Fitxategi-sistema banatuak

Sistema Banatuak

Mikel Larrea, KAT Saila

Fitxategi-sistema banatuak

- Sarrera
- Eredua
- Direktorio-zerbitzua
- Fitxategi-zerbitzua
- Adibideak
  - NFS (Network File System)
  - AFS (Andrew File System)
  - Coda (Constant Data Availability)
  - CIFS (Common Internet File System)
    - SMB (Server Message Block): *Samba*

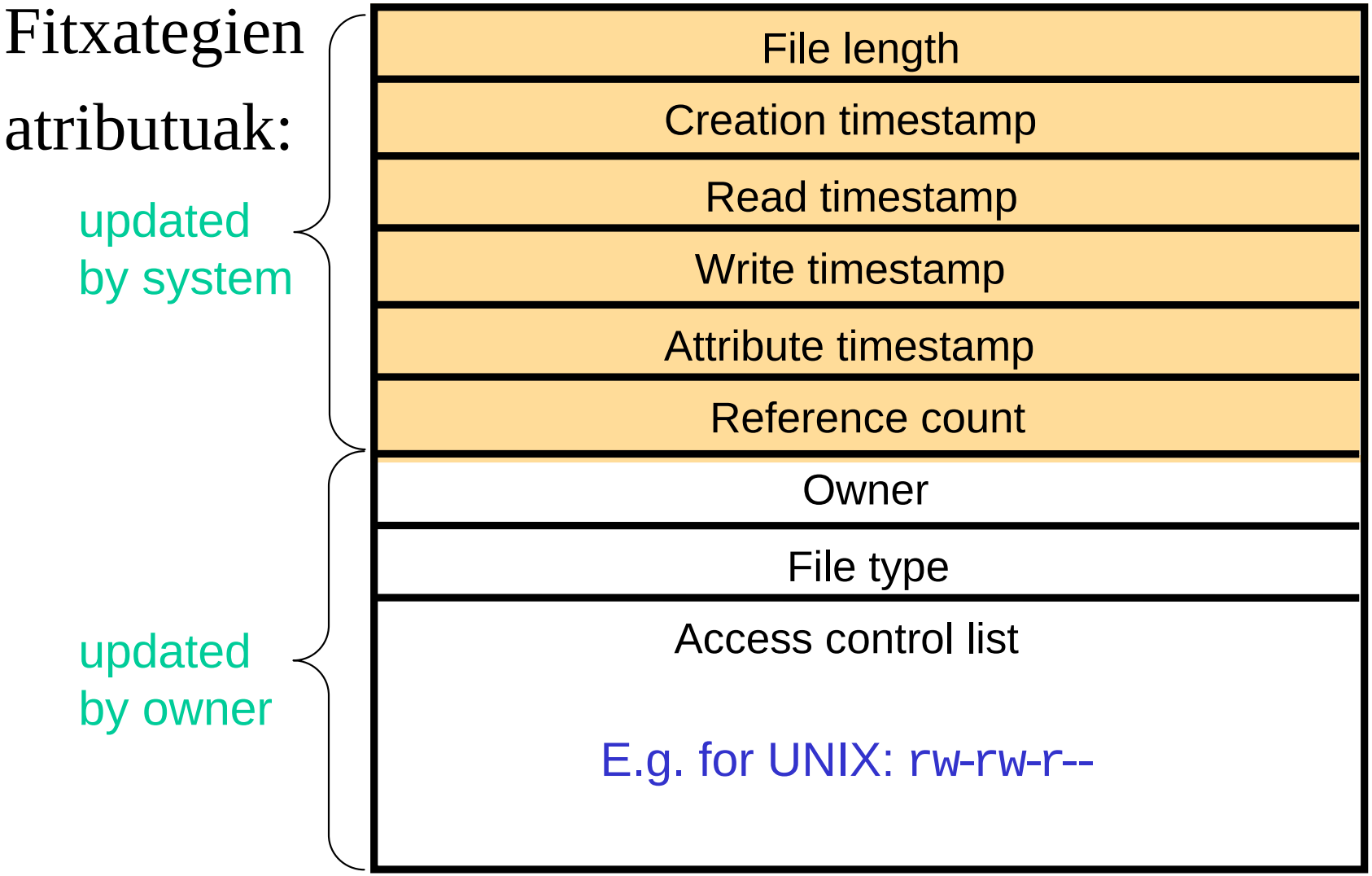
1 Sarrera

- Fitxategi-sistemak eta datu-baseak banatzeko eta konpartitzeko lehen hautagaiak izan dira
  - sare lokalentzako produktuak aspalditik daude: NFS
- Fitxategi-sistema banatuen kudeaketa bi zerbitzu desberdinetan oinarritzen da:
  - direktorio-zerbitzua (katalogoa)
  - fitxategi-zerbitzua
- Eraginkortasuna lehen helburua da
  - informazioaren eskuragarritasunaren eta sendotasunaren arteko konpromisoa
    - *caching* (bezeroak), banaketa eta erreplikazioa (zerbitzariak)

1 Sarrera (2)

- Fitxategi-sistemen ezaugarri orokorrak:
  - informazioaren euskarri iraunkorra eskaintzen dute
  - fitxategiak identifikatzeko izen-esparrua kudeatzen dute, normalean hierarkikoa dena:
    - katalogoak, azpikatalogoak
  - aldi bereko atzipenak kudeatzen dituzte
  - erabiltzaile anitzeko sistemetan erabiltzaileen arteko babesa eskaintzen dute:
    - idatzi, irakurri eta exekutatzeko eskubideak
  - fitxategien edukia:
    - datuak (8 biteko byteak normalean)
    - atributuak (*metadata*)

1 Sarrera (3)


Fitxategien atributuak:
updated by system
updated by owner
File length
Creation timestamp
Read timestamp
Write timestamp
Attribute timestamp
Reference count
Owner
File type
Access control list
E.g. for UNIX: rw-rw-r--

1 Sarrera (4)

- *UNIX* fitxategi-sistemaren operazioak:

filedes = open(name, mode) *filedes = creat(name, mode)*	Opens an existing file with the given *name*. Creates a new file with the given *name*. Both operations deliver a file descriptor referencing the open file. The *mode* is *read*, *write* or both.
status = close(filedes)	Closes the open file *filedes*.
count = read(filedes, buffer, n) *count = write(filedes, buffer, n)*	Transfers *n* bytes from the file referenced by *filedes* to *buffer*. Transfers *n* bytes to the file referenced by *filedes* from buffer. Both operations deliver the number of bytes actually transferred and advance the read-write pointer.
pos = lseek(filedes, offset, whence)	Moves the read-write pointer to offset (relative or absolute, depending on *whence*).
status = unlink(name)	Removes the file *name* from the directory structure. If the file has no other names, it is deleted.
status = link(name1, name2)	Adds a new name (*name2*) for a file (*name1*).
status = stat(name, buffer)	Gets the file attributes for file *name* into *buffer*.

1 Sarrera (5)

- *UNIX* fitxategi-sistema:
  - aurreko operazioak *UNIX*-en nukleoak eskaintzen ditu
  - aplikazioek normalean liburutegiak erabiltzen dituzte:
    - C lengoaiaren sarrera/irteerako liburutegi estandarra
    - Java lengoaiaren fitxategiak erabiltzeko klaseak
  - *UNIX*-en, fitxategi-sistemak irekitako fitxategi bakoitzarentzat egoera-informazioa gordetzen du:
    - irekitako fitxategien zerrenda kudeatzen du beraz
    - zerrendako fitxategi bakoitzarentzat, irakurketa/idazketa erakuslea gordetzen du (hurrengo operazioaren posizioa)
  - atzipen kontrola fitxategia irekitzerakoan egiten da:
    - *mode* parametroa eta fitxategiaren atributuen bitartez

1 Sarrera (6)

- Fitxategi-sistema banatuen propietateak:
  - gardentasuna:
    - atzipenean: fitxategi lokalak eta urrunak berdin atzitzen dira
    - kokapenean: izen-esparru bakarra eta uniformea. Fitxategiak tokiz aldatu daitezke, aplikazioak aldaketarik jasan gabe
    - eraginkortasunean: *caching* bezeroan
    - eskalagarritasunean
  - erreplikazioaren kudeaketa:
    - eskalagarritasuna (karga-banaketa)
    - eskuragarritasuna, hutsegite-tolerantzia
  - hardware eta sistema eragilearen heterogeneotasuna:
    - irekitasuna, zabalgarritasuna

1 Sarrera (7)

- Fitxategi-sistema banatuen propietateak (2):
  - hutsegite-tolerantzia (erreplikaziorik gabe):
    - operazio idenpotenteak:

$$f(f(x)) = f(x)$$

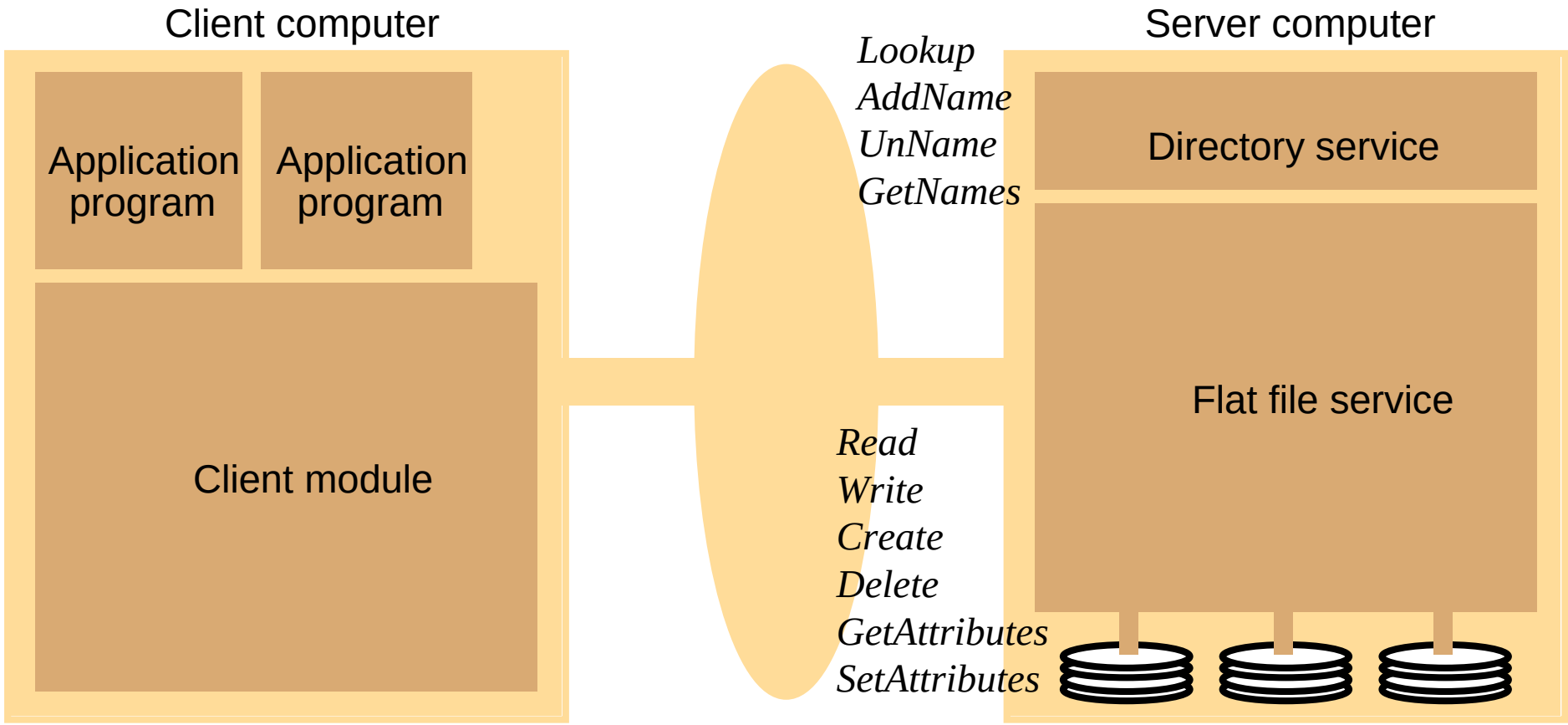
    - egoera gabeko zerbitzariak
  - sendotasuna:
    - aldi bereko atzipenen kudeaketa (idazketen kasuan)
    - sistema zentralizatuen semantika *(UNIX)* eskaini nahi da. Batzutan ezinezkoa izaten da (*caching*, erreplikazioa)
  - segurtasuna (egiaztapen urruna):
    - atzipenen kontrolerako zerrendak + kredentzialak

1 Sarrera (8)

- Fitxategien erabileraren ezaugarriak:
  - fitxategi gehienak txikiak dira (atzipen unitatea)
  - idazketak gutxitan gertatzen dira: *caching*, erreplikazioa
  - aldi bereko atzipenak ez dira maiz gertatzen:
    - irakurle edo/eta idazle bakarra gehienetan, gutxitan *n* irakurle eta idazle bat, eta oso gutxitan *n* irakurle eta *m* idazle
    - *caching* eta erreplikazioaren kudeaketa optimista
  - atzipen sekuentziala, eta informazioaren lokaltasuna: atzipen aurreratua *(buffering)*
  - fitxategi askok bizi laburra dute (adibidez, aldi baterako fitxategiak): kudeaketa lokala

2 Eredua

- Bezero-zerbitzari egitura (3 modulu):



File service architecture

2 Eredua (2)

- Bezeroen modulua *(Client module)*:
  - aplikazioekin interfaze lokala
  - aplikazioen deiak interpretatu eta atzipen urruneko eskarietaz arduratzen da (normalean *RPC*-ak erabiliz)
  - direktorio eta fitxategi zerbitzuen kokapena ezagutzen du
  - fitxategien kopia lokalak kudeatzen ditu (*caching*)
- Fitxategi-zerbitzua *(Flat file service)*:
  - fitxategi eta katalogoen edukia eta atributuak kudeatzen ditu
  - *UFID*: fitxategi bakoitzaren identifikadore bakarra
- Direktorio-zerbitzua *(Directory service)*:
  - gardentasuna kokapenean eskaintzen du
  - (*izena*, *UFID*) motako elementuzko datu-basea da

2 Eredua (3)

- Fitxategien identifikazioa eta atzipena:
  - fitxategiaren izena erabiliz, bezeroen moduluak direktorio-zerbitzuari fitxategiaren *UFID*-a eskatzen dio
    - direktorio-zerbitzuak bueltatutako *UFID*-ak honako informazioak ditu: zerbitzaria (S), fitxategia (F), eta atzipen eskubideak (D)
  - ondoren, operazio guztiak *UFID*-a erabiliz zerbitzariari eskatzen zaizkio
  - *UFID*-ak manipulazioetatik babestu egin behar dira, aplikazioek ez ditzaten *UFID* faltsurik sortu/erabili
    - egiaztatze prozedurak (*authentication*), kredentzialetan oinarrituta

2 Eredua (4)

- Egiaztatze prozedura:
  - *UFID*-a fitxategia atzitzeko kredentziala da (*capability*)
  - fitxategi bat sortzerakoan, zerbitzariak zenbaki aleatorio bat sortzen du (R), eta fitxategiari lotzen dio
  - direktorio-zerbitzuak bezeroen moduluari bueltatzen dion *UFID*-ak ez darama R balioa, baizik eta R eta D kodifikatuz kalkulatzen duen C balioa
  - baliozko atzipena dela konprobatzeko, zerbitzariak kodifikazio funtzioa aplikatzen dio berak gordetzen duen R eta *UFID*-ak dakarren D-ri, emaitza *UFID*-ak dakarren C-kin konparatuz

3 Direktorio-zerbitzua

- Bezeroen moduluak fitxategien kokapena (zehazki, bere *UFID*-a) direktorio-zerbitzuari eskatzen dio
  - fitxategien izen sinbolikoa *(path)* pasatuz

Lookup(Dir, Name) -> FileId — throws *NotFound*	Locates the text name in the directory and returns the relevant UFID. If *Name* is not in the directory, throws an exception.
AddName(Dir, Name, FileId) — throws *NameDuplicate*	If *Name* is not in the directory, adds (*Name*, *FileId*) to the directory and updates the file's attribute record. If *Name* is already in the directory: throws an exception.
UnName(Dir, Name) — throws *NotFound*	If *Name* is in the directory: the entry containing *Name* is removed from the directory. If *Name* is not in the directory: throws an exception.
GetNames(Dir, Pattern) -> NameSeq	Returns all the text names in the directory that match the regular expression *Pattern*.

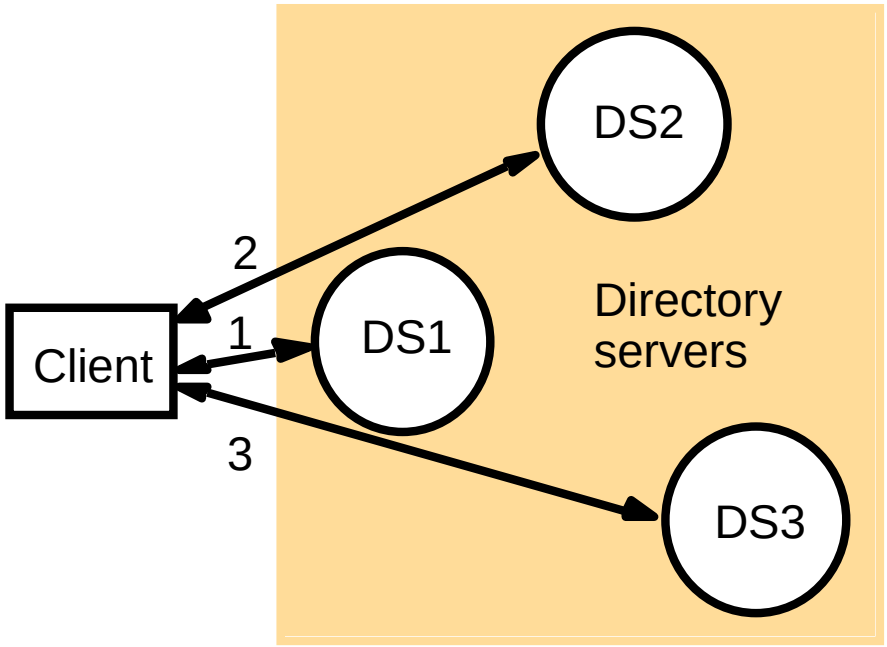
Directory service operations

3 Direktorio-zerbitzua (2)

- Izenen ebazpenaren ikuspuntutik, katalogoen egitura hierarkikoa eremutan banatzen da
  - eremu bat fitxategiaren *path*-aren zati bati lotu daiteke
  - direktorio-zerbitzari batek eremu bat baino gehiago kudeatu ditzake
  - bezeroen moduluak *(eremu, direktorio-zerbitzari)* motako taula kudeatzen du
- Izenen ebazpena:
  - iteratiboa (bezeroak ala zerbitzariak kontrolatuta)
  - errekurtsiboa

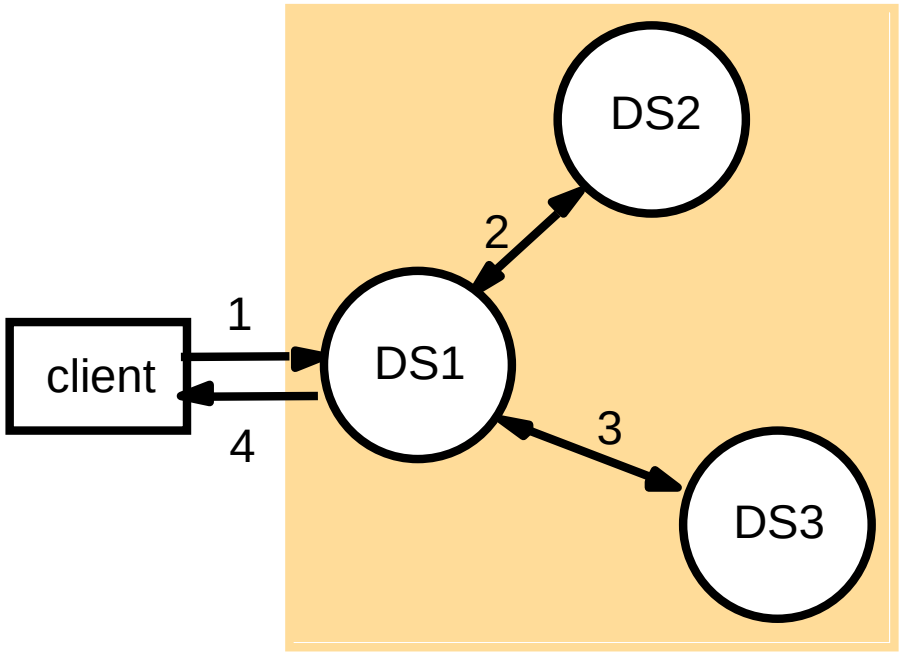
3 Direktorio-zerbitzua (3)

- Izenen ebazpen iteratiboa (bezeroen moduluak kontrolatuta):

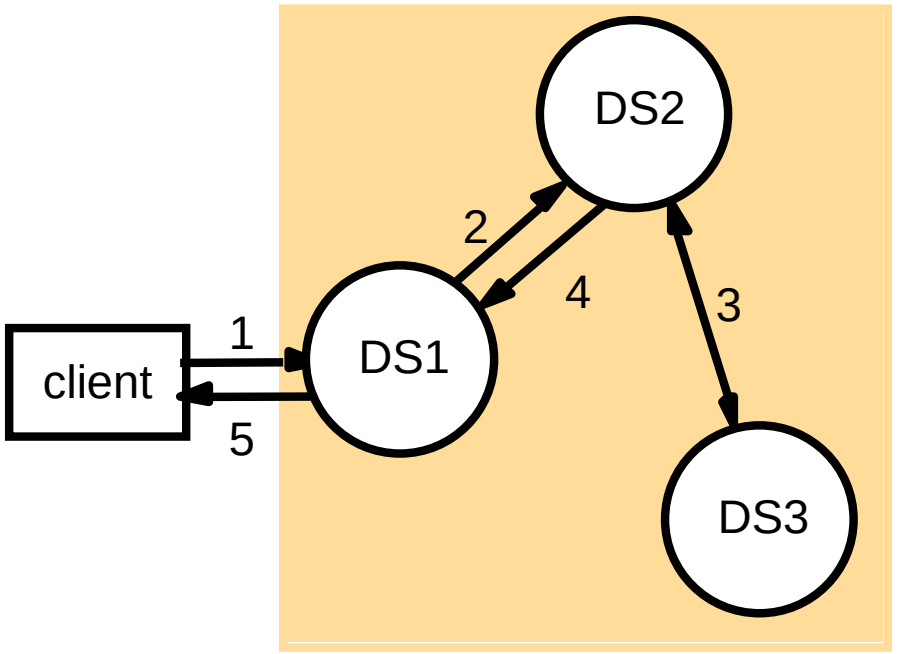


3 Direktorio-zerbitzua (4)

- Izenen ebazpen iteratiboa (zerbitzariak kontrolatuta) eta errekurtsiboa:



Non-recursive
server-controlled



Recursive
server-controlled

4 Fitxategi-zerbitzua

- Helburua: sistema zentralizatuekiko semantikarik gertuena eskaintzea, errendimendua (latentzia) asko gutxitu gabe: *caching*, erreplikazioa

Read(FileId, i, n) -> Data — throws *BadPosition*	If *1 ≤ i ≤ Length(File)*: Reads a sequence of up to *n* items from a file starting at item *i* and returns it in *Data*.
Write(FileId, i, Data) — throws *BadPosition*	If *1 ≤ i ≤ Length(File)+1*: Writes a sequence of *Data* to a file, starting at item *i*, extending the file if necessary.
Create() -> FileId	Creates a new file of length 0 and delivers a UFID for it.
Delete(FileId)	Removes the file from the file store.
GetAttributes(FileId) -> Attr	Returns the file attributes for the file.
SetAttributes(FileId, Attr)	Sets the file attributes (only those attributes that are not shaded).

Flat file service operations

4 Fitxategi-zerbitzua (2)

- Aldi bereko atzipenen kudeaketa: sendotasuna eta errendimenduaren arteko konpromisoa
  - *UNIX* semantika (*one-copy*): sistema zentralizatuetakoa
  - sesio semantika (kopia lokala irekitzean, itxieran idazten dena)
  - fitxategi aldagaitzak (ordezkapen atomikoa)
  - transakzioen semantika (*ACID* propietateak)
- Zerbitzari-motak:
  - egoera gabekoak: *NFS (Network File System)*
    - bezeroen hutsegiteek ez dute eraginik zerbitzarian
  - egoeradunak: *RFS (Remote File System)*, *AFS*, *NFSv4*
    - bezeroen hutsegiteek zerbitzarian eragina dute

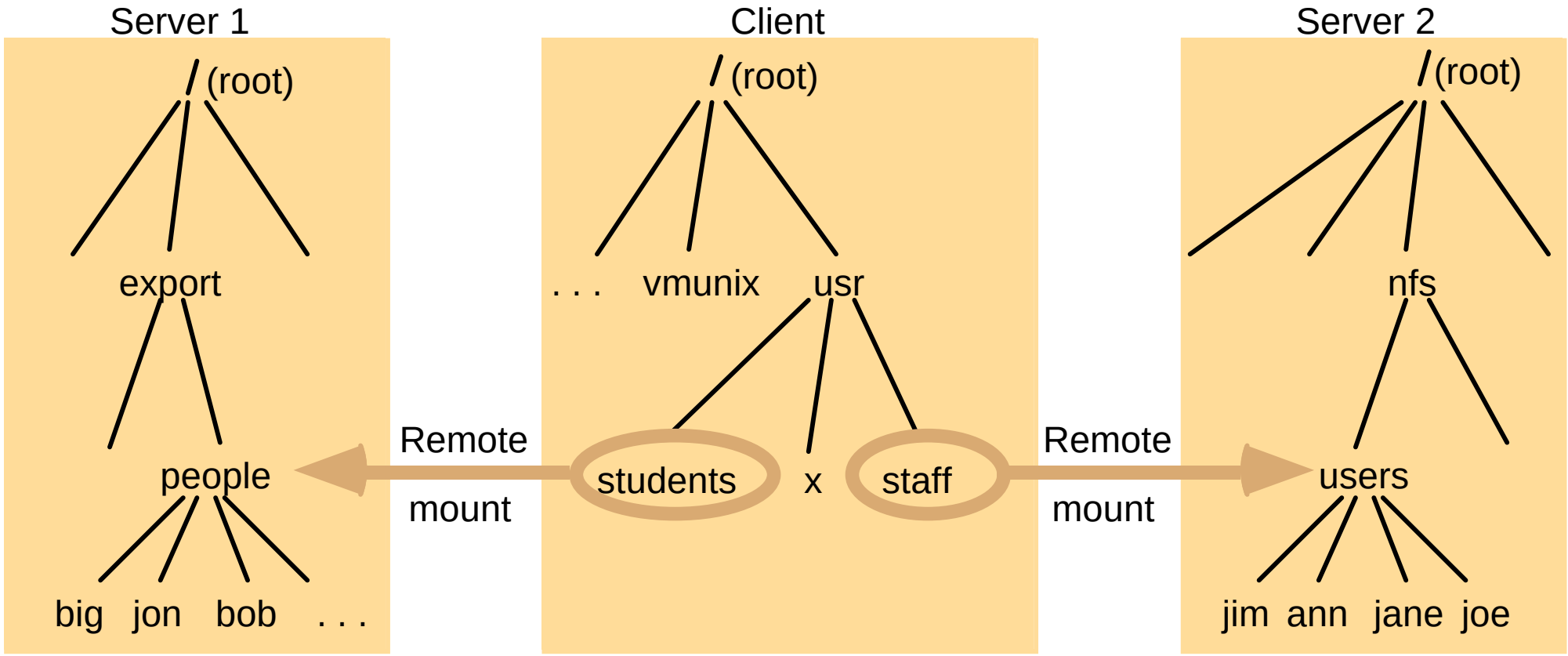
4 Fitxategi-zerbitzua (3)

- *Caching*, *buffering*:
  - bezeroen moduluak fitxategien zatiak kudeatzen dituzte, atzipen urruneko komunikazio kostua gutxitzeko
  - zerbitzariak ere bere aldetik blokeen *buffer cache* egin dezake
  - eskariak betetzerakoan, zerbitzariak irakurketa aurreratua *(buffering)* eskaintzen du
  - Sendotasunaren kudeaketa (*caching* dagoenenan):
    - *write-through*: aldaketak zuzenean zerbitzarian kopiatzen dira. Baliotasun mekanismoa *UNIX* semantika lortzeko (bezeroak edota zerbitzariak zuzenduta). Idazketa atzeratuak errendimendua hobetzeko
    - *write-on-close*: sesio semantika
    - kudeaketa zentralizatua: zerbitzariak kontrolatuta. *UNIX* semantika. Eskalagarritasun eta hutsegite-tolerantzia eskasa

5 Network File System (NFS)

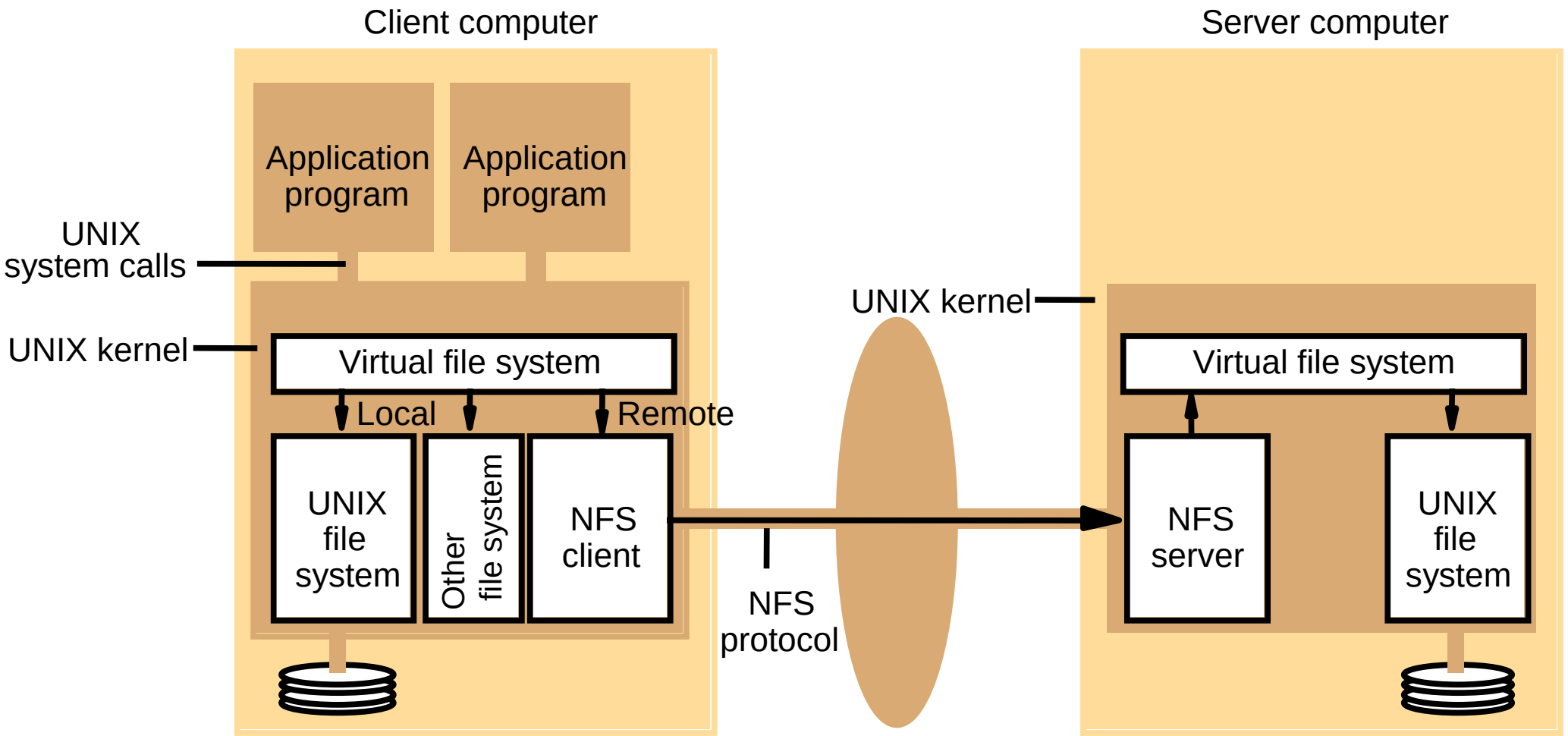
- *Sun Microsystems* enpresak garatua (1985ean)
  - hasieran *UNIX* sistema eragilerako
  - gaur egun: *Windows*, *MacOS*, *VMS*, *Linux*, *UNIX*
  - *NIS (Network Information Service)* zerbitzuarekin konbinatzen da: erabiltzaileen mugikortasuna
- Zerbitzariek katalogoak esportatu egiten dituzte, eta bezeroek muntatu:
  - *UNIX*-en: `/etc/exports` eta `/etc/fstab` fitxategietan
  - muntaia arrankatzerakoan edota eskari baten ondorioz egin daiteke *(automounter)*
- Aplikazioek *UNIX* interfazea erabiltzen dute:
  - gardentasuna atzipenean

5 NFS - Adibidea



Note: The file system mounted at */usr/students* in the client is actually the sub-tree located at */export/people* in Server 1; the file system mounted at */usr/staff* in the client is actually the sub-tree located at */nfs/users* in Server 2.

5 NFS - Arkitektura



5 NFS - Arkitektura (2)

- Direktorio-zerbitzua eta fitxategi-zerbitzua zerbitzari bakar bat osatzen dute *(NFS Server)*
- Fitxategi-zerbitzaria egoera gabekoa da:
  - hutsegiteak errazago konpontzen dira
  - atzipen kontrola: erabiltzailearen identitatea eskari bakoitzean konprobatzen da
- Operazioak “idenpotenteak” dira (*create* ezik)
- *Caching* bezeroen moduluan:
  - 16 KB blokeak (irakurketa aurreratua)
  - *write-through* semantika, idazketa atzeratuekin. Bezeroen moduluak periodikoki blokeen baliotasuna aztertzen du

5 NFS - Arkitektura (3)

- Aldi bereko atzipenen kudeaketa:
  - *UNIX* semantika (mugatua, *caching* egiten baita)
- Bi protokolo:
  - MOUNT: katalogoen muntaiarako
  - NFS: fitxategi urrunak atzitzeko
  - biak RPC protokoloan oinarritzen dira
    - RPC protokoloa bai UDP eta bai TCP erabili ditzake
    - datuen errepresentazio sendoa: XDR formatua
- Inplementazioa: *VFS (Virtual File System)*
  - irekitako fitxategien *v-node* taula: lokala bada, *i-node* bat adieraziko du, bestela *r-node* bat (*remote i-node*), bezeroen moduluak kudeatuko duelarik

5 NFS - Operazioak (1)

lookup(dirfh, name) -> fh, attr	Returns file handle and attributes for the file *name* in the directory *dirfh*.
create(dirfh, name, attr) -> newfh, attr	Creates a new file name in directory *dirfh* with attributes *attr* and returns the new file handle and attributes.
remove(dirfh, name) status	Removes file name from directory *dirfh*.
getattr(fh) -> attr	Returns file attributes of file *fh*. (Similar to the UNIX *stat* system call.)
setattr(fh, attr) -> attr	Sets the attributes (mode, user id, group id, size, access time and modify time of a file). Setting the size to 0 truncates the file.
read(fh, offset, count) -> attr, data	Returns up to *count* bytes of data from a file starting at *offset*. Also returns the latest attributes of the file.
write(fh, offset, count, data) -> attr	Writes *count* bytes of data to a file starting at *offset*. Returns the attributes of the file after the write has taken place.
rename(dirfh, name, todirfh, toname) -> status	Changes the name of file *name* in directory *dirfh* to *toname* in directory to *todirfh*
link(newdirfh, newname, dirfh, name) -> status	Creates an entry *newname* in the directory *newdirfh* which refers to file *name* in the directory *dirfh*.

5 NFS - Operazioak (2)

symlink(newdirfh, newname, string) -> status	Creates an entry *newname* in the directory *newdirfh* of type symbolic link with the value *string*. The server does not interpret the *string* but makes a symbolic link file to hold it.
readlink(fh) -> string	Returns the string that is associated with the symbolic link file identified by *fh*.
mkdir(dirfh, name, attr) -> newfh, attr	Creates a new directory *name* with attributes *attr* and returns the new file handle and attributes.
rmdir(dirfh, name) -> status	Removes the empty directory *name* from the parent directory *dirfh*. Fails if the directory is not empty.
readdir(dirfh, cookie, count) -> entries	Returns up to *count* bytes of directory entries from the directory *dirfh*. Each entry contains a file name, a file handle, and an opaque pointer to the next directory entry, called a *cookie*. The *cookie* is used in subsequent *readdir* calls to start reading from the following entry. If the value of *cookie* is 0, reads from the first entry in the directory.
statfs(fh) -> fsstats	Returns file system information (such as block size, number of free blocks and so on) for the file system containing a file *fh*.

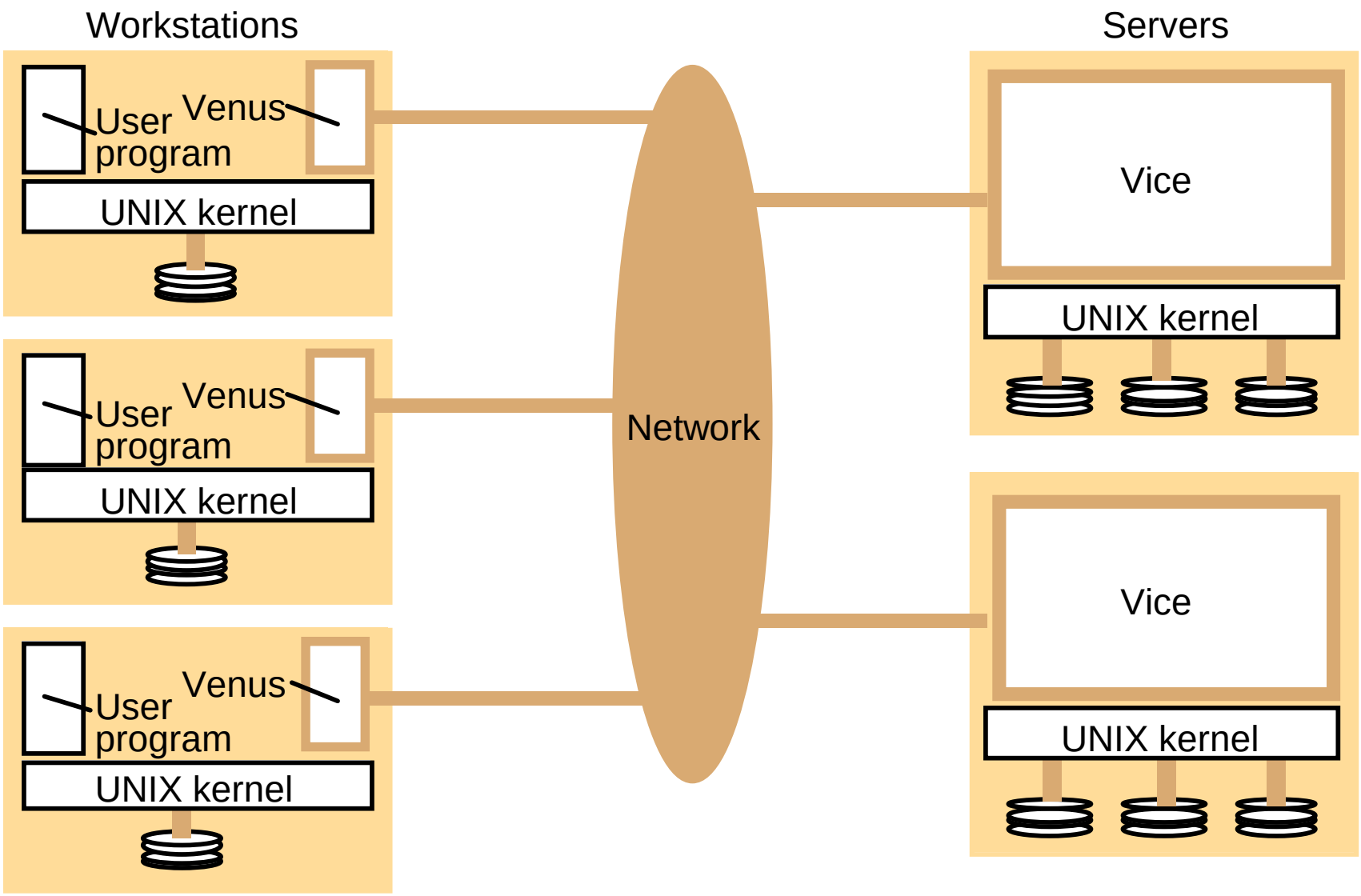
5 NFS - Ahuleziak

- Sendotasuna: *UNIX* semantika eskaintzea zaila da
- Gardentasuna kokapenean ez da automatikoa
- Elkarrekiko esklusioa lortzeko atzipenetan mekanismo aparte bat behar da
  - *UNIX*-en, *lockd* prozesua erabiltzen da
- Zerbitzariak ezin dira errepikatu
  - bakarrik atzipen guztiak irakurketak direnean
  - aldaketak eskuz kudeatu behar dira (NIS erabiliz)
- Eskalagarritasun mugatua
  - sare lokalentzako diseinatua

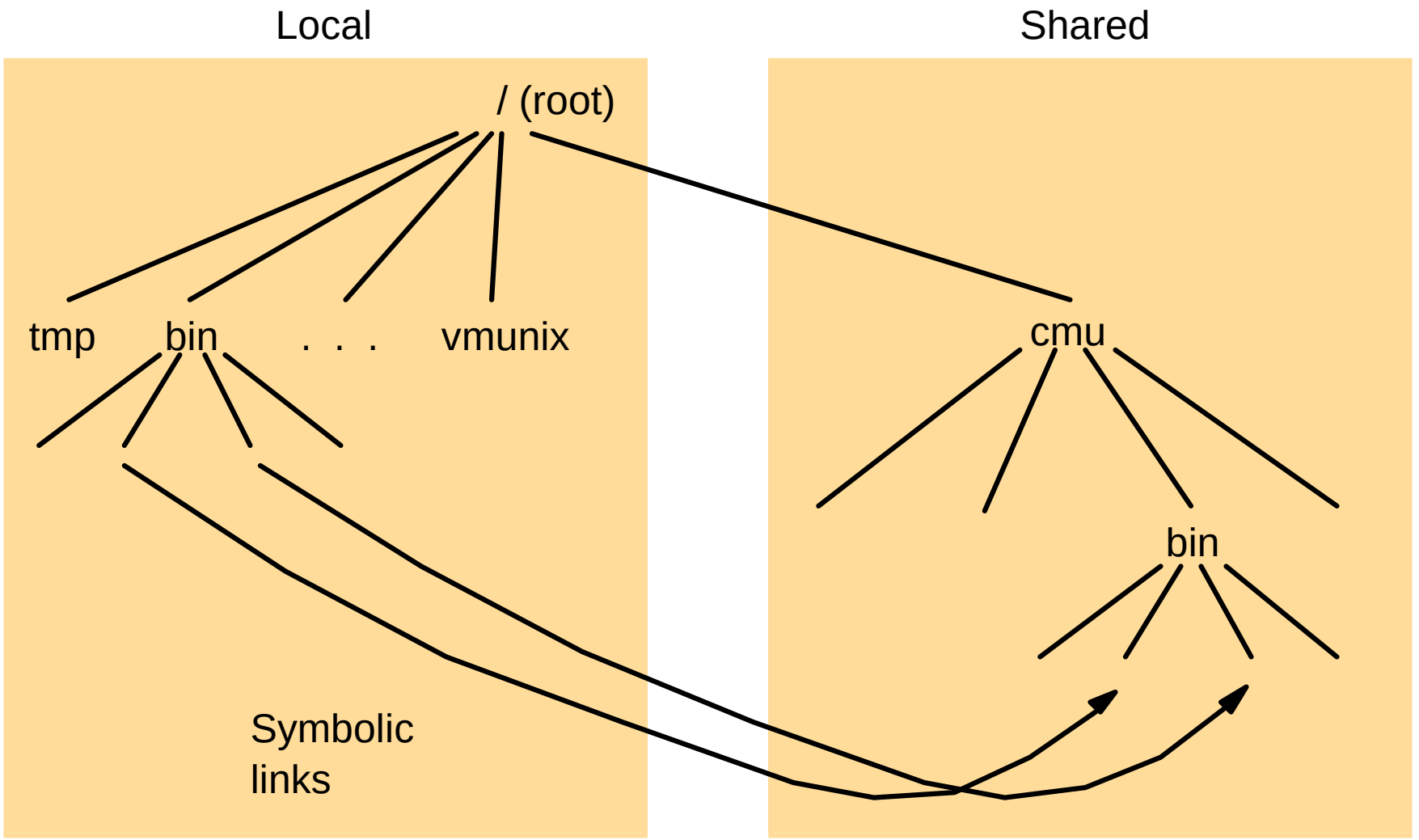
5 Andrew File System (AFS)

- Carnegie-Mellon University (CMU), 1986
- Supports information sharing on a large scale
    - 10000 workstations and servers running UNIX
- Access transparency: client programs use normal UNIX file primitives
- *Scalability* as the most important design goal:
  - *whole-file serving* by AFS servers to client computers
  - *whole-file caching* (on local disk) in client nodes
    - size of local cache: 100 MB
- Uses a session semantics (*write-on-close*)
    - assumption: concurrent updates are very rare

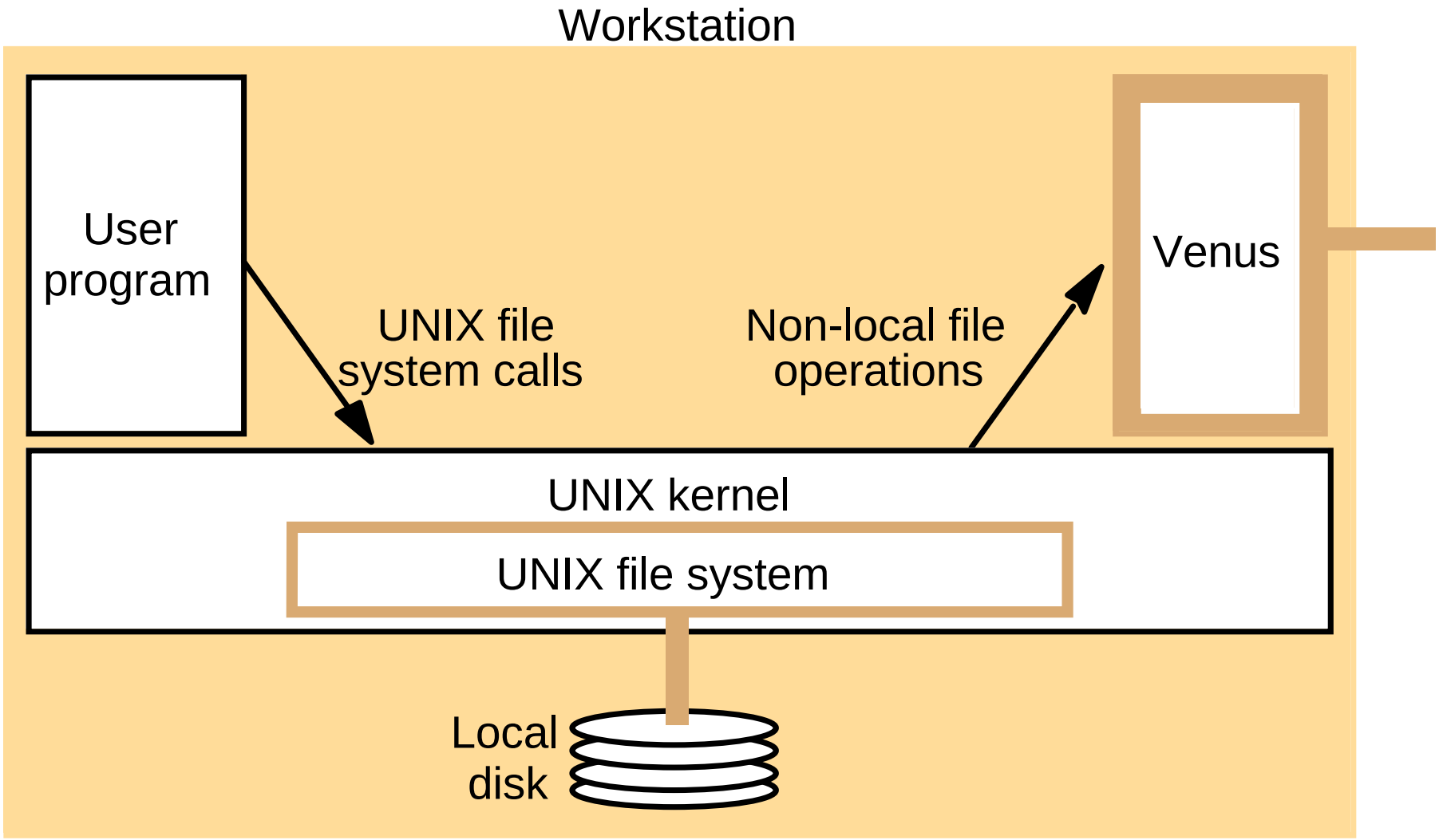
5 AFS - Arkitektura



5 AFS - Arkitektura (2)



5 AFS - Arkitektura (3)



5 AFS - Cache sendotasuna

- When Vice supplies a copy of a file to a Venus process it also provides a *callback promise*
- When a server performs a request to update a file it notifies all of the Venus processes to which it has issued callback promises by sending a *callback* to each
  - servers are not stateless!!!
  - the callback-based mechanism offers the most scalable approach
- On *open*: if Venus has a *valid* copy it can open it directly, otherwise it has to fetch a fresh copy from the Vice server
- When Venus is restarted after a failure or a shut-down, the validity of the files in the cache must be checked, since some callbacks may have been missed.

5 AFS - Operazioak

User process	*UNIX kernel*	*Venus*	*Net*	*Vice*
open(FileName, mode)	If *FileName* refers to a file in shared file space, pass the request to Venus.	Check list of files in local cache. If not present or there is no valid *callback promise*, send a request for the file to the Vice server that is custodian of the volume containing the file.	→	Transfer a copy of the file and a *callback promise* to the workstation. Log the callback promise.
	Open the local file and return the file descriptor to the application.	Place the copy of the file in the local file system, enter its local name in the local cache list and return the local name to UNIX.	←	
read(FileDescriptor, Buffer, length)	Perform a normal UNIX read operation on the local copy.			
write(FileDescriptor, Buffer, length)	Perform a normal UNIX write operation on the local copy.			
close(FileDescriptor)	Close the local copy and notify Venus that the file has been closed.	If the local copy has been changed, send a copy to the Vice server that is the custodian of the file.	→	Replace the file contents and send a *callback* to all other clients holding *callback promises* on the file.

5 Coda

- Descendant of AFS that is substantially more resilient to server and network failures
  - Carnegie-Mellon University (CMU), 1990
- Coda *(COnstant Data Availability)*: AFS + ...
  - replication (for load-balancing and fault-tolerance)
  - support for mobile users: *disconnected operation* mode
- Directories are replicated in several Vice servers
- When the Venus is disconnected, it uses local versions of files. When Venus reconnects, it reintegrates using an optimistic update scheme