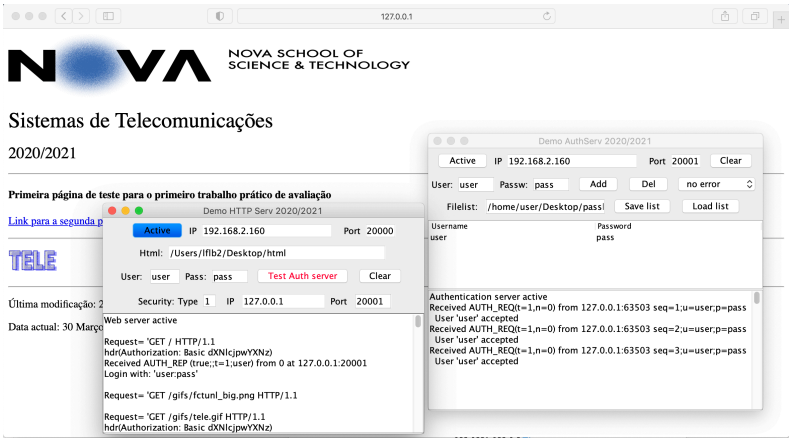
NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING



Sistemas de Telecomunicações

2020/2021

Trabalho 4:

Aplicação sobre sockets – serviço web com autenticação
PARTE 1 – Serviço de autenticação

Aula 5

Mestrado integrado em Engenharia Eletrotécnica e de Computadores

http://tele1.dee.fct.unl.pt **V1.0**

Luis Bernardo
Paulo da Fonseca Pinto

Índice

1. OBJETIVO

Familiarização pelo aluno e verificação pelo docente da capacidade de usar *sockets* para comunicação entre máquinas e trabalhar com a interface de programação *sockets* do Java.

O trabalho consiste no desenvolvimento de

- um servidor de autenticação e de
- um servidor web

para criar um sistema onde o acesso às páginas web requer o conhecimento de uma palavra de passe e, portanto, só pode ser feito por um utilizador registado no serviço de autenticação.

O servidor web comunica com o servidor de autenticação usando *sockets datagrama* de forma a validar as palavras de passe entretanto recebidas dos utilizadores. Para além disso, guarda numa lista local (*cache*) as palavras de passe durante um intervalo de tempo, podendo usá-las para validar utilizadores sem ter de voltar a perguntar ao servidor de autenticação. Os utilizadores usam browsers genéricos que comunicam com o servidor web pelo protocolo *HyperText Transfer Protocol* (HTTP).

O trabalho deve ser feito nas aulas de laboratório, sendo dividido em duas aulas: trabalho 4 na aula 5 e trabalho 5 na aula 6. Durante a aula 6, já presencial, o docente vai avaliar o trabalho 4 e no final dessa aula avaliar o trabalho 5.

2. ESPECIFICAÇÕES

Pretende-se desenvolver um serviço web com controlo de acesso por nome de utilizador e palavra de passe, geridos por um serviço externo (*Authentication*). O servidor web (entidade do meio na figura em baixo) tem um endereço IP e porto conhecidos publicamente, e conhece o endereço IP e porto do servidor de autenticação (à direita na figura em baixo). O utilizador usa um *browser* (à esquerda na figura em baixo) para aceder a páginas no servidor web.

A comunicação entre o cliente (*browser web*) e o servidor web usa o protocolo de aplicação *HyperText Transfer Protocol* (HTTP) sobre um *socket* orientado à ligação. A comunicação entre o(s) servidor(es) web e de autenticação é realizada através da troca de pacotes datagrama com pedidos e respostas para validação das palavras de passe.

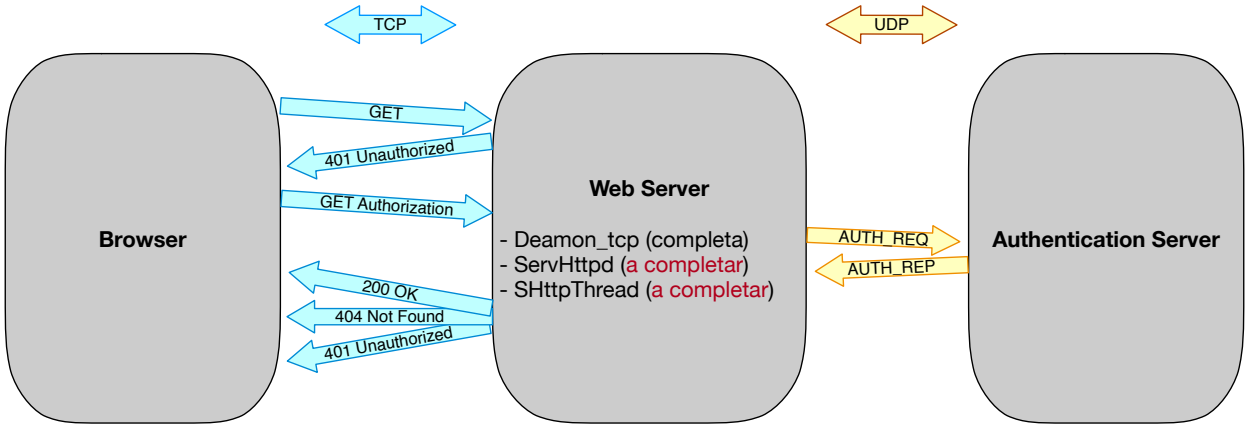
Podem haver vários servidores de autenticação.

O servidor web responde a pedidos para transferência de ficheiros locais, validando a palavra de passe em cada acesso. Também permite ao utilizador configurar qual é o servidor de autenticação usado na validação. Para os dois servidores é possível definir quais são os portos TCP e UDP usados respetivamente por cada um.

Os clientes acedem ao serviço utilizando um *browser* web genérico (Firefox, Google Chrome, Internet Explorer, Microsoft Edge, Safari, etc.).

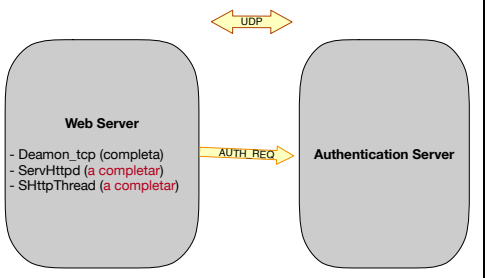
Na figura em baixo esquematiza-se a troca de informação TCP e UDP entre as aplicações. O *browser* envia um pedido de um ficheiro/página, através da mensagem "GET". No caso de esta mensagem não incluir um campo "Authorization" o servidor devolve uma mensagem "401 Unauthorized" que força o browser a pedir um nome de utilizador e uma palavra de passe. Caso o campo seja recebido, é enviada uma mensagem UDP com um pedido AUTH_REQ (4) para o servidor de autenticação, que responde com uma mensagem AUTH_REP (5) a aceitar ou não a palavra de passe. O servidor web devolve ao browser uma mensagem 401, 404 ou 200, respetivamente quando a palavra de passe não é válida, quando o ficheiro não existe, ou em caso de sucesso. O ficheiro pedido apenas é enviado com a mensagem "200 OK".

Durante a primeira aula vai ser realizada a comunicação entre o servidor web e o servidor de autenticação.



2.1. COMUNICAÇÃO ENTRE SERVIDOR WEB E SERVIDOR DE AUTENTICAÇÃO

O pedido de validação de palavras de passe é feito enviando uma mensagem AUTH_REQ para o servidor de autenticação e recebendo uma mensagem de resposta AUTH_REP que vão ter um de vários tipos possíveis, e que têm uma estrutura diferente para cada tipo. A mensagem AUTH_REQ tem o seguinte conteúdo:

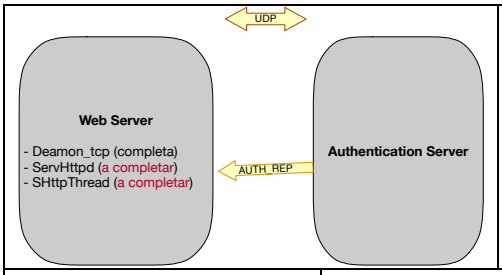



```
Message AUTH_REQ:       // Ordered sequence of
 byte type;             // Message type
 ...
followed by a sequence of fields with an order
that depends of the type value, which includes:
  int seq;          // Request sequence number
  int number;       // number of student
  short len_user;   // Length of the username string
  byte[] user;      // username string
  short len_pass;   // Length of the password string
  byte[] pass;      // password string
```

type == 1	*type == 2*	*type == 3*	*type == 4*
byte **type**=1; int **seq**; int ***number***; short **len_user**; byte[] **user**; short **len_pass**; byte[] **pass**;	byte **type**=2; int **seq**; short **len_user**; byte[] **user**; short **len_pass**; byte[] **pass**; int ***number***;	byte **type**=3; int ***number***; short **len_user**; byte[] **user**; short **len_pass**; byte[] **pass**; int **seq**;	byte **type**=4; short **len_user**; byte[] **user**; short **len_pass**; byte[] **pass**; int ***number***; int **seq**;
type == 5	*type == 6*	*type == 7*	*type == 8*
byte **type**=5; int **seq**; int ***number***; short **len_pass**; byte[] **pass**; short **len_user**; byte[] **user**;	byte **type**=6; int **seq**; short **len_pass**; byte[] **pass**; short **len_user**; byte[] **user**; int ***number***;	byte **type**=7; int ***number***; short **len_pass**; byte[] **pass**; short **len_user**; byte[] **user**; int **seq**;	byte **type**=8; short **len_pass**; byte[] **pass**; short **len_user**; byte[] **user**; int ***number***; int **seq**;

Cada pedido inclui duas strings com a autenticação recebida do browser: `user` é o nome de utilizador e `pass` é a palavra de passe; antes de transmitir cada string é transmitido o seu comprimento (respetivamente, `len_user` e `len_pass`). Também inclui o número do estudante que realizou o trabalho (`number`), um número de sequência de pedido (`seq`) e o tipo de mensagem AUTH_REQ enviado, que também deve ser usado na mensagem de resposta (`type`).

O servidor de autenticação valida o valor de `user` e `pass`, e devolve em AUTH_REP o resultado (`accepted`) `true` ou `false`, o número do estudante que realizou o servidor de autenticação (`number`), com um código de sequência que está relacionado com o número de sequência da mensagem AUTH_REQ e com o nome do utilizador que recebeu em AUTH_REQ. Para evitar ficar bloqueado, o servidor web deve definir um tempo máximo de espera pela mensagem AUTH_REP que varia conforme o aluno.




```
Message AUTH_REP:          // Ordered sequence of
 byte type;    // Message type
 ...
followed by a sequence of fields with an order
that depends of the type value, which includes:
 int number;        // Number of student
 short len_user;    // Length of username
 byte[] user;       // username received in AUTH_REQ
 int seqcheck;      // Sequence number check number
 boolean accepted; // true if valid, false otherwise
```

type == 1	*type == 2*	*type == 3*	*type == 4*
byte **type**=1; int ***number***; int **seqcheck**; short **len_user**; byte[] **user**; boolean **accepted**;	byte **type**=2; int **seqcheck**; int ***number***; short **len_user**; byte[] **user**; boolean **accepted**;	byte **type**=3; boolean **accepted**; int **seqcheck**; short **len_user**; byte[] **user**; int ***number***;	byte **type**=4; int **seqcheck**; boolean **accepted**; short **len_user**; byte[] **user**; int ***number***;
type == 5	*type == 6*	*type == 7*	*type == 8*
byte **type**=5; int ***number***; short **len_user**; byte[] **user**; int **seqcheck**; boolean **accepted**;	byte **type**=6; int **seqcheck**; short **len_user**; byte[] **user**; int ***number***; boolean **accepted**;	byte **type**=7; boolean **accepted**; short **len_user**; byte[] **user**; int ***number***; int **seqcheck**;	byte **type**=8; int **seqcheck**; boolean **accepted**; int ***number***; short **len_user**; byte[] **user**;

O tipo de mensagem (*type*) a usar no trabalho depende do número de protocolo (parâmetro P) que lhe vai ser fornecido pelo professor no início da aula de laboratório. O tempo de espera pela mensagem AUTH_REP (*WTime*) também varia com o tipo número de protocolo, conforme a tabela abaixo.

Espera/Tipo	*type*= 1	*type*= 2	*type*= 3	*type*= 4	*type*= 5	*type*= 6	*type*= 7	*type*= 8
WTime= 2s	P=1	P=2	P=3	P=4	P=5	P=6	P=7	P=8
WTime= 4s	P=9	P=10	P=11	P=12	P=13	P=14	P=15	P=16
WTime= 6s	P=17	P=18	P=19	P=20	P=21	P=22	P=23	P=24

O valor de *seqcheck* varia com o tipo de mensagem enviado e é calculado somando ao valor de *seq* recebido na mensagem AUTH_REQ o valor de *n* indicado na tabela abaixo.

seqcheck/Tipo	*type*= 1	*type*= 2	*type*= 3	*type*= 4	*type*= 5	*type*= 6	*type*= 7	*type*= 8
seqcheck=seq+n	*n*=8	*n*=7	*n*=6	*n*=5	*n*=4	*n*=3	*n*=2	*n*=1

NO INÍCIO DA AULA VAI RECEBER O NÚMERO DE PROTOCOLO A IMPLEMENTAR. Caso não programe o protocolo com os parâmetros corretos tem ZERO valores no exercício.

3. DESENVOLVIMENTO DO PROGRAMA

O programa está estruturado em duas aplicações: o servidor web é baseado no código de um servidor HTTP simplificado apresentado no documento tutorial ("Introdução ao desenvolvimento de aplicações que funcionam em rede usando a linguagem Java"), que já apresenta os mecanismos de leitura dos comandos mais importantes do protocolo HTTP.

O servidor de autenticação é fornecido completo sob a forma de um binário Java (ficheiro demoAuthServer.jar).

O trabalho consiste em completar o código do servidor web fornecido, seguindo as instruções apresentadas neste documento.

3.1. SERVIDOR DE AUTENTICAÇÃO

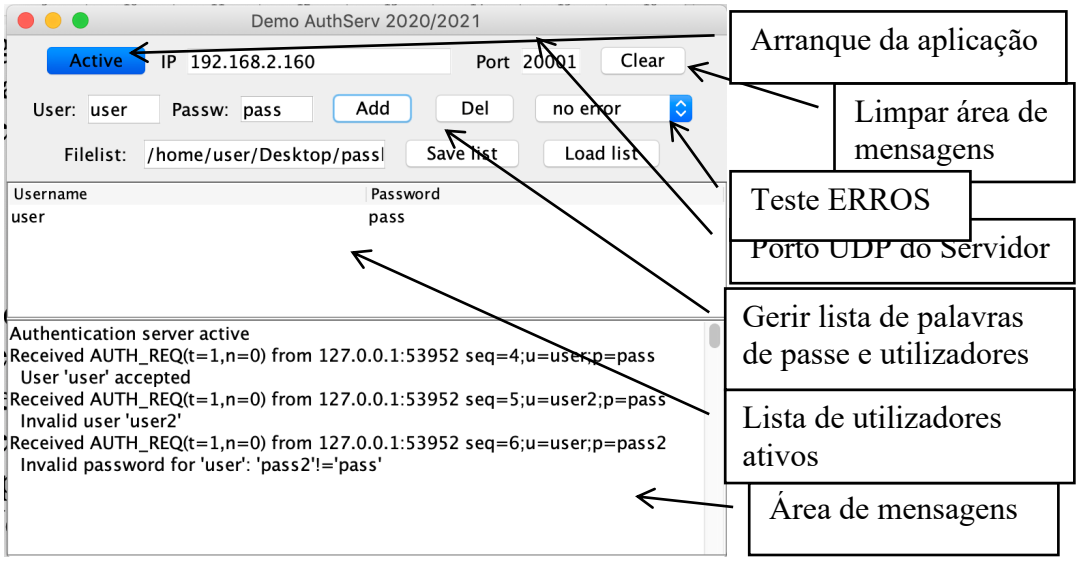
O servidor de autenticação gere uma lista em que cada entrada tem um nome de utilizador e a sua palavra de passe. A interface gráfica do servidor de autenticação apresenta uma tabela com a lista tendo botões para acrescentar (*Add*) e remover (*Del*) utilizadores e para a gravar no (*Save list*) ou a ler do (*Load list*) ficheiro indicado em (*Filelist:*).

Permite também especificar o porto UDP local (*UDP Port*), sendo o *socket* UDP criado quando o botão *Active* é ligado. Nessa altura, está preparado para responder a pedidos de autenticação.

O programa é fornecido na forma de um ficheiro binário, que pode ser corrido no computador local.

Para permitir testar o servidor web que vai desenvolver, o servidor de autenticação pode ser configurado para quatro modos de funcionamento:

- *no error* – cumpre o protocolo especificado;
- *bad type* – responde com o tipo de mensagem errado;
- *bad user* – responde com o nome de utilizador errado;
- *bad seqcheck* – responde com o valor errado para o campo *seqcheck*.



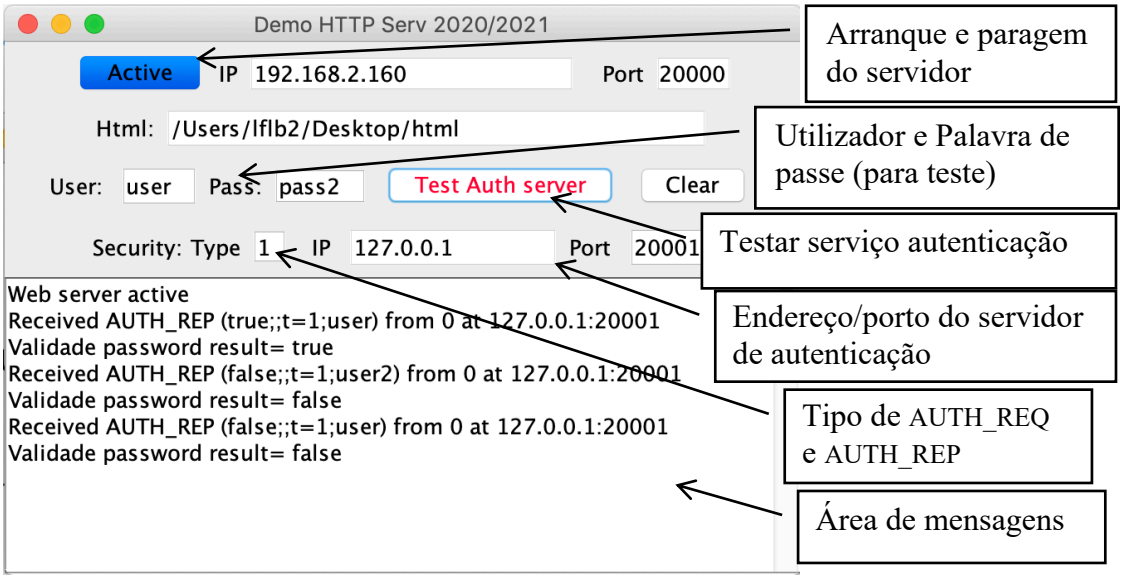
3.2. SERVIDOR WEB – PRIMEIRA PARTE

A função principal do servidor web é responder a pedidos de um browser, enviando o conteúdo de ficheiros no caso do utilizador ser válido.

O código que foi disponibilizado aceita o pedido do browser e assume (erradamente) que o ficheiro existe sempre devolvendo o seu conteúdo. Terá de ser modificado no trabalho 5.

Esse código terá de chamar um método para autenticar o par utilizador:palavrapasse. Como esta parte da autenticação (sem a cache) vai ser feita neste trabalho 4, o código disponibilizado chama esse método com os dados que forem introduzidos na parte gráfica (ver figura em baixo e ver também o código). O pedido é feito carregando no botão *Test Auth server*. O servidor web deve estar ativo.

A parte gráfica contém também campos para se especificar o endereço IP e porto do servidor de autenticação (*Security: IP* and *Port*), o tipo de AUTH_REP pretendido (*Security: Type*), o nome de utilizador (*User:*) e a palavra de passe (*Pass:*).



O programa fornecido é composto por três classes do pacote *server*:

- *Daemon_tcp.java* (completa) – *Thread* que recebe ligações no *ServerSocket*. Tem a ver com *sockets* TCP. Foi introduzida no trabalho 3;
- *SHttpThread.java* (a completar no trabalho 5) – *Thread* que processa os pedidos HTTP e envia a resposta. As mesmas considerações da classe anterior;
- *ServHttpd.java* (**a completar parcialmente neste trabalho**) – Classe principal com interface gráfica, que realiza a comunicação com o servidor de autenticação.

Faltam fazer três tarefas, como explicado abaixo: o envio da mensagem AUTH_REQ; a leitura e validação da mensagem AUTH_REP; e programar o *socket* UDP *ds* para esperar um tempo máximo pela mensagem AUTH_REP.

Comece por modificar a constante *SERVER_NAME*, substituindo o 00000 pelo seu número de estudante:

```
public final static String SERVER_NAME = "HTTP Serv by 00000";
```

As tarefas a realizar na classe *ServHttpd* são:

1. Na função *jToggleButton1ActionPerformed*, após criar o socket UDP, deve configurar o socket *ds* para ter um tempo máximo de espera por mensagens AUTH_REP de acordo com o número de protocolo que lhe foi atribuído. No caso de expirar o tempo a validação considera-se falhada, e tem de considerar nula qualquer resposta que venha atrasada no sítio apropriado.

Sugestão: Leia a secção 5.2.3 do documento tutorial, sobre a classe DatagramSocket, para saber como se define um tempo máximo de espera por pacotes.

2. Complete a função *send_AUTH_REQ* de forma a preparar uma mensagem AUTH_REQ de acordo com o protocolo que lhe foi atribuído.

```
private boolean send_AUTH_REQ(byte type, InetAddress servIP, int servPort, int number,
                              String user, String pass, int seq);
```

3. Programar a função *validate_password* de forma a descodificar os campos da mensagem AUTH_REP a partir do objeto *dis*, retornando o valor do campo *accepted*. Deve validar se a mensagem recebida é válida (deve ter o mesmo nome de utilizador que a mensagem AUTH_REQ enviada e o valor de *seqcheck* de acordo com o protocolo que lhe foi atribuído). Para esta aula não considere a existência da cache local;

```
public boolean validate_passwd(String url, String auth);
```

Sugestão: Analise como pode usar a função *read_String(DataInputString dis)*, da classe ServHttpd, para facilitar a leitura de strings.

Os **pesos na nota final do trabalho das três tarefas** são respetivamente: **20%**, **40%** e **40%**.

Teste o código com o servidor de autenticação e para os vários tipos de erros.

3.3. AVALIAÇÃO DO TRABALHO

A avaliação deste trabalho é realizada durante a aula em que os alunos estão a fazer o trabalho 5. Os alunos devem:

1. No início da aula, pedir o número de protocolo (P) ao docente;
2. No final de cada aula, preparar um ficheiro comprimido com o código desenvolvido para o projeto, com o nome, *00000-PXX.zip* (ou *.tgz*), onde 00000 deve ser substituído pelo número de aluno, e PXX ter o número do turno e deve entregar esse ficheiro ao docente por email;
3. Quando o docente lhe pedir, deve mostrar o código a funcionar, e estar preparado para responder a eventuais perguntas que possam ser feitas.

Nos casos onde existam dúvidas sobre a autoria do trabalho realizado pode ser marcada uma discussão posterior.

Caso sejam detetados erros na realização, pode haver uma valorização parcial de cada elemento de avaliação, através da análise do código entregue.

Caso o trabalho não seja entregue no final da aula, pode ser ainda entregue no início da aula seguinte (a da avaliação deste trabalho) com uma penalização de 50% (i.e. a nota máxima passa a ser de 10 valores com tudo feito corretamente).

POSTURA DOS ALUNOS

Cada aluno deve ter em consideração o seguinte:

- Não perca tempo com a estética de entrada e saída de dados
- Programe de acordo com os princípios gerais de uma boa codificação (utilização de indentação, apresentação de comentários, uso de variáveis com nomes conformes às suas funções...)