Universidade Federal do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Processamento Paralelo e Distribuído – Turmas 01 e 02 (EARTE) – 2022/1

Prof. Rodolfo da Silva Villaça – rodolfo.villaca@ufes.br

Laboratório VI – Especificações Estendidas – Alteração 09/08

1. Limites do Desafio (*Challenge*)

- Desconsiderar a definição de desafio (*challenge*) anterior, contido na especificação original do Laboratório VI, que tinha desafios no intervalo de [1..10] *bits*;
- Considerar que os desafios (*challenge*) propostos pelos líderes deverão estar no intervalo [20..40] *bits*, definidos aleatoriamente (será verificado no código fonte durante a avaliação do trabalho).

2. Semente (*seed*)

- Não serão aceitas sementes (*seeds*) de *hashing* no formato inteiro. As sementes (*seeds*) NÃO poderão ser codificadas como inteiros;
- Sempre usar o *str.encode()* como entrada da função *hashing* SHA-1 (biblioteca *hashlib* em Python);
- As sementes *(seeds)* deverão ser representadas no formato texto, codificado em caracteres de 8 *bits* do tipo ASCII, compatíveis com o tipo *str* em Python. Especificamente, a semente gerada deverá ser composta somente por caracteres MAIÚSCULOS, minúsculos, dígitos decimais e pontuação, listados a seguir:
  - **ascii_lowercase**: abcdefghijklmnopqrstuvwxyz
  - **ascii_uppercase:** ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - **digits:** 0123456789
  - **punctuation:** !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~.
- NÃO usar caracteres acentuados ou sensíveis à localidade.

Exemplo: a *string* “245” NÃO pode ser representada pelo Byte ‘11110101’$_2$, mas precisará ser representada por 3 Bytes, especificamente ‘01110010 01110100 01110101’ $_2$, que representam os caracteres ‘2’, ‘4’ e ‘5’. Este deverá ser o valor usado como entrada da função *hashing* SHA-1.

3. Mensagens

- Usar uma codificação baseada na estrutura de dados dicionário, na linguagem Python, com chaves e valores conforme a tabela abaixo:

Mensagem	**Chaves de Dicionário***	**Tipos Internos**	**Fila no Rabbit**
InitMsg	NodeId	int	ppd/init
PubKeyMsg	NodeId PubKey	int str	ppd/pubkey
ElectionMsg	NodeId ElectionNumber Sign	int int str	ppd/election

ChallengeMsg	NodeId TransactionNumber Challenge Sign	int int int str	ppd/challenge
SolutionMsg	NodeID TransactionNumber Seed Sign	int int str str	ppd/solution
VotingMsg	NodeID TransactionNumber Seed Vote Sign	int int str int, 0 (Não) ou ≠0 (Sim) str	ppd/voting

* A ordem das chaves no dicionário é muito relevante. NÃO alterar!

- Entende-se que a assinatura (*Sign*) não faz parte da estrutura interna do dicionário, mas sim, apenas à mensagem enviada ao *broker* pelo publicador e recebida pelos assinantes;
- Na transmissão, a mensagem deverá ser codificada em formato JSON, com strings no formato ASCII, e adicionadas da assinatura no final do dicionário (quando for o caso).

4. JSON

- Usar a biblioteca *json*[1] em Python, com *indent=None (default)* e *ensure_ascii=True (default)*, para codificar dicionários em strings no formato JSON*;*

a) Suponha que se deseja criar uma mensagem SolutionMsg onde ‘NodeID’=635251, ‘TransactionNumber’=13, e ‘Seed’=“213wyuyqgde2&*!@#ewiu12”.
b) Antes de assinar, você deverá criar um dicionário como:
>>> solutionMsgDict = {‘NodeID’:635251, ‘TransactionNumber’:13, e ‘Seed’:“213wyuyqgde2&!@#ewiu12”}*
c) Utilize a função *dumps* da biblioteca *json* do Python para codificar esse dicionário em uma *string* no formato JSON:
>>> solutionMsg = json.dumps(solutionMsgDict)
d) A partir de uma *string* no formato JSON, recupere o dicionário original usando a função *loads* da biblioteca *json* do Python:
>>> solutionMsgDict = json.loads(solutionMsg)

- No exemplo, assine a *string solutionMsg*. Suponha que a assinatura gerada (*Sign*), em formato texto, seja (essa não é uma assinatura real neste exemplo!):

Sign =
b1599cea8fd8ada2f133bd127a9b5c6dea1cedc4804d595b08df6bcd0ee15fca5c26b12c48e1ba83d38
2859805b5375bffe68c127a11b137d9ce8d20f18a13ec507f9177d23039f01ea96b95a8f16841105ee44

1 https://docs.python.org/3/library/json.html#

6f3bacc4c4a1a97261c03a6168f879f4ace79ae42abd3c485f1dbb23ba2e683a7c3f7cf1c6f67204fbe955437

- Faça um *update* na mensagem (*solutionMsg*) com mais um novo dicionário (SignDict) contendo somente a chave *Sign*, gerando então uma mensagem assinada (*solutionMsgSigned*):

```
>>> signDict = {'Sign':
"b1599cea8fd8ada2f133bd127a9b5c6dea1cedc4804d595b08df6bcd0ee15fca5c26b12c48e1ba83d382859805b5375bffe68c127a11b137d9ce8d20f18a13ec507f9177d23039f01ea96b95a8f16841105ee446f3bacc4c4a1a97261c03a6168f879f4ace79ae42abd3c485f1dbb23ba2e683a7c3f7cf1c6f67204fbe955437"}
>>> solutionSigned = json.dumps(solutionMsg)
>>> solutionSigned.update(signDict)
>>> solutionMsgSigned = json.dumps(solutionSigned)
```

- A *string solutionMsgSigned* deverá ser publicada no *broker* como SolutionMsg*;*
- Ao receber a SolutionMsg assinada (*solutionMsgSigned)*, o receptor deverá ler a assinatura e armazenar localmente e em seguida, remover a chave *Sign* usando a função *del*, nativa do tipo de dados dicionário em Python;
- Pronto, a SolutionMsg original, antes da assinatura, poderá ser verificada com a assinatura recebida.

5. Competição Final da Disciplina

- Data: 18/08 (quinta-feira), 17h às 19h de duração total no LabGrad3;
- A competição acontecerá por 50 minutos, de 17:40h às 18:30h;
- A participação é opcional, porém só acontecerá se pelo menos 3 grupos confirmarem presença no dia 16/08;
- No dia 16/08 (terça-feira) os grupos que estivem prontos, poderão testar seus códigos e fazer uma competição teste;
- Apenas 1 e somente 1 nó de cada grupo poderá publicar/assinar mensagens no *broker*;
- O código fonte deverá estar público no *github* às 17h do dia 18/08;
- Os grupos deverão publicar o *link* do *github* no Classroom (aberto) até as 17h do dia 18/08;
- Os grupos, antes de começar a competição, precisarão atualizar suas cópias locais antes de iniciar o seu nó na rede;
- PENDENTE: disponibilizar o IP do broker!
- Premiação:
  - 1º lugar: +0,5 pontos na Média Semestral de cada integrante do grupo que estiver presente na competição + Gift Card de R$50,00 no IFood para o grupo;
  - 2º lugar: +0,25 pontos na Média Semestral de cada integrante do grupo que estiver presente na competição;
  - 3º lugar: +0,1 pontos na Média Semestral de cada integrante do grupo que estiver presente na competição;

- Importante: os pontos só serão computados se toda a especificação for cumprida (verificação posterior) e se o trabalho funcionar durante a competição. Se não funcionar, não ganhará!

OBSERVAÇÃO: me lembrem se eu omiti algo relevante na opinião de vocês! Desde já agradeço muito… :D