U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

[dcc]

NETWORK FILE SYSTEM

ADMINISTRAÇÃO DE SISTEMAS

2021/2022

ROLANDO MARTINS

Referências dos slides

- O conteúdo destes slides é baseado no livro da disciplina: “Unix and Linux System Administration Handbook (4ªEd)” por Evi Nemeth, Garth Snyder, Trent R. Hein e Ben Whaley, Prentice Hall, ISBN: 0-13-148005-7
- As imagens usadas têm a atribuição aos autores ou são de uso livre.

Sistemas de ficheiros em rede

- Manutenção de estado
- Performance
- Segurança (controlo de acesso)

Manutenção de estado

- Servidor com estado (Stateful): mantem estado sobre todos os ficheiros que todos os clientes têm abertos no servidor.
  - **Pode levar a incoerência com crashes do servidor ou cliente.**
- Servidor sem estado (stateless): todos os pedidos são independentes uns dos outros. Não sabe quem tem que ficheiros abertos.
  - **Não gere a concorrência (o estado do ponto de vista dos clientes)**

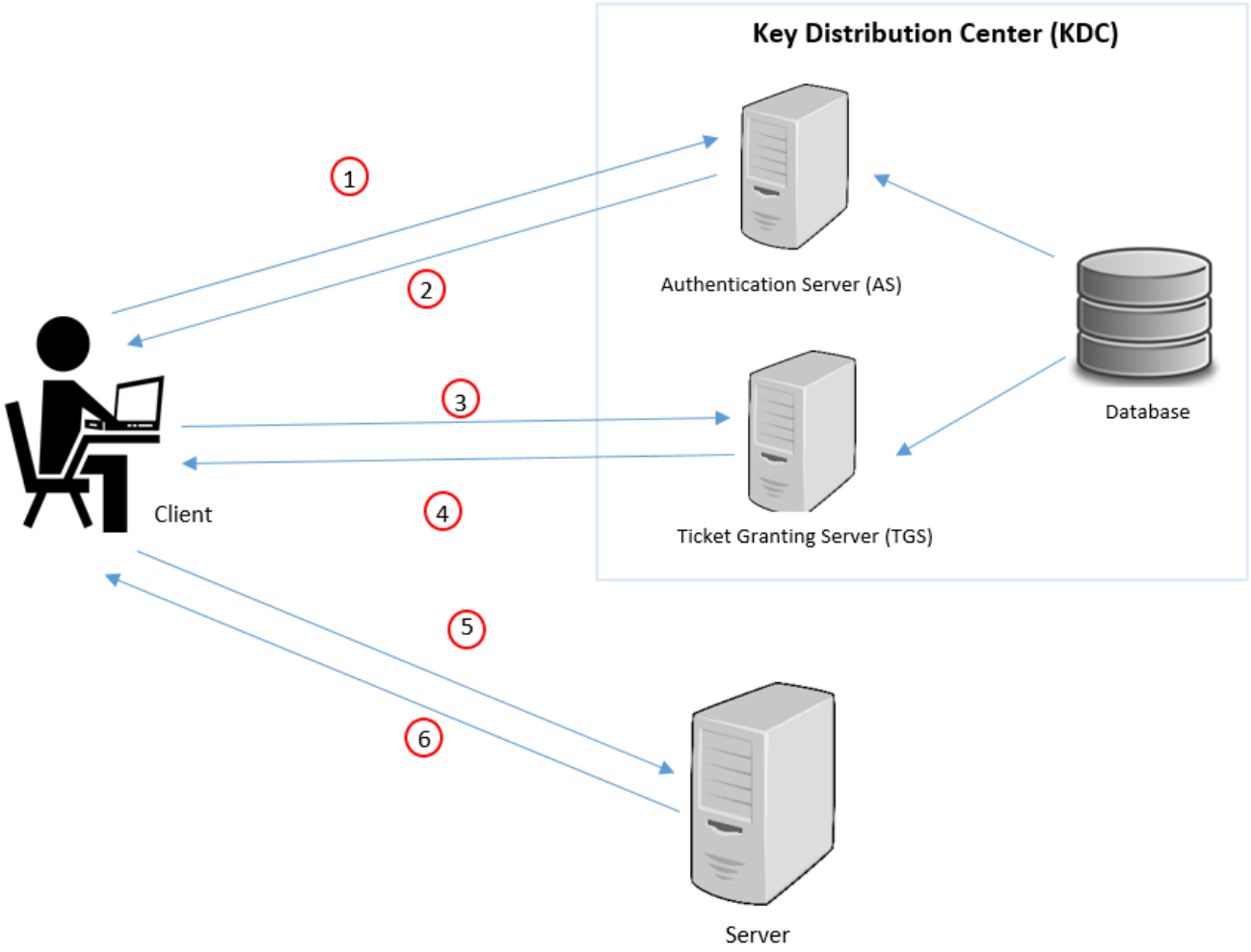
Performance

- Fazer cache de escritas no cliente
  - **Para as fazer em "série" (batch)**
- "Ler da rede" parte dos ficheiros abertos antes de ser necessário
  - **Buffer local**
- Objetivo: diminuir os pedidos à rede do conteúdo dos ficheiros
  - **➔ diminuir tempo de acesso**

Segurança

- Definição de uma forma de controlo de acesso centralizado
- Manter o controlo de acesso granular aos ficheiros
- Autenticação do acesso utilizando GSS-API [RFC2743]
  - Generic Security Service Application Program Interface
  - Suporte de Kerberos V5 obrigatório
- Acesso por *filehandles* opacas (podem ser adivinhadas)
- Dados podem ir cifrados (operação RPC READ e WRITE)
  - Mas não usualmente

Kerberos


Key Distribution Center (KDC)
Authentication Server (AS)
Database
Ticket Granting Server (TGS)
Client
Server
1
2
3
4
5
6

[dcc]

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

NFS

História

- Desenvolvido pela SUN Microsystems
- Atualmente aberto e sem licenças:
  - RFC 7530: Network File System (NFS) Version 4 Protocol
    - Com extensão na 4.2, RFC7862
  - Versão WebNFS (para versão NFS 2 e 3)
    - Ver Public File Handle no NFS 4.0, secção 16.21
    - RFC 2054: WebNFS Client Specification (para versão NFS 2 e 3)
    - RFC 2055: WebNFS Server Specification
- V2: (1ª pública) (RFC 1094)
  - Cliente necessitava de confirmação do servidor da escrita
  - Servidor apenas confirmava depois de escrever no disco
- V3: (RFC 1813, diferenças para v2)
  - Permite escrita assíncrona
  - Elevada melhoria de performance

NFS v4

- Compatibilidade e cooperação com firewalls e serviços NAT
- Integração das operações de lock e mount
- Operações com estado
- Segurança integrada (ver NFS3, RFC1813-8)
- Suporte para replicação e migração (RFC7530-9.14.4)
- Suporte de clientes Unix e Windows (refªs a Windows API RFC7530)
- ACL (listas de controlo de acesso) (RFC7530-6)
- Suporte de UNICODE para nomes de ficheiros
- Melhoria da performance

Retro-compatibilidade

- Não existe retro-compatibilidade com as versões anteriores
- Em geral servidores suportam separadamente as 3 versões

Características

- V4 usa TCP
  - V3 tinha opção entre UDP e TCP
- V4 mantem estado
  - **V2 e v3 eram sem estado**
    - Utilizam cookies para manter informação do lado do cliente
  - V4 mantem info sobre locks e ficheiros abertos
- Lista de diretórios exportados
  - `/etc/exports`
- Lock ficheiros:
  - V4: diretamente suportado
  - V2 e v3: utiliza o `statd` e `lockd`

V3 vs V4

https://archive.fosdem.org/2018/schedule/event/nfs3_to_nfs4/attachments/slides/2702/export/events/attachments/nfs3_to_nfs4/slides/2702/FOSDEM_Presentation_Final_pdf.pdf

Controlo acesso

- Segurança, controlo de acesso
  - `AUTH_NONE`: sem autenticação
  - `AUTH_SYS`: autenticação usa mapeamento do UID e GID enviado pelo cliente para ser mapeado num UID e GID do servidor (utilizando o `/etc/passwd`)
    - Não há garantias/validação dos dados enviados pelo cliente
  - `RPCSEC_GSS`: definição de autenticação para controlo de acesso, integridade e privacidade
  - Servidores V4 têm de implementar `RPCSEC_GSS`

https://pike.lysator.liu.se/docs/ietf/rfc/26/rfc2623.xml

Mapeamento de utilizadores

- Mapeia utilizadores do cliente em utilizadores do servidor (nome)
  - Servidor utiliza o `/etc/passwd` local para mapear `UID` e `GID` em `user@server`
  - Cliente mapeia `user` usando `/etc/passwd` local para um `UID`.
    - No caso de se ter de mostrar o nome do utilizador (`ls -l`), remapeia-se `UID` para nome
- Não é utilizado o nome para controlo de acesso
  - Para controlo de acesso são utilizados os `UID` e `GID` diretamente
- Usado por exemplo para `stat()` (ou o comando) e `chown`

Mapeamento em V4

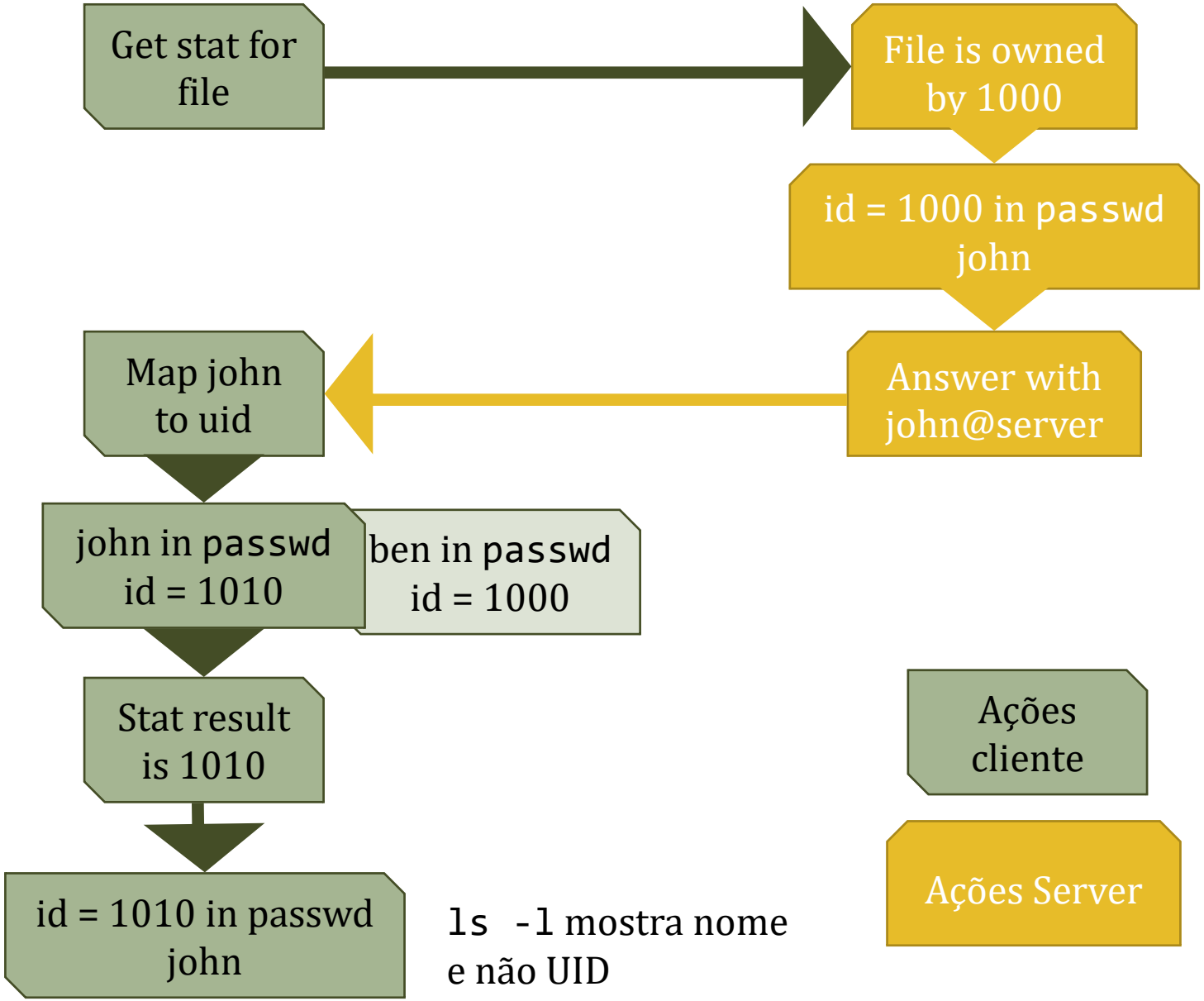
- Utiliza `rpc.idmapd`
- Identificadores
  - username@domínio-nfs
  - groupname@domínio-nfs
  - Mapeiam para UID e GID locais ao servidor

Exemplo (com cliente NFS 4)

No servidor UID 1000 é do john

```
[ben@nfs-client]$ id ben
uid=1000(ben) gid=1000(ben) groups=1000(ben)
[ben@nfs-client]$ id john
uid=1010(john) gid=1010(john) groups=1010(john)
[ben@nfs-client]$ ls -ld ben/
drwxr-xr-x 2 john root 4096 May 27 16:42 ben
[ben@nfs-client]$ touch ben/file
[ben@nfs-client]$ ls -l ben/file
-rw-rw-r-- 1 john nfsnobody 0 May 27 17:07 ben/file
```

Mapeamento UID



`ls -l` mostra nome e não UID

Apesar da indicação de id=1010 o controle de acesso não é baseado neste mapeamento

Mapeamento (cont.)

- Usado por chamadas à API do sistema de ficheiros
- Se usadas implicitamente (`touch` no exemplo) o sistema de autenticação/autorização é usado
  - Não usando o mapeamento, mas o `uid` no filehandle
  - Daí o `ben` poder criar o ficheiro
  - Daí o ficheiro ter o uid do `ben` no cliente (mapeado no `john` no servidor)

root squash

- Utilizador `root` é mapeado no utilizador `nobody`
- Existe a possibilidade de fazer o mesmo para todos os utilizadores:
  - `all_squash`
    - “The "**all_squash**" option maps all client requests to a single anonymous uid/gid on the **NFS** server, negating the ability to track file access by user ID”

Exemplo /etc/exports

```
/home harp(rw,no_root_squash) monk(rw)
/usr/share/man *.atrust.com(ro)
```

Ver mais: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/deployment_guide/s1-nfs-server-config-exports

Type	Syntax	Meaning
Hostname	hostname	Individual hosts
Netgroup	@groupname	NIS netgroups (not frequently used)
Wild cards	* and ?	FQDNs with wild cards; "*" will not match a dot
IP networks	ipaddr/mask	CIDR-style specifications (e.g., 128.138.92.128/25)

FQDN – Fully Qualified Domain Name

OPÇÕES

Option	Description
`ro`	Exports read-only
`rw`	Exports for reading and writing (the default)
`rw=list`	Exports read-mostly. The list enumerates the hosts allowed to mount for writing ; all others must mount read-only.
`root_squash`	Maps (“squashes”) UID 0 and GID 0 to the values specified by anonuid and anongid. This is the default.
`no_root_squash`	Allows normal access by root. Dangerous.
`all_squash`	Maps all UIDs and GIDs to their anonymous versions. Useful for supporting PCs and untrusted single-user hosts.
`anonuid=xxx`	Specifies the UID to which remote roots should be squashed
`anongid=xxx`	Specifies the GID to which remote roots should be squashed
`secure`	Requires remote access to originate at a privileged port
`insecure`	Allows remote access from any port
`noaccess`	Blocks access to this dir and subdirs (used with nested exports)
`wdelay`	**Delays writes in hopes of coalescing multiple updates**
`no_wdelay`	Writes data to disk as soon as possible
`async`	Makes server reply to write requests before actual disk write
`nohide`	Reveals filesystems mounted within exported file trees
`no_subtree_check`	Verifies only that file requests refer to an exported filesystem
`secure_locks`	Requires authorization for all lock requests
`insecure_locks`	Specifies less stringent locking criteria (supports older clients)
`sec=flavor`	Specifies a list of security methods for the exported directory. Values include sys (UNIX authentication), dh (DES), krb5 (Kerberos authentication), krb5i (Kerberos authentication and integrity), krb5p (Kerberos authentication, integrity, and privacy), and none (anonymous access, not recommended).

Servidor – `nfsd`

- Daemon que recebe os comandos
  - Emite comandos para a parte NFS embebida no Kernel
- Deve-se estipular o número de threads `nfsd` que devem correr
  - Deve ser verificado manualmente qual o melhor número

cliente

- Ver exports disponíveis (apenas se servidor suportar v3)

```
$ showmount -e server
```

- Usar o mount para montar os diretórios exportados

```
# mount -t nfs4 -o rw,hard,bg server:/exportdir/auser \
      /home/auser
```

- (ver man nfs)

Opções mount NFScliente

Flag	Description
`rw`	Mounts the filesystem read-write (must be exported that way)
`ro`	Mounts the filesystem read-only
`bg`	If the mount fails (server doesn't respond), keeps trying it in the background and continues with other mount requests
`hard`	If a server goes down, causes operations that try to access it to block until the server comes back up
`soft`	If a server goes down, causes operations that try to access it to fail and return an error, thereby avoiding processes "hanging" on inessential mounts
~~`intr`~~	~~Allows users to interrupt blocked operations (and return an error)~~
~~`nointr`~~	~~Does not allow user interrupts~~
`retrans=n`	Specifies the number of times to repeat a request before returning an error on a soft-mounted filesystem
`timeo=n`	Sets the timeout period (in 10ths of a second) for requests
`rsize=n`	Sets the read buffer size to n bytes
`wsize=n`	Sets the write buffer size to n bytes
`sec=flavor`	Specifies the security flavor
`vers=na`	Sets the NFS protocol version
`proto=proto`	Selects a transport protocol; must be tcp for NFS version 4

Outros

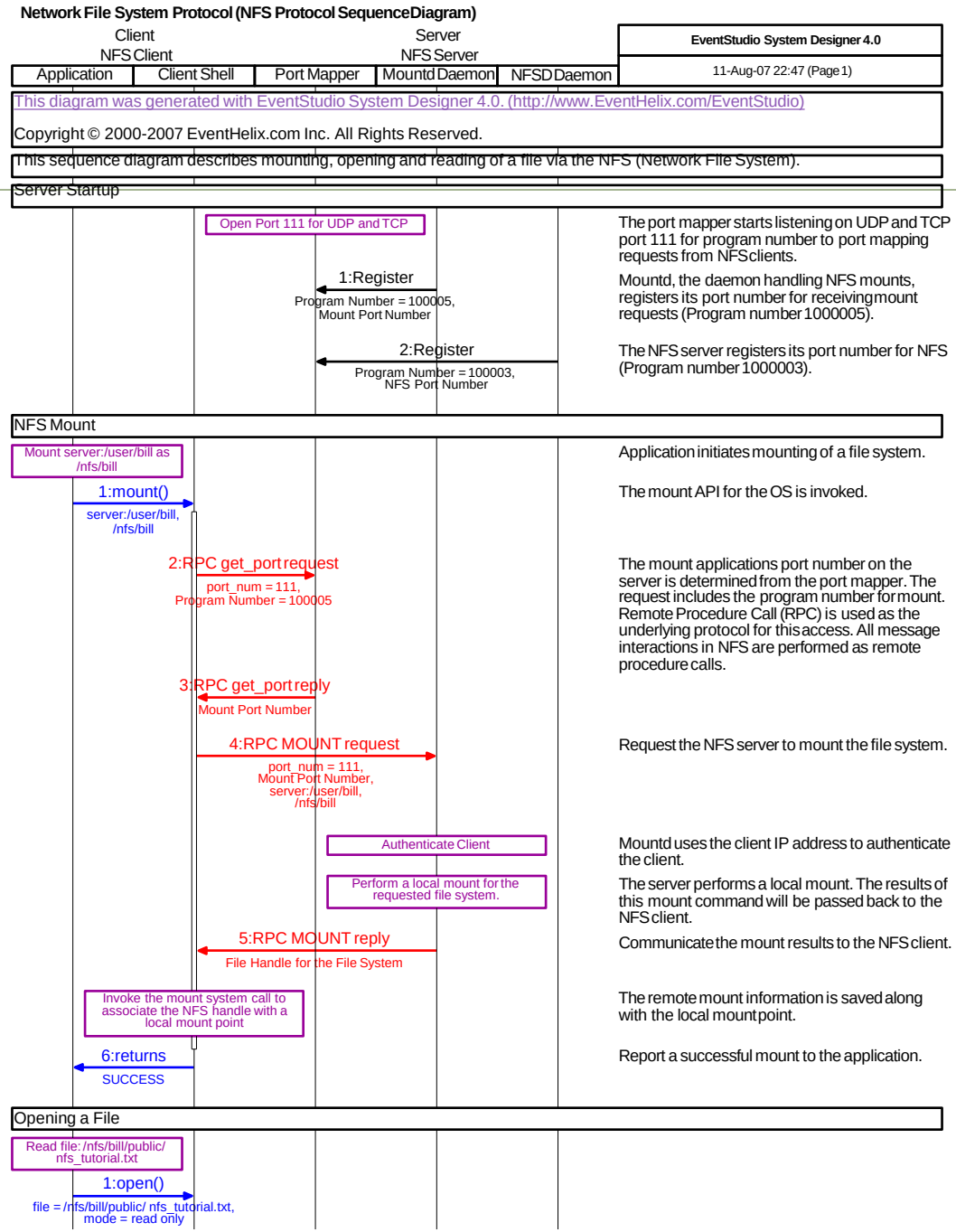
`nfsstat`

- `-c`: estatísticas do lado do cliente
- `-s`: lado servidor

`automount`

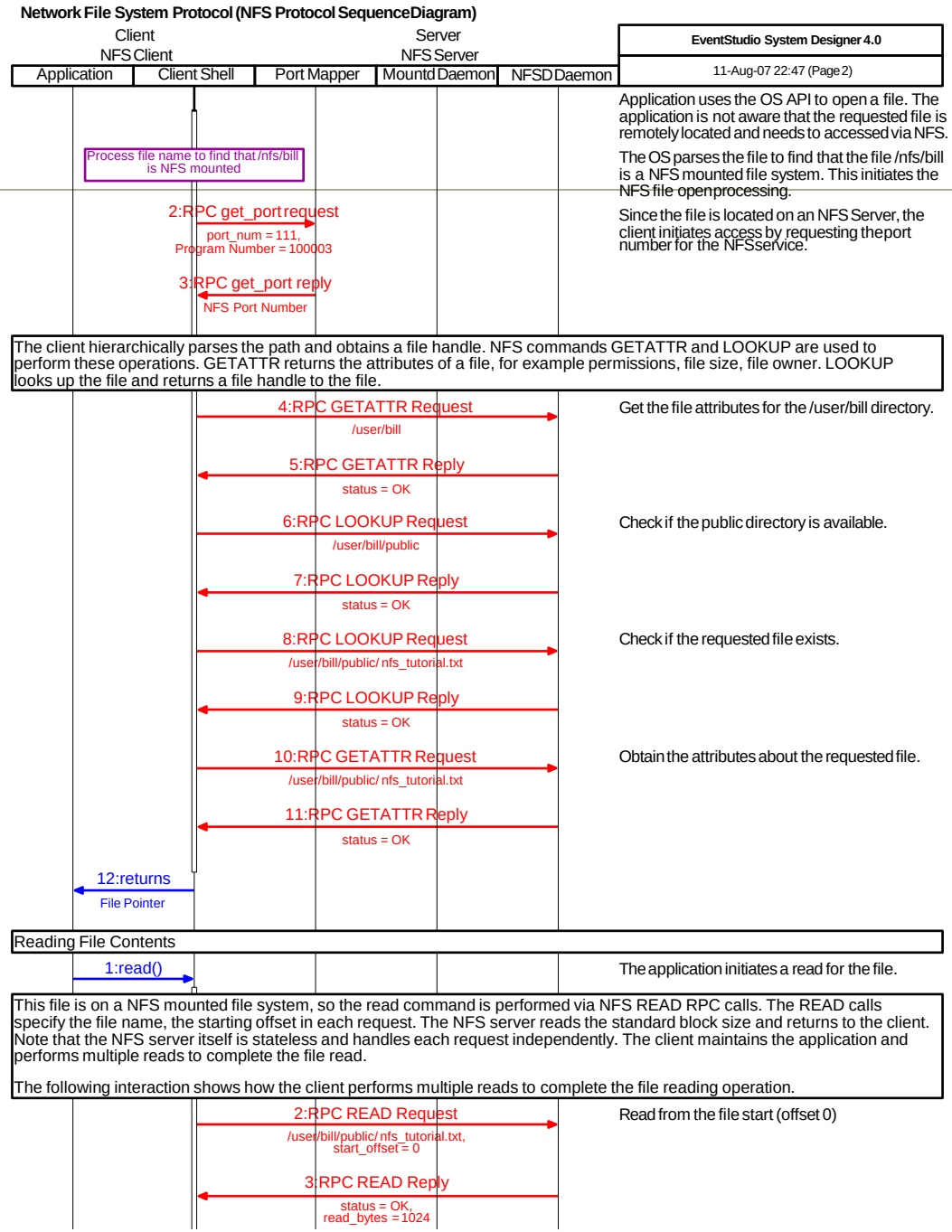
- Auto montar sistemas

  `/etc/auto.master`


Network File System Protocol (NFS Protocol SequenceDiagram)
Client
NFS Client
Server
NFS Server
EventStudio System Designer 4.0
11-Aug-07 22:47 (Page 1)
Application
Client Shell
Port Mapper
Mountd Daemon
NFSD Daemon
This diagram was generated with EventStudio System Designer 4.0. (http://www.EventHelix.com/EventStudio)
Copyright © 2000-2007 EventHelix.com Inc. All Rights Reserved.
This sequence diagram describes mounting, opening and reading of a file via the NFS (Network File System).
Server Startup
Open Port 111 for UDP and TCP
The port mapper starts listening on UDP and TCP port 111 for program number to port mapping requests from NFS clients.
1:Register
Program Number = 100005, Mount Port Number
Mountd, the daemon handling NFS mounts, registers its port number for receiving mount requests (Program number 1000005).
2:Register
Program Number = 100003, NFS Port Number
The NFS server registers its port number for NFS (Program number 1000003).
NFS Mount
Mount server:/user/bill as /nfs/bill
Application initiates mounting of a file system.
1:mount()
server:/user/bill, /nfs/bill
The mount API for the OS is invoked.
2:RPC get_port request
port_num = 111, Program Number = 100005
The mount applications port number on the server is determined from the port mapper. The request includes the program number for mount. Remote Procedure Call (RPC) is used as the underlying protocol for this access. All message interactions in NFS are performed as remote procedure calls.
3:RPC get_port reply
Mount Port Number
4:RPC MOUNT request
port_num = 111, Mount Port Number, server:/user/bill, /nfs/bill
Request the NFS server to mount the file system.
Authenticate Client
Mountd uses the client IP address to authenticate the client.
Perform a local mount for the requested file system.
The server performs a local mount. The results of this mount command will be passed back to the NFS client.
5:RPC MOUNT reply
File Handle for the File System
Communicate the mount results to the NFS client.
Invoke the mount system call to associate the NFS handle with a local mount point
The remote mount information is saved along with the local mount point.
6:returns
SUCCESS
Report a successful mount to the application.
Opening a File
Read file: /nfs/bill/public/ nfs_tutorial.txt
1:open()
file = /nfs/bill/public/ nfs_tutorial.txt, mode = read only

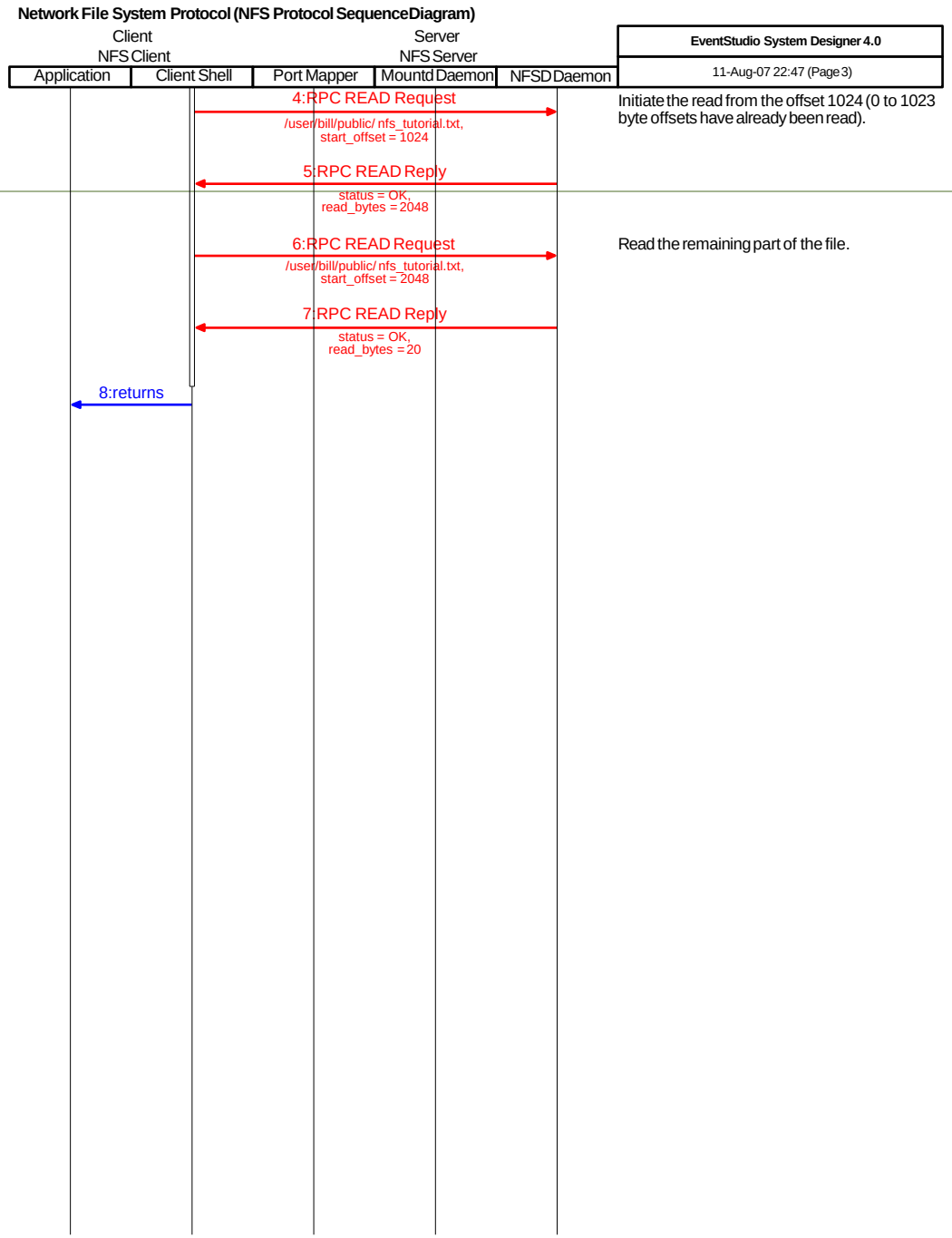
[dcc]

U.PORTO


Network File System Protocol (NFS Protocol SequenceDiagram)
Client
NFS Client
Server
NFS Server
Application
Client Shell
Port Mapper
Mountd Daemon
NFSD Daemon
EventStudio System Designer 4.0
11-Aug-07 22:47 (Page 2)
Application uses the OS API to open a file. The application is not aware that the requested file is remotely located and needs to accessed via NFS.
Process file name to find that /nfs/bill is NFS mounted
The OS parses the file to find that the file /nfs/bill is a NFS mounted file system. This initiates the NFS file openprocessing.
2:RPC get_port request
port_num = 111,
Program Number = 100003
Since the file is located on an NFS Server, the client initiates access by requesting theport number for the NFSservice.
3:RPC get_port reply
NFS Port Number
The client hierarchically parses the path and obtains a file handle. NFS commands GETATTR and LOOKUP are used to perform these operations. GETATTR returns the attributes of a file, for example permissions, file size, file owner. LOOKUP looks up the file and returns a file handle to the file.
4:RPC GETATTR Request
/user/bill
Get the file attributes for the /user/bill directory.
5:RPC GETATTR Reply
status = OK
6:RPC LOOKUP Request
/user/bill/public
Check if the public directory is available.
7:RPC LOOKUP Reply
status = OK
8:RPC LOOKUP Request
/user/bill/public/ nfs_tutorial.txt
Check if the requested file exists.
9:RPC LOOKUP Reply
status = OK
10:RPC GETATTR Request
/user/bill/public/ nfs_tutorial.txt
Obtain the attributes about the requested file.
11:RPC GETATTR Reply
status = OK
12:returns
File Pointer
Reading File Contents
1:read()
The application initiates a read for the file.
This file is on a NFS mounted file system, so the read command is performed via NFS READ RPC calls. The READ calls specify the file name, the starting offset in each request. The NFS server reads the standard block size and returns to the client. Note that the NFS server itself is stateless and handles each request independently. The client maintains the application and performs multiple reads to complete the file read.
The following interaction shows how the client performs multiple reads to complete the file reading operation.
2:RPC READ Request
/user/bill/public/ nfs_tutorial.txt,
start_offset = 0
Read from the file start (offset 0)
3:RPC READ Reply
status = OK,
read_bytes = 1024

[dcc]

U.PORTO


Network File System Protocol (NFS Protocol SequenceDiagram)
Client
NFS Client
Application
Client Shell
Server
NFS Server
Port Mapper
Mountd Daemon
NFSD Daemon
EventStudio System Designer 4.0
11-Aug-07 22:47 (Page 3)
4:RPC READ Request
/user/bill/public/ nfs_tutorial.txt,
start_offset = 1024
Initiate the read from the offset 1024 (0 to 1023 byte offsets have already been read).
5:RPC READ Reply
status = OK,
read_bytes = 2048
6:RPC READ Request
/user/bill/public/ nfs_tutorial.txt,
start_offset = 2048
Read the remaining part of the file.
7:RPC READ Reply
status = OK,
read_bytes = 20
8:returns

Resumo

- Estado, performance, segurança
- NFS características
- Controlo de acesso
- Configuração

[dcc]

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

QUESTÕES/ COMENTÁRIOS