Chamada para documentação de Exercícios Práticos em vídeo usando o testbed FIBRE

O testbed FIBRE[1] constitui um recurso importante para a comunidade acadêmica brasileira. A motivação inicial para a sua construção foi a necessidade de uma infraestrutura para pesquisa na área de redes de computadores e internet do futuro. No entanto, a infraestrutura resultante também pode apoiar a formação de recursos humanos na área de redes de computadores, permitindo tanto a familiarização com novas tecnologias como também o domínio de conceitos e tecnologias em utilização hoje e dos problemas envolvidos. O testbed é muito relevante como um laboratório remoto disponível para disciplinas de redes de computadores e sistemas distribuídos em qualquer universidade do país, em um momento em que muitas de nossas universidades não têm condições de implantar laboratórios locais de redes de computadores.

Para estimular esse uso, precisamos diminuir as barreiras de acesso ao testbed FIBRE. Não é trivial para um professor ou estudante ultrapassar as dificuldades iniciais de uso do testbed. Por isso pensamos em criar um repositório de exercícios propostos e resolvidos que funcionem como módulos prontos, que possam ser usados diretamente por professores em suas disciplinas, e também como instigadores do uso do testbed, servindo como base para o desenvolvimento de outros exercícios.

Esta chamada visa a criação de um repositório de exercícios práticos resolvidos no ambiente do testbed FIBRE, documentados em páginas web e em vídeos, para uso em disciplinas de redes de computadores.

Os exercícios práticos devem conter descrição do experimento fazendo uso de diagramas e figuras para melhor ilustrar, e ainda devem ser gravados em vídeo (com narração em áudio e/ou legenda explicativa) contendo todo passo a passo da seleção, configuração, execução, demonstração dos resultados e liberação de recursos do testbed FIBRE.

A descrição e os vídeos dos exercícios práticos devem ser feitos em duas etapas:

1. *Preparação do Ambiente*
   - 1.1. *Descrição do cenário do experimento;*
   - 1.2. *Procedimentos para seleção de recursos nas ilhas do testbed FIBRE (criação do projeto e slice);*
   - 1.3. *Procedimentos para configuração do Plano de Controle entre os recursos selecionados (incluindo instalação de software, se necessário);*

2. *Execução do Exercício*
   - 2.1. *Procedimentos para configuração do Plano de Dados entre os recursos selecionados (incluindo instanciação de máquinas virtuais e instalação de software, se necessário);*
   - 2.2. *Descrição da execução do experimento quanto ao Plano de Controle;*
   - 2.3. *Descrição da execução do experimento quanto ao Plano de Dados;*
   - 2.4. *Obtenção dos resultados da execução do experimento;*
   - 2.5. *Visualização dos resultados da execução do experimento;*
   - 2.6. *Procedimentos de desalocação dos recursos selecionados (liberação do slice);*

[1] www.fibre.org.br

Cada exercício deverá ainda fazer o uso de duas ilhas de experimentação do FIBRE. I.e., recursos deverão ser alocados em mais de uma instituição.

O que deve ser entregue

- Documento no formato .DOC com a documentação completa passo a passo e ilustrada de um exercício (seguir exemplo do material disponível em http://fibre.org.br/documentation/courseware/).
- Vídeo-aula editada usando a ferramenta de autoria Cacuriá[2] (treinamento será fornecido aos selecionados).

Para a gravação em vídeo da tela do computador, poderá ser usada qualquer ferramenta capaz de gerar vídeos nos formatos MPEG, WMV, AVI ou HTML5. Uma licença temporária do software Camtasia Studio poderá ser disponibilizada.

Prazo e remuneração

Espera-se que as entregas sejam realizadas no prazo de 1 mês.
A remuneração oferecida para essa atividade é de R$ 500,00 (quinhentos reais) líquido, pago em parcela única, na forma de contrato com pessoa física.

Seleção

Nesta primeira chamada serão selecionados até 3 candidatos. Cada candidato deverá documentar 1 exercício escolhido da lista de exercícios em anexo.
Os critérios de seleção serão:

- Experiência de uso e familiaridade do candidato com o testbed FIBRE.
- Experiência prévia na gravação de videoaulas ou instruções em vídeo com gravação de desktop.

Os interessados poderão se candidatar até o dia 8/11/2015.

O resultado da seleção será anunciado até o dia 13/11/2015.

Dúvidas

Questões sobre esta chamada podem ser enviadas para info@fibre.org.br

[2] www.cacuria.com.br

Anexo I

Listamos a seguir 3 exercícios selecionados do Livro "**Redes de computadores e a internet: uma abordagem top-down**", 6ª edição, dos autores James Kurose e Keith Ross. Os códigos-fonte requeridos pelos exercícios estão disponíveis nos links abaixo:

http://fibre.org.br/wp-content/uploads/2015/09/Solutions-to-Programming-Assignments.zip

http://fibre.org.br/wp-content/uploads/2015/09/Retired-Java-Socket-Programming-Solutions.zip

Cada exercício deverá ser executado de forma federada, alocando recursos em duas ilhas do testbed FIBRE. Uma mesma ilha não poderá ser usada por mais de um exercício. Para cada candidado selecionado, serão designadas duas ilhas para a execução do exercício. Por exemplo:

Candidato A executará o Exercício 1 nas ilhas da UFPA e UFG
Candidato B executará o Exercício 2 nas ilhas da RNP e UFSCAR
Candidato C executará o Exercício 3 nas ilhas da USP e UFPE

EXERCÍCIO 1

- Página 130:
  - P31 e P32: execução de códigos em python para testar conexões TCP e UDP (programação de sockets):

P31. Instale e compile os programas Python TCPClient e UDPClient em um hospedeiro e TCPServer e UDPServer em outro.

a. Suponha que você execute TCPClient antes de executar TCPServer. O que acontece? Por quê?

b. Imagine que você execute UDPClient antes de UDPServer. O que acontece? Por quê?

c. O que acontece se você usar números de porta diferentes para os lados cliente e servidor?

P32. Suponha que, em UDPClient.py, depois de criarmos o *socket*, acrescentemos a linha:

```
clientSocket.bind(('', 5432))
```

Será necessário mudar UDPServer.py? Quais são os números de porta para os *sockets* em UDPClient e UDPServer? Quais eram esses números antes dessa mudança?

UDPClient.py

Aqui está o código para o lado cliente da aplicação:

```
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(socket.AF_INET, socket.SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```

UDPServer.py

Vamos agora dar uma olhada no lado servidor da aplicação:

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:

      message, clientAddress = serverSocket.recvfrom(2048)

      modifiedMessage = message.upper()
      serverSocket.sendto(modifiedMessage, clientAddress)
```

TCPClient.py

Eis o código para o lado cliente da aplicação:

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

TCPServer.py

Agora vamos examinar o programa servidor.

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while 1:
      connectionSocket, addr = serverSocket.accept()
      sentence = connectionSocket.recv(1024)
      capitalizedSentence = sentence.upper()
      connectionSocket.send(capitalizedSentence)
      connectionSocket.close()
```

EXERCÍCIO 2

- Página 131:
  - Tarefa 2: UDP Pinger:

Tarefa 2: UDP Pinger

Nesta tarefa de programação, você escreverá um programa *ping* do cliente em Python. Seu cliente enviará uma mensagem *ping* simples a um servidor, receberá uma mensagem *pong* correspondente de volta do servidor e determinará o atraso entre o momento em que o cliente enviou a mensagem *ping* e recebeu a mensagem *pong*. Esse atraso é denominado tempo de viagem de ida e volta (*round-trip time* — RTT). A funcionalidade oferecida pelo cliente e servidor é semelhante à fornecida pelo programa *ping* padrão, disponível nos sistemas operacionais modernos. Porém, os programas *ping* padrão usam o Internet Control Message Protocol (ICMP) (que veremos no Capítulo 4). Aqui, criaremos um programa *ping* baseado em UDP, fora do padrão (porém simples!).

Seu programa *ping* deverá enviar 10 mensagens *ping* ao servidor de destino por meio de UDP. Para cada mensagem, seu cliente deverá determinar e imprimir o RTT quando a mensagem *pong* correspondente for retornada. Como o UDP é um protocolo não confiável, um pacote enviado pelo cliente ou servidor poderá ser perdido. Por esse motivo, o cliente não poderá esperar indefinidamente por uma resposta a uma mensagem *ping*. Você deverá fazer que o cliente espere até 1 s por uma resposta do servidor; se nenhuma resposta for recebida, o cliente deverá considerar que o pacote foi perdido e imprimir uma mensagem de acordo.

Nesta tarefa, você receberá o código completo para o servidor (disponível no site de apoio). Sua tarefa é escrever o código cliente, que será semelhante ao código do servidor. Recomendamos que, primeiro, você estude cuidadosamente o código do servidor. Depois, poderá escrever seu código cliente, cortando e colando à vontade as linhas do código do servidor.

EXERCÍCIO 3

Capítulo 8 - Segurança em Redes de Computadores

- Página 549:
  - P1: usar o código dos exercícios P31 e P32 (EXERCICIO 1 - programação de sockets: UDPClient.py / UDPServer.py / TCPClient.py / TCPServer.py) para implementar o envio de mensagens criptografadas a partir dos códigos de Criptografia de Chaves Simétricas (página 498 a 500):

P1. Usando a cifra monoalfabética da Figura 8.3, codifique a mensagem "This is an easy problem" (este é um problema fácil). Decodifique a mensagem "rmij'u uamu xyj".

- Um cifra monoalfabética (página 499) ~ Figura 8.3:

FIGURA 8.3 UM CIFRA MONOALFABÉTICA

Letra no texto aberto:	a b c d e f g h i j k l m n o p q r s t u v w x y z
Letra no texto cifrado:	m n b v c x z a s d f g h j k l p o i u y t r e w q

- Um cifra polialfabética que utiliza duas cifras de césar ~ Figura 8.4:

FIGURA 8.4 UMA CIFRA POLIALFABÉTICA QUE UTILIZA DUAS CIFRAS DE CÉSAR

Letra do texto aberto:	a b c d e f g h i j k l m n o p q r s t u v w x y z
$C_1(k = 5)$:	f g h i j k l m n o p q r s t u v w x y z a b c d e
$C_2(k = 19)$:	t u v w x y z a b c d e f g h i j k l m n o p q r s