de baze de date distribuite bazate pe infrastructuri P2P
    - Modelul *Local Relational Model* (LRM)
    - Exemple: PIER – motor scalabil de interogare distribuita, Edutella – proiect *open source* pentru interogari si stocare de meta-date (date privitoare la date)

Tipuri de aplicatii

- Distribuirea de continut digital
  - Sisteme & infrastructuri pentru partajarea resurselor digitale (multimedia si alte date) intre utilizatori
    - Aplicatii pentru partajarea fisierelor (*e.g.*, Napster, Gnutella, KaZaA, Freenet, BitTorrent, eDonkey etc.)
    - Medii de stocare distribuita pentru publicarea, organizarea, indexarea, cautarea si regasirea datelor in maniera securizata & eficienta (PAST, Chord, Groove, Mnemosyne, Avalanche,…)

Tipuri de aplicatii

- Cerinte practice pentru existenta unei arhitecturi P2P efective
  - O retea fizica usor de accesat si scalabila (Internet, intranet, LAN etc.)
  - Protocoale de comunicare intre noduri
  - Conventii de numire a resurselor (noduri, date, servicii, utilizatori)
  - Atasarea de meta-date (date despre date): informatii descriptive despre resurse
  - Un mecanism de cautare
  - Software jucind rolurile de client & server
  - Achizitionarea dispozitivelor computationale formind reteaua P2P + alte resurse (organizatie, voluntari, cele disponibile *ad hoc*)

Tipuri de aplicatii

- Avantaje fata de alte abordari:
  - Dependenta mult redusa
    fata de dispozitive individuale si de sub-retele
  - Rezilienta imbunatatita
    - O resursa este disponibila in copii multiple
  - Scalabilitate mai mare
  - Abilitatea de a oferi un serviciu,
    atunci cind cererea este mare
  - Rezistenta la atacuri de tip DoS (*Denial Of Service*)

Tipuri de aplicatii

- Probleme:
  - Vulnerabilitati la atacuri mascate (*masquerade*)
  - Vulnerabilitati la atacuri de poluare (*pollution*)
  - Drepturile de autor asupra continutului digital

Tipuri de aplicatii

- Dezavantaje/probleme:
  - Arhitecturile P2P sunt probabilistice
    - Localizarea impredictibila a resurselor
    - Resursele sunt volatile
  - Inexistenta unui control centralizat
    - Probleme privind impunerea unei autoritati asupra aplicatiilor, continutului si utilizatorilor
    - Dificultati in detectarea si identificarea utilizatorilor (aspecte anti-sociale)
  - Diverse probleme noi de securitate
  - Folosirea in mod abuziv a unor resurse/dispozitive
  - Incurajarea folosirii sistemelor P2P in scop abuziv si ilegal
  - Lipsa increderii la nivel comercial, de afaceri

Tipuri de aplicatii

- Distribuirea de continut prin P2P
  - Sisteme P2P de “interschimb de fisiere”
    - Nodurile transfera un fisier la un moment dat
    - Se ofera facilitati pentru realizarea unei retele P2P si pentru cautarea & transferul de fisiere intre noduri
    - Nu se ofera suport pentru securitate, disponibilitate si persistenta
    - Exemple: Napster, KaZaA, Gnutella

Tipuri de aplicatii

- Distribuirea de continut prin P2P
  - Sisteme P2P pentru publicarea & stocarea continutului
    - Utilizatorii pot publica, stoca si distribui continut digital, pe baza unor drepturi de acces (privilegii)
    - Se focalizeaza asupra securitatii si persistentei
    - Unele ofera si facilitati privind colaborarea intre utilizatori
    - Exemple: Scan, Groove, Freenet, MojoNation, Tangler

Tipuri de aplicatii

- Distribuirea de continut prin P2P
  - Infrastructuri pentru:
    - Dirijare & localizare:
      Chord, CAN, Pastry, Tapestry, Kademlia
    - Anonimitate:
      Onion Routing, ZeroKnowledge Freedom, Tarzan
    - Managementul reputatiei:
      Eigentrust, PeerTrust

Infrastruc. (localizare & dirijare)

- Mecanismele de localizare si dirijare ce pot fi adoptate depind de:
  - topologia
  - structura
  - gradul de centralizare

  ...ale retelei suprapuse, acoperitoare (*overlay network*)

Infrastruc. (localizare & dirijare)

- Aspecte privind centralizarea:
  - Arhitecturi pur descentralizate
    toate nodurile realizeaza exact aceleasi activitati, jucind simultan roluri de servere si clienti, fara a beneficia de o coordonare centrala
    Nodurile se numesc si servents (SERVers + cliENTS)

Infrastruc. (localizare & dirijare)

- Aspecte privind centralizarea:
  - Arhitecturi partial centralizate
    unele noduri au un rol mai important
    (*e.g.*, stocind indecsi locali
    pentru fisierele partajate)
    - Nodurile devin supernoduri conform politicilor fiecarui sistem P2P folosit
    - Rolul de supernod este stabilit dinamic

Infrastruc. (localizare & dirijare)

- Aspecte privind centralizarea:
  - Arhitecturi descentralizate hibride

    exista un server central facilitind interactiunea intre noduri, mentinind cataloage de meta-date ale fisierelor
    - Serverele pot identifica si verifica nodurile de stocare
    - Sistemele se mai numesc *broker mediated*

Infrastruc. (localizare & dirijare)

- Aspecte privind structura retelei:
  - Nestructurata

    plasarea continutului este complet independenta de topologia retelei suprapuse
    - Continutul trebuie localizat
    - Strategii de cautare prin “forta bruta”: inundarea retelei – cereri propagate via BFS/DFS
    - Strategii mai sofisticate: drumuri aleatorii, probabilistice, indici de dirijare etc.

Infrastruc. (localizare & dirijare)

- Aspecte privind structura retelei:
  - Structurata

    topologia este controlata, iar fisierele (sau pointerii la ele) sunt plasate in locatii precise
    - Se realizeaza o asociere (*mapping*) intre continut (identificatorul de fisier) si locatie (adr. nodului) – un gen de tabela de rutare distribuita
    - Cautarile exacte (*exact-match queries*) pot fi realizate in mod scalabil
    - Structura folosita la dirijarea eficienta a mesajelor e dificil de mentinut in cazul unor noduri tranzien-te, cu rata mare de atasare/deconectare de la retea

Infrastruc. (localizare & dirijare)

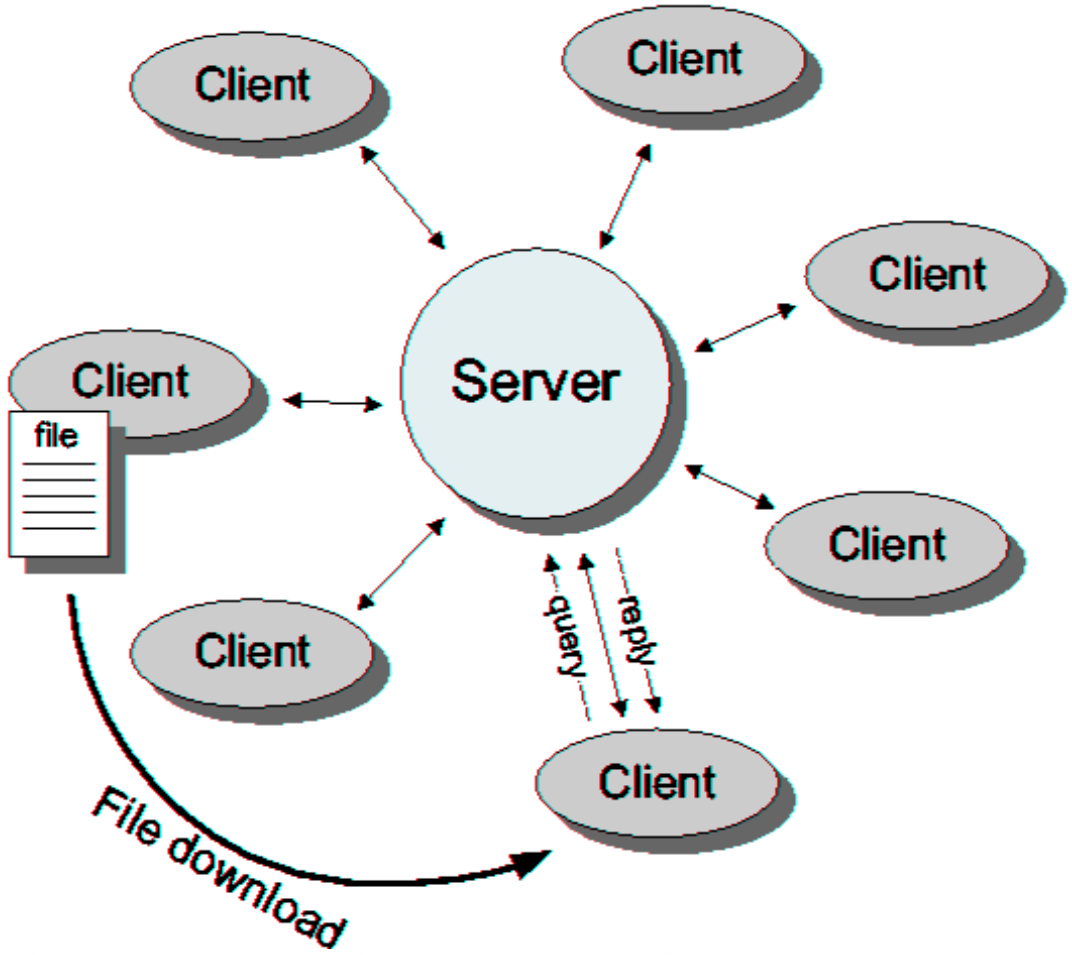
- Aspecte privind structura retelei:
  - Slab structurata (*loosely structured*)
    desi localizarea continutului nu e complet specificata, aceasta este afectata de dirijare
    - Categorie aflata intre retelele structurate si cele nestructurate

Infrastruc. (localizare & dirijare)

	Centralizare		
	Hibridă	**Parțială**	**Absentă**
Nestructurată	Napster Publius	KaZaA Morpheus Gnutella Edutella	Gnutella FreeHaven
Infrastruc. structurată			Chord, CAN Tapestry, Pastry
Sisteme structurate			OceanStore Scan, PAST Kademlia Tarzan

Arhitecturi nestructurate

- Descentralizate hibride
  - Fiecare calculator client stocheaza continut (fisiere) partajat(e)
  - Serverul central mentine o tabela cu conexiunile utiliz. inreg. (IP, latime de banda,...) + o tabela cu lista fisierelor fiecarui utiliz. & meta-date



Arhitecturi nestructurate

- Descentralizate hibride
  - Clientul se conecteaza via TCP la serverul central
  - Clientul transmite lista fisierelor sale partajate spre server (*upload*)
  - Clientul trimite interogarea (cuvinte-cheie) la server
  - Se selecteaza cele mai “bune” raspunsuri corecte (*ping*-uri)
    - Utilizatorul alege cea mai buna rata de transfer

Arhitecturi nestructurate

- Descentralizate hibride
  - Usor de implementat
  - Localizarea fisierelor e rapida si eficienta
  - Apar vulnerabilitati la atac & erori tehnice
  - Sistemele nu sunt scalabile,
    unele nu suporta adaugarea altor servere
    (lista serverelor disponibile este statica)

  - Exemple: Napster, Publius

Arhitecturi nestructurate

- Descentralizate pure
  - Se construieste o retea acoperitoare (*overlay*) cu propriile mecanisme de rutare prin IP
  - Nu exista o coordonare centrala
  - Utilizatorii se conecteaza via o aplicatie ce are rol dublu (client + server) – servent
  - Comunicarea intre serventi se bazeaza pe un protocol la nivel de aplicatie, cu 4 tipuri de mesaje:
    - **Ping** – cerere ca un nod sa se anunte
    - **Pong** – replica la mesajul *ping* (IP, port, numarul & marimea fisierelor gazdei)
    - **Query** – cerere de cautare (sir de caractere + viteza minima de transfer)
    - **Query hits** – raspuns (IP, port, viteza, nr.fis., index fis.)

Arhitecturi nestructurate

- Descentralizate pure
  - Cautarea se realizeaza prin inundare (*flooding*)
    - Daca nu ai fisierul dorit, intreaba-i pe *N* (*e.g.*, *N*=7) dintre vecini
    - Daca nici ei n-au fisierul, vor intreba pe vecinii lor, in maxim *H hop*-uri (uzual, *H*=10)
    - Pe calea de intoarcere, se vor intoarce raspunsurile (nu continutul fisierelor)
  - Fiecare mesaj are un TTL atasat
  - Un nou nod al retelei foloseste un nod de *bootstrap* pentru a se inregistra (*join*)
  - Exemplu: Gnutella

Arhitecturi nestructurate

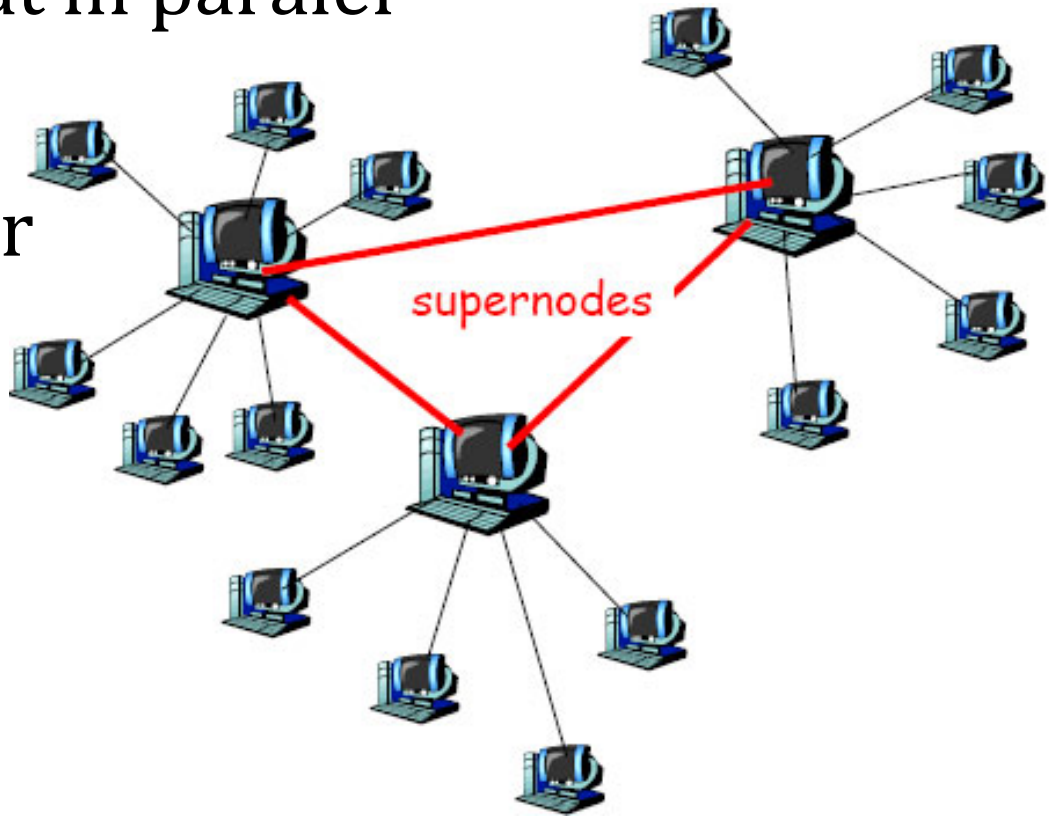
- Partial centralizate
  - Folosesc conceptul de supernod:
    are activitati de servire a unei sub-retele P2P (indexare, *caching*)
  - Nodurile sunt alese automat ca fiind supernoduri daca au suficienta latime de banda si putere computationala
  - Toate cererile se trimit initial la supernoduri
  - Avantaje: timpul descoperirii resurselor e mai redus + eterogenitatea este exploatata
  - Exemplu: KaZaA

Arhitecturi nestructurate

- Partial centralizate
  - Software-ul (protocolul) KaZaA este proprietar
  - Datele de control P2P sunt criptate
  - Mesajele folosesc HTTP ca protocol de transfer
  - Un nod e fie supernod, fie asignat unui supernod
  - Un supernod are 100-150 noduri-copil
  - O retea poate avea ~30000 supernoduri
  - Fiecare supernod are conexiuni TCP cu 30-50 supernoduri
  - Pentru fiecare fisier, se mentin meta-date (nume, dimens., *content hash*, descriptor de fisier)
  - *Content hash*-ul e folosit pentru cautarea altei copii unui fisier partial transferat

Arhitecturi nestructurate

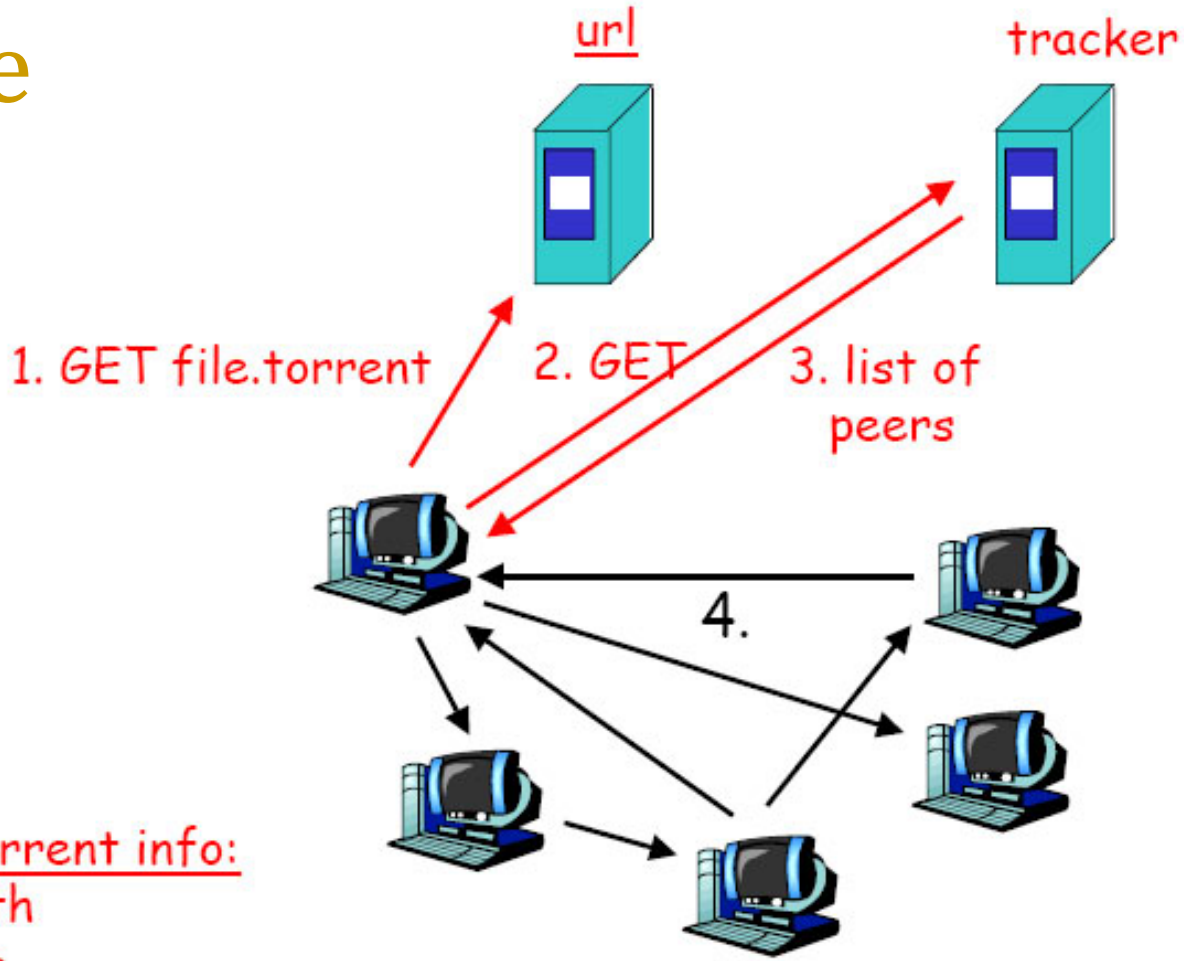
- Partial centralizate
  - Daca un fisier este gasit pe mai multe noduri, transferul poate fi realizat in paralel
    - Copiile identice se identif. via *content hash*
  - Diferite portiuni din fisier sunt transferate de pe noduri diferite
  - Pentru transferuri intrerupte, se realizeaza o recuperare automata (*automatic recovery*)



Arhitecturi nestructurate

- Partial centralizate

BitTorrent



file.torrent info:
- length
- name
- hash
- url of tracker

Arhitecturi nestructurate

- Problema:
  - Noduri ale caror adrese IP sunt disponibile via NAT
    - Nu pot fi servere TCP pentru reteaua P2P
    - Solutie partiala: *reverse call*
      - A vrea sa transfere de la B, iar B foloseste NAT
      - A si B au conexiune TCP cu serverul C (cu IP rutabil)
      - A poate cere lui B, via C, sa realizeze o conexiune TCP de la B la A
      - A poate trimite o cerere lui B, iar B ii ofera fisierul
    - Ce se intimpla daca si A si B utilizeaza NAT?

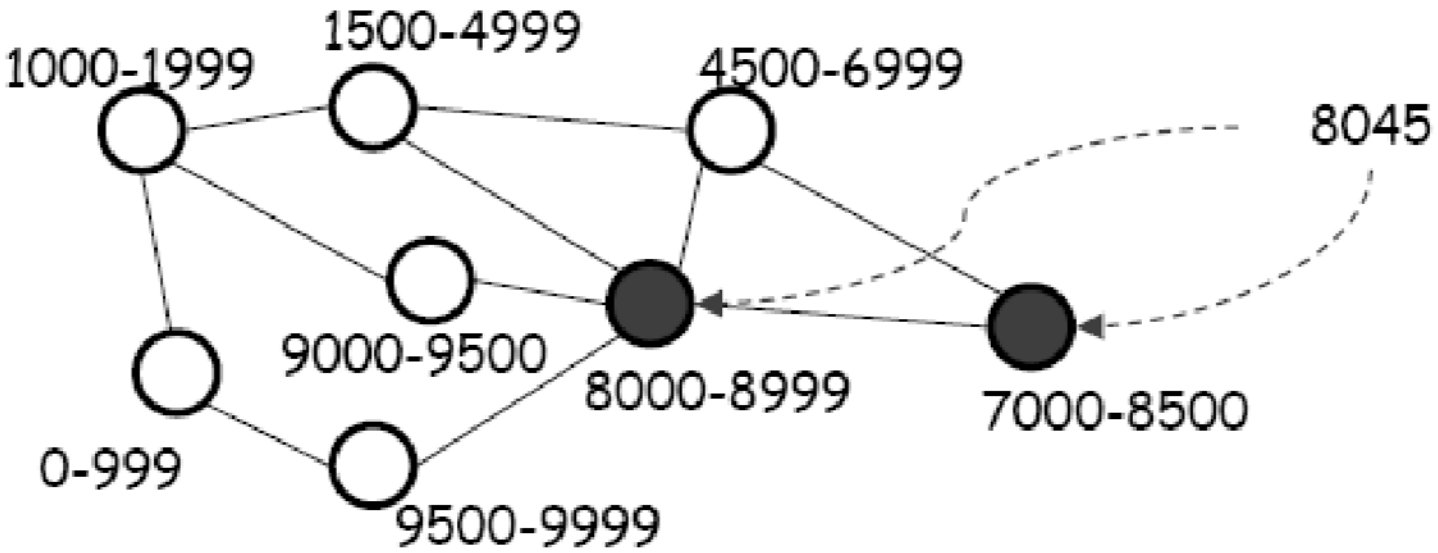
Arhitecturi nestructurate

- Problema:
  - *Flash crowd*: o crestere neasteptata de cereri pentru o resursa particulara
    - Continutul dorit era “rece” si nu exista suficiente copii incarcate in *cache*
    - Cit timp ia unui utilizator sa localizeze fisierul?
    - Cite mesaje va primi un nod datorita cautarilor realizate de alte noduri?
    - Se poate folosi un protocol de cautare generic, bazat pe TTL

Arhitecturi structurate

- Aspect de interes: **localizarea continutului**
- Idee: asignarea unui noduri particulare ce contin (pointeri la) continuturi particulare
- Dorim ca responsabilitatile sa fie distribuite mai multor noduri ale retelei de acoperire, intr-un mod adaptiv
- Fiecarei resurse i se asociaza o cheie-unica via o functie *hash*: *h* (“Curs retele”) → 8045
- Intervalul de valori ale functiei *hash* se distribuie in reteaua P2P
- Fiecare nod trebuie sa “cunoasca” locatia macar a unei singure copii a fiecarei resurse pentru care functia sa *hash* ia valori in intervalul lui

Arhitecturi structurate



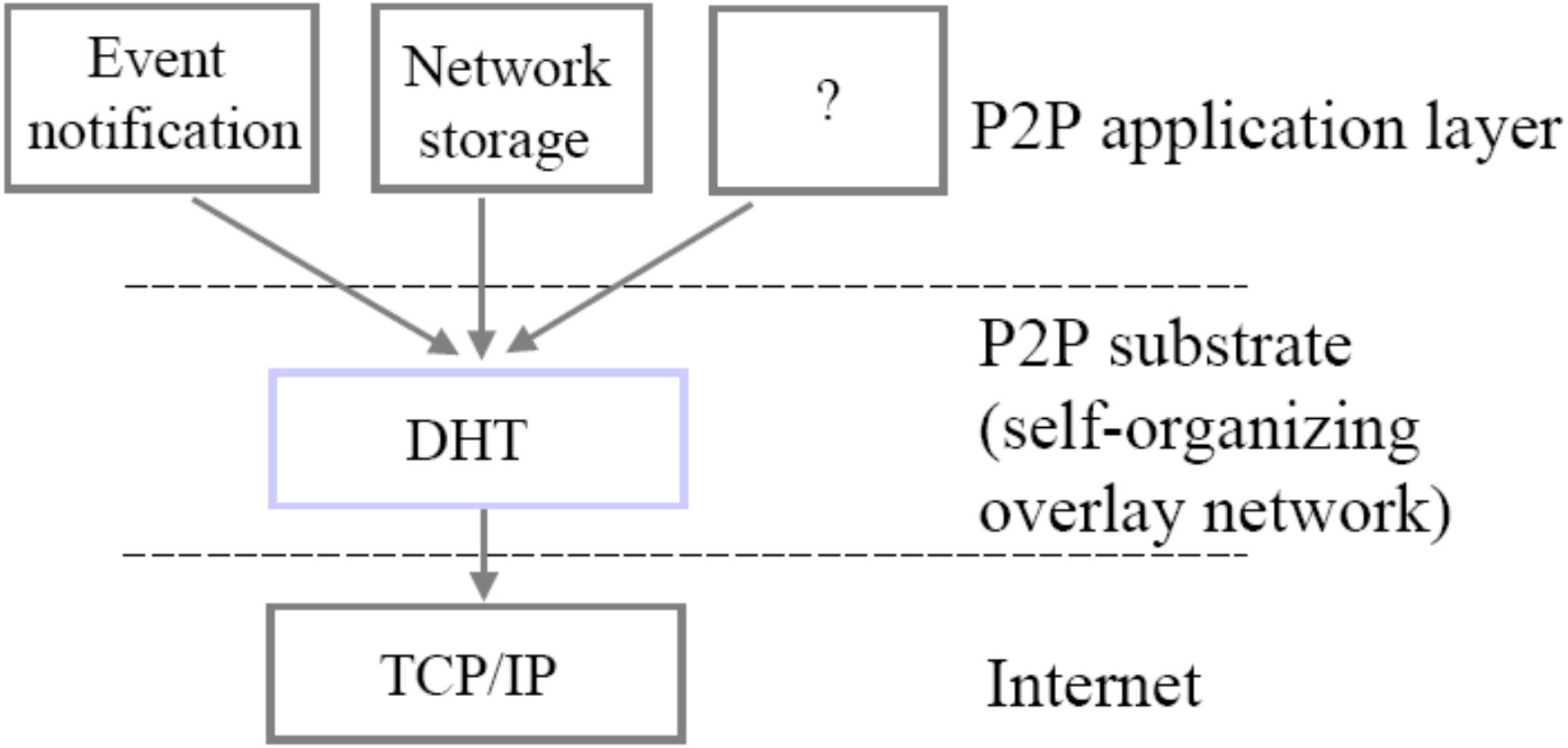
- Nodurile pot mentine in *cache*-ul propriu o copie a fiecarei resurse pe care trebuie sa o “cunoasca”
- Nodurile pot stoca doar pointeri spre nodurile ce stocheaza resursele

Arhitecturi structurate

- Aspect de interes: **dirijarea**
- Pentru fiecare resursa, un nod ce "cunoaste" resursa trebuie sa fie accesat pe calea cea mai "scurta"
  - De catre un nod de interogare
  - De catre nodurile ce au copii ale resurse (cind se folosesc pointeri de localizare)
- Abordarile de sisteme P2P structurate difera prin strategia de dirijare
  - Orice functie *hash* "buna" poate fi suficienta
- Se ofera un API pentru tabelele distribuite de *hash*-uri (DHT – *Distributed Hash Table*)
  - Dind o cheie *k*, API-ul va returna adresa IP a nodului responsabil pentru valoarea cheii *k*

Arhitecturi structurate

Organizare stratificata:

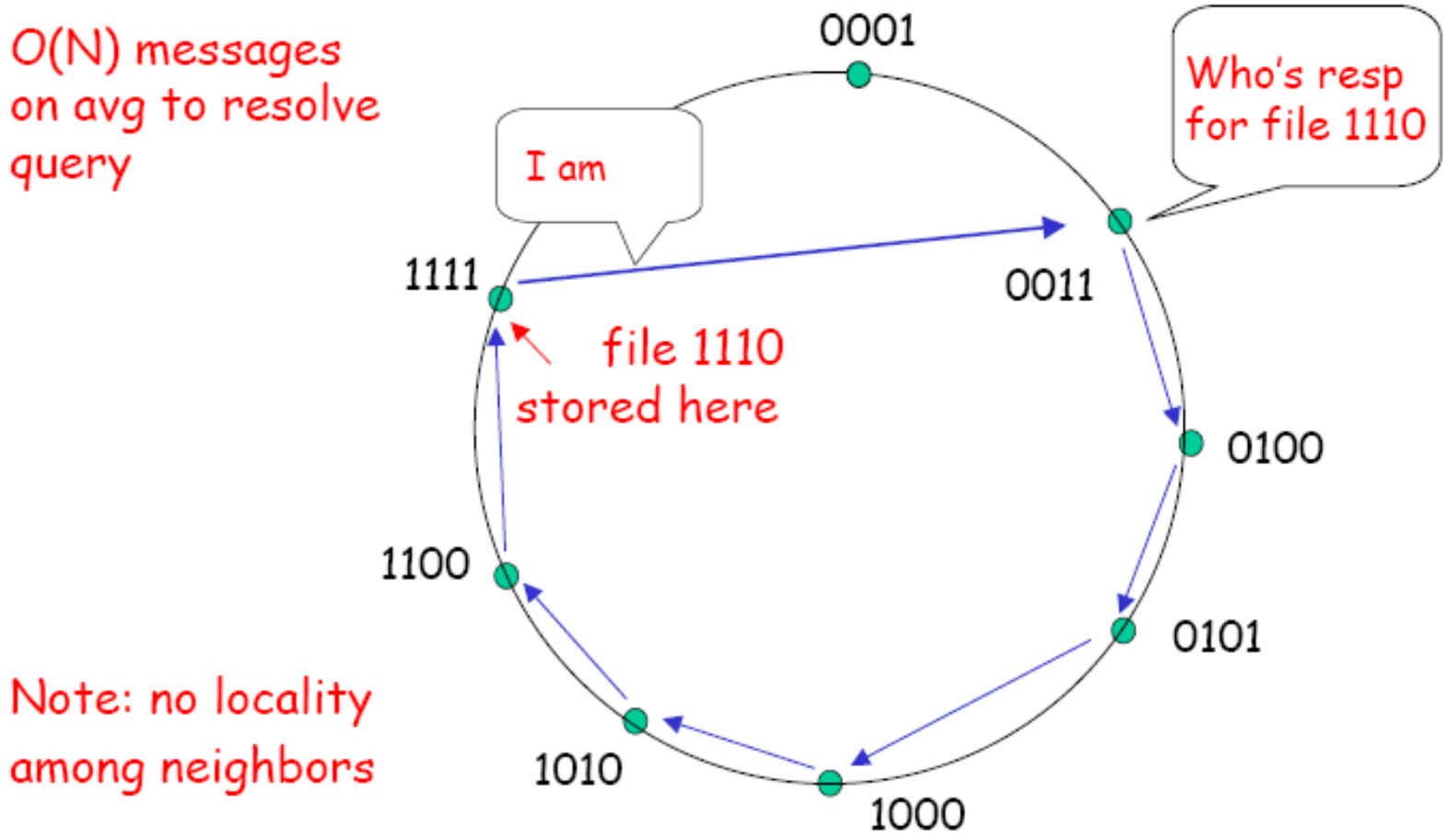


Arhitecturi structurate

- Aspect de interes: **consistenta *hash*-ului**
- Reteaua de acoperire este un cerc
- Fiecare nod are un identificator ales aleatoriu
  - Cheile fac parte din acelasi spatiu de valori ale ID-urilor
- Succesorul unui nod in cadrul cercului este nodul cu urmatorul cel mai mare ID
  - Fiecare nod stie adresa IP a nodului succesor
- Cheia este stocata la cel mai apropiat succesor

Arhitecturi structurate

Realizarea unei interogari:



Arhitecturi structurate

- Parasirea retelei de catre noduri:
  - Fiecare nod trebuie sa cunoasca macar 2 succesori
  - Daca un succesor pleaca, trebuie folosit urmatorul
  - Se detecteaza urmatorul succesor din lista de succesori si se actualizeaza aceasta lista
- Aparitia de noi noduri in retea:
  - Pentru fiecare nod nou, se stabileste un ID
  - Se interogheaza fiecare nod *N* pentru a cunoaste succesorul nodului nou
  - Predecesorii trebuie sa-si actualiz. listele de succesori
  - Fiecare nod trebuie sa-si stie predecesorii
- Reteaua *overlay* ≡ cerc avind corzi (*chords*) scurte pentru a se cunoaste predecesorii & succesorii

Arhitecturi structurate

- Slab structurate
  - Nodurile pot estima ce noduri stocheaza resursele cautate
    - Se evita *broadcast*-urile oarbe
    - Se foloseste o propagare in lant (*chain mode propagation*): fiecare nod ia decizii locale privitoare la care va fi nodul urmator interogat
  - Exemplu: Freenet
  - Cautarea unui fisier presupune utilizarea unei chei si a unui *timeout* (*hops-to-live*)
  - Tipuri de mesaje: *data insert*, *data request*, *data reply*, *data failed*

Arhitecturi structurate

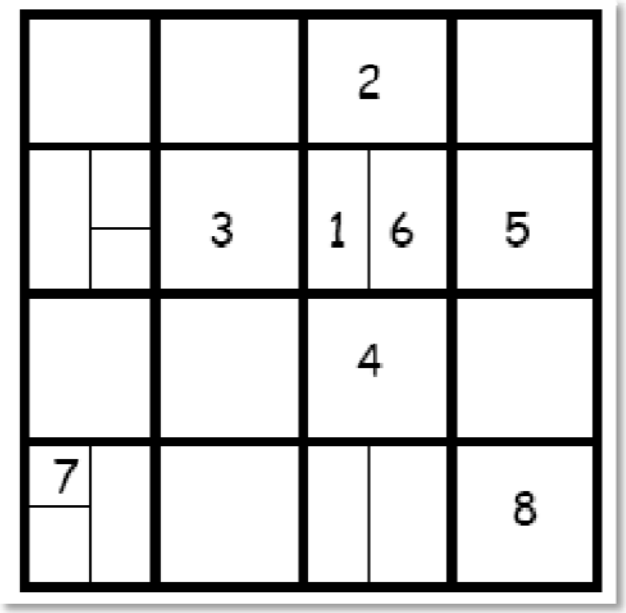
- Tabela de rutare distribuita + *tabela finger*
  - Exemplu: Chord
  - Nodurile au asociate ID-uri unidimensionale (*e.g.*, se pot folosi *hash*-uri ale adreselor IP)
  - Intervalul acoperit de un nod este de la ID-ul precedent la ID-ul propriu (*modulo* dimensiunea maxima)
  - O singura eroare in reteaua fizica poate conduce la erori multiple in cadrul retelei P2P de acoperire

Arhitecturi structurate

- Spatiu de coordonate D-dimens.
  - Exemplu: CAN (*Content Addresable Network*)
  - Fiecare nod stocheaza o parte (zona) a tabelei *hash* + informatii despre zonele adiacente
  - Cererile de inserare, localizare, stergere a unei chei sunt dirijate via zonele intermediare spre nodul ce mentine zona continind acea cheie
  - Fiecare cheie *k* e determinist asociata unui punct *P* din spatiul cartezian de *D* dimensiuni

Arhitecturi structurate

- Spatiu de coordonate *D*-dimens.
  - *D*=2
  - Vecinii nodului 1: 2, 3, 4, 6
  - Vecinii nodului 6: 1, 2, 4, 5
  - Nodurile 7 si 8 sunt vecine
  - Numarul asteptat de noduri: O(*D*)
  - Rutarea are loc alegind distanta carteziana cea mai mica
  - Se folosesc diversi algoritmi pentru atasarea de noi noduri & parasirea retelei

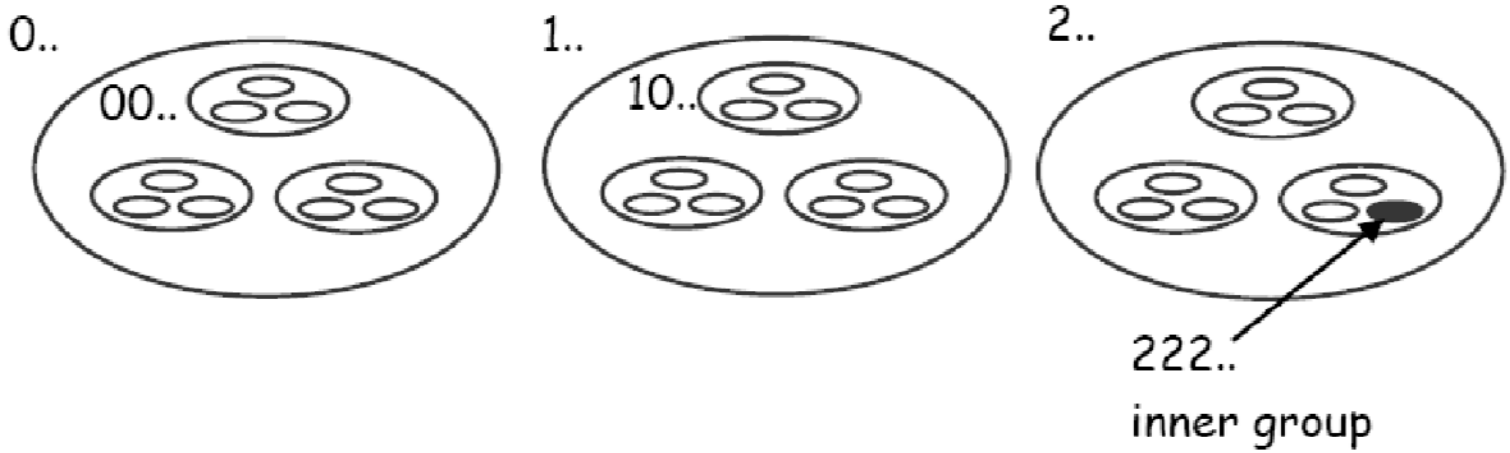


Arhitecturi structurate

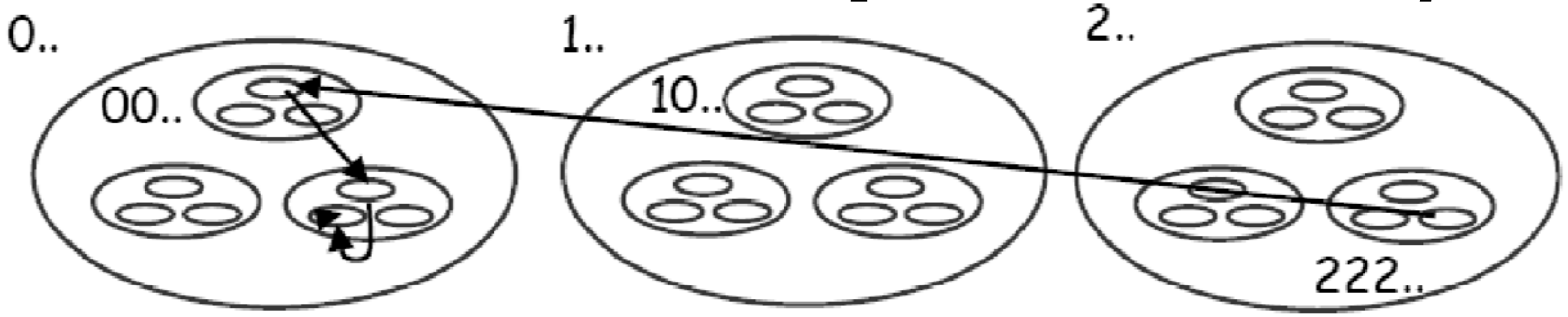
- Folosirea structurii de date *Plaxton mesh*
  - Se mentin pointeri la nodurile ale caror ID-uri se potrivesc cu elementele unei structuri arborescente de prefixuri ale ID-urilor
  - Exemple: Pastry, Kademlia, Tapestry
  - Nodurile si cheile au ID-uri de *N* cifre, scrise in baza *B* (de exemplu, *B*=3)
  - O cheie e stocata in nodul cu cel mai apropiat ID
  - Adresarea nodurilor se face prin grupuri imbricate
  - Nodurile dintr-un grup stiu adresele IP ale celorlalte
  - Fiecare nod stie IP-ul unui nod delegat al altui grup

Arhitecturi structurate

Folosirea structurii de date *Plaxton mesh*



Cautarea unei chei se face prin *divide-et-impera*

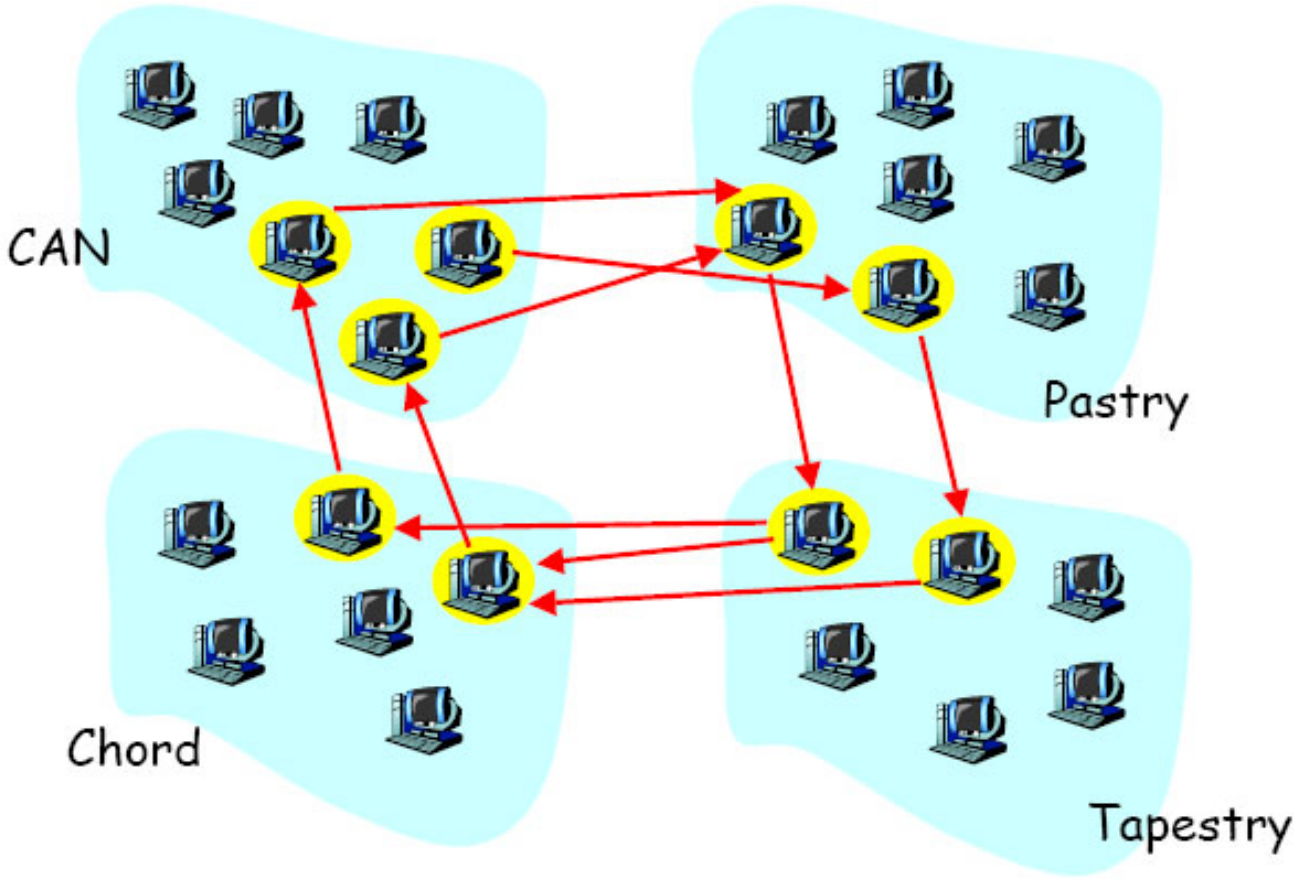


Arhitecturi structurate

- Folosirea structurii de date *Plaxton mesh*
  - Pentru Pastry, fiecare nod are un identif. pe 128 biti
    - In baza 16
    - 16 subgrupuri pentru fiecare grup
  - Fiecare nod mentine o tabela de rutare si o multime a nodurilor frunza
    - Tabela de rutare este folosita pentru a stabili noduri-delegat ale fiecarui grup
  - Mecanismele de rutare & localizare se bazeaza pe *Plaxton mesh*, dar sistemul Tapestry le extinde pentru populatii de noduri P2P dinamice

Arhitecturi ierarhice

- Nodurile *peer* pot fi organizate in grupuri
- Localizarea se poate face initial in cadrul grupului, apoi in alte grupuri de noduri



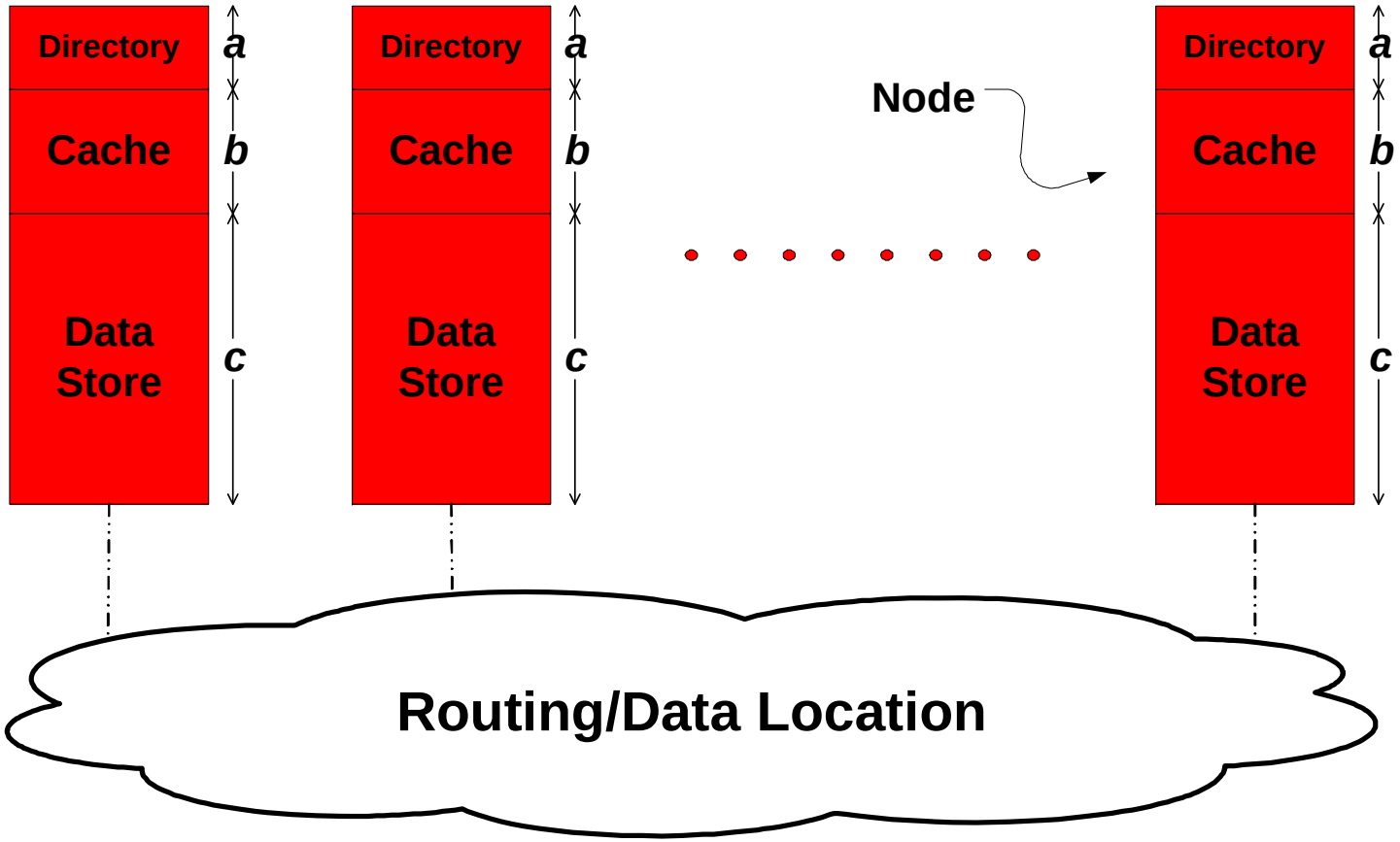
Aplicatii bazate pe DHT

- Partajare de fisiere
  - Exemple: Overnet bazat pe Kademlia
  - Probleme: replicare optimala, *load balancing*, cautare pe baza de cuvinte-cheie, *caching*
- Stocare persistenta a fisierelor
  - Sisteme de fisiere P2P: Oceanstore, Farsite
  - Exemplu: PAST bazat pe Pastry
- Managementul dispozitivelor mobile
  - Vezi cursul viitor
- SOS
  - Prevenirea atacurilor DoS

Aspecte tehnice

Spatiu distribuit de memorare

(*distributed shared memory* – DSM)



Aspecte tehnice

Variante de sisteme, conform DSM:

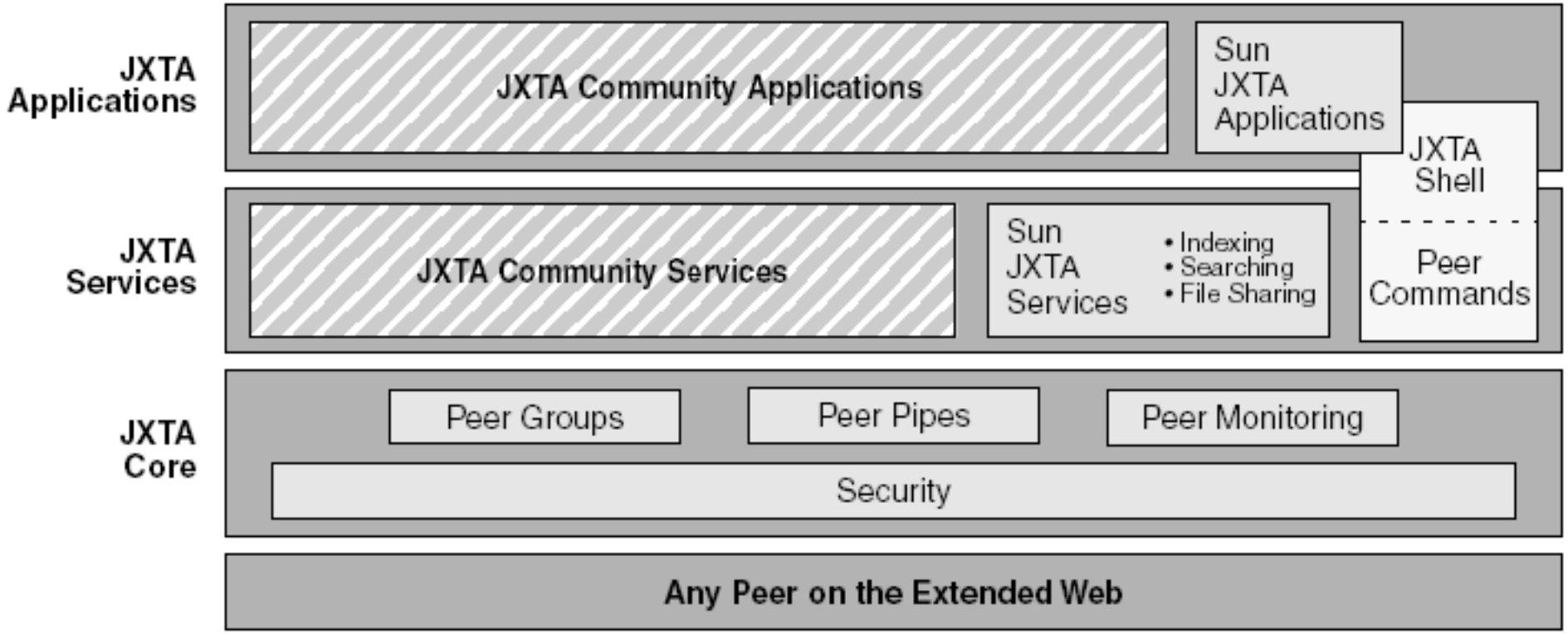
	Directory	Cache	Data Store	Examples
Data-Store-Only	No	No	Yes	**N/A**
Cache-Only	No	Yes	No	**N/A**
Directory-Less	No	Yes	Yes	**PAST**
Cache-Less	Yes	No	Yes	**Naspter**
Data-Store-Less	Yes	Yes	No	**Freenet**
Fullness	Yes	Yes	Yes	**OceanStore**

Aspecte tehnice

- Descoperirea resurselor
  - Localizare & dirijare
  - Folosirea meta-datelor
  - Cautari semantice, nu bazate pe cuvinte-cheie
- Performanta
  - Scalabilitate, toleranta la defecte etc.
- Fiabilitatea (*reliability*)
- *Zero-administration*
- Increderea (*trust*)
- Rezistenta la cenzura + anonimitatea
- Securitatea
  - Autentificare, transfer sigur al datelor

Aplicatii

- JXTA – **www.jxta.org**
  - Mediu de dezvoltare a sistemelor & aplicatiilor P2P
  - Bazat pe Java, disponibil in regim *open source*



Rezumat

- Paradigma *peer-to-peer* (P2P)
  - Preliminarii
  - Definitii
  - Caracterizare
  - Tipuri de aplicatii
  - Infrastructuri
  - Aspecte tehnice & aplicatii

Intrebari?