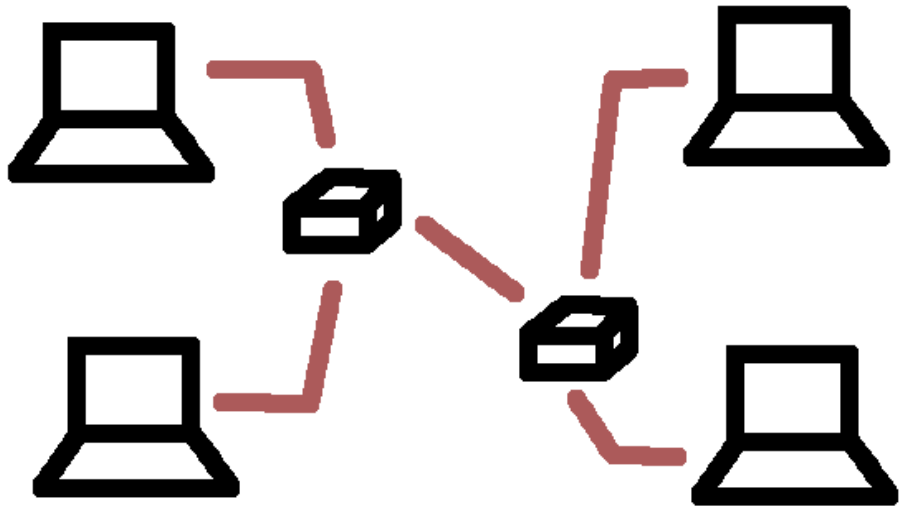
NetSim: um simulador de redes simplificado.

Carlos Eduardo Leão Elmadjian Renan Fichberg

Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP)

23 de Novembro de 2014



Conteúdo

- O simulador: como funciona?
- Testes realizados com o simulador
- Dificuldades encontradas

O simulador: como funciona?

- Estrutura do NetSim
- Entradas
- Saídas
- As classes e as suas funções

O NetSim é dividido em módulos e possui um número considerável de classes. Os arquivos Java ficam na pasta principal, mas o programa também faz uso de outros dois subdiretórios:

- **/inputs** - Contém arquivos .txt com entradas para alimentar o simulador. É aqui que o usuário deve deixar suas entradas, **obrigatoriamente**.
- **/logs** - Contém arquivos .log com as saídas. As saídas são os pacotes capturados pelos *Sniffers* definidos na entrada que alimentou o programa.

O simulador: como funciona?

- Estrutura do NetSim
- Entradas
- Saídas
- As classes e as suas funções

Há no total apenas **12** tipos de entrada esperadas pelo programa. São elas:

- Entrada para criação de computadores (*hosts*)
- Entrada para criação de roteadores (*routers*)
- Entrada para criação de enlaces do tipo duplex-link
- Entrada para configuração dos *hosts* com relação aos endereços de IP do próprio computador, do roteador padrão e do servidor DNS

- Entrada para configuração dos *routers* com relação às portas e aos endereços de IP
- Entrada para a configuração dos *routers* com relação às rotas
- Entrada para a configuração dos *routers* com relação aos seus dados de *performance*
- Entrada para a configuração dos agentes da camada de aplicação: declaração do agente e da sua natureza

- Entrada para a configuração dos agentes da camada de aplicação: associação do agente declarado a um *host*
- Entrada para a configuração dos agentes da camada de aplicação: declaração dos *sniffers*
- Entrada para a configuração dos agentes da camada de aplicação: locais da rede onde os *sniffers* agirão.
- Entrada para a configuração das comunicações entre os agentes.

O simulador: como funciona?

- Estrutura do NetSim
- Entradas
- Saídas
- As classes e as suas funções

As saídas são os resultados das capturas dos pacotes pelo *sniffers*, e ficam armazenadas por padrão no subdiretório de logs. Cada *sniffer* tem seu próprio log, cujo o nome do arquivo tem o formato **nome_do_sniffer.log**. Este comportamento pode ser alterado caso o usuário passe uma saída da sua escolha.

O formato da saída é o seguinte:

- Identificador do pacote
- Instante de tempo em que o pacote foi visto (a partir da execução do programa)
- Identificador do *sniffer*
- Informações da camada de rede (IP):
  - IP de origem
  - IP de destino
  - Identificação do protocolo da camada acima
  - Tamanho cabeçalho IP + tamanho das camadas superiores
  - TTL

- Informações da camada de transporte (se for TCP):
  - Porta origem
  - Porta destino
  - Tamanho cabeçalho TCP + tamanho da camada superior
  - Número de Seqüência
  - Número de Reconhecimento
  - Bit ACK
  - Bit FIN
  - Bit SYN
- Informações da camada de transporte (se for UDP):
  - Porta origem
  - Porta destino
  - Tamanho cabeçalho UDP + tamanho da camada superior

- Informações da camada de aplicação (DNS/FTP/HTTP):
  - Pergunta ou resposta contida no pacote.

Ainda, além das informações estarem presentes no log, a cada captura os resultados daquele *sniffer* são impressos no *prompt* em tempo de execução.

O simulador: como funciona?

- Estrutura do NetSim
- Entradas
- Saídas
- As classes e as seus papeis

- Agent: classe abstrata que representa um agente da rede
- ApplicationLayer: classe que representa uma camada de aplicação
- Clock: classe responsável pelo tempo de execução
- DNSServer: classe que representa um servidor DNS
- DuplexLink: classe que representa um enlace do tipo *duplex-link*

- FTPClient: classe que representa um cliente FTP
- FTPServer: classe que representa um servidor FTP
- Host: classe que representa um *host* da rede
- HTTPClient: classe que representa um cliente HTTP
- HTTPServer: classe que representa um servidor HTTP

- InputReader: classe responsavel por manipular os arquivos de entrada do simulador
- NetSim: classe que representa o simulador de redes
- Node: classe que representa um nó. Um nó pode ser tanto um *host* quanto um *router*
- Packet: classe que representa um pacote
- Router: classe que representa um *router* da rede

- RouterBuffer: classe que representa um buffer do *router*. Funciona em FIFO
- SimulatorLogger: classe responsável por escrever os arquivos de saída
- Sniffer: classe que representa um *sniffer*
- TCP: classe que armazena os dados do TCP na camada de transporte
- TransportLayer: classe abstrata que representa uma camada de transporte
- UDP: classe que armazena os dados do UDP na camada de transporte

O simulador: como funciona?
Exemplo de saída

```
Packet Identification: ac7a298c-a439-41d8-a5b9-c9b288e52db0
Time Elapsed (from the start of the program execution): 1494ms
Sniffer: sniffer1
Internet Layer (IP)
        Source IP: 10.0.0.1
        Destination IP: 192.168.2.2
        Upper Layer Protocol Identification: 6
        Packet Length (IP header + upper layers):  20 + 20
        TTL: 62
Transport Layer (TCP)
        Source Port: 54823
        Destination Port: 80
        Packet Length (TCP header + upper layer):  20 + 0
        Sequence Number: 0
        Recognition Number: 0
        Bit ACK: 0
        Bit FIN: 0
        Bit SYN: 1
Application Layer ( )
        Text Data:
-------------------------------------------------------
```

Início do 3-way-handshake pelo cliente 10.0.0.1

O simulador: como funciona?
Exemplo de saída

```
------------------------------------------------------------

Packet Identification: d1b2340c-3f12-4560-9940-f5450f0420bc
Time Elapsed (from the start of the program execution): 10440ms
Sniffer: sniffer1
Internet Layer (IP)
        Source IP: 10.0.0.1
        Destination IP: 192.168.2.2
        Upper Layer Protocol Identification: 6
        Packet Length (IP header + upper layers):  20 + 53
        TTL: 62
Transport Layer (TCP)
        Source Port: 54823
        Destination Port: 80
        Packet Length (TCP header + upper layer):  20 + 33
        Sequence Number: 1
        Recognition Number: 711
        Bit ACK: 1
        Bit FIN: 1
        Bit SYN: 0
Application Layer (HTTP)
        Text Data: HTTP/1.1 200 OK
Server: Apache

------------------------------------------------------------

System shutting down NOW...
```

Fim da transmissão, evidenciado pelo bit FIN

O simulador: como funciona?

- Estrutura do NetSim
- Entradas
- Saídas
- As classes e as seus papeis

Conteúdo

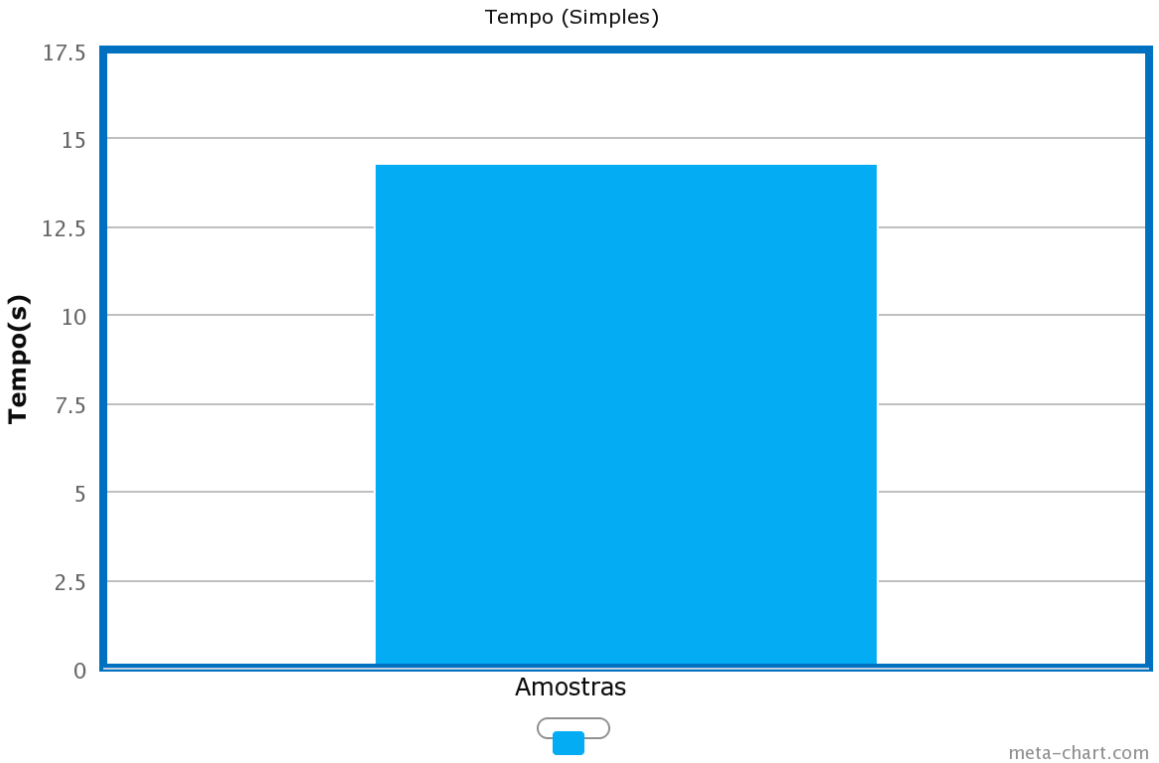
- O simulador: como funciona?
- Testes realizados com o simulador
- Dificuldades encontradas

Testes realizados com o simulador

Aqui serão apresentados gráficos dos testes. O valor da barra representado nas imagens é a **média**. Mais informações podem ser encontradas no arquivo **testes.txt**. Valores medidos para um intervalo de confiança de 95%.

Testes realizados com o simulador

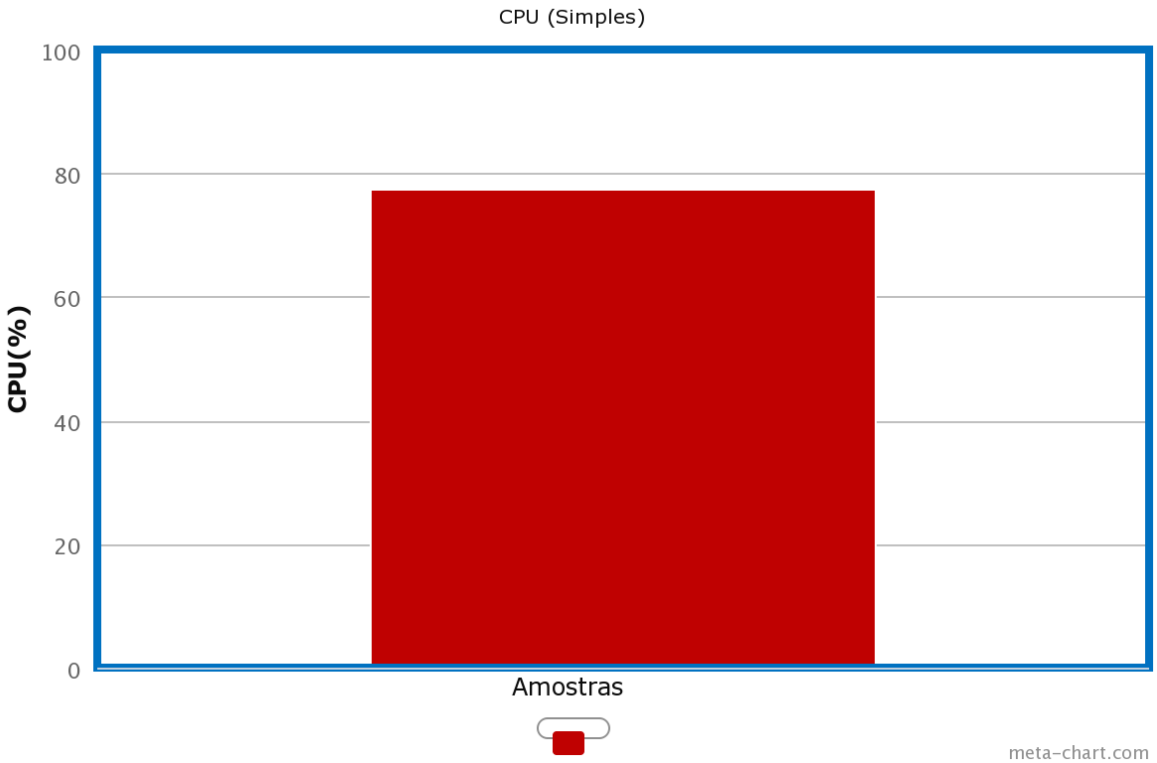
Cenário simples - Tempo



$$\text{Média} = 14.3107$$

$$\text{IC} = [14.294149138 \ ; \ 14.327250862]$$

Testes realizados com o simulador
Cenário simples - CPU

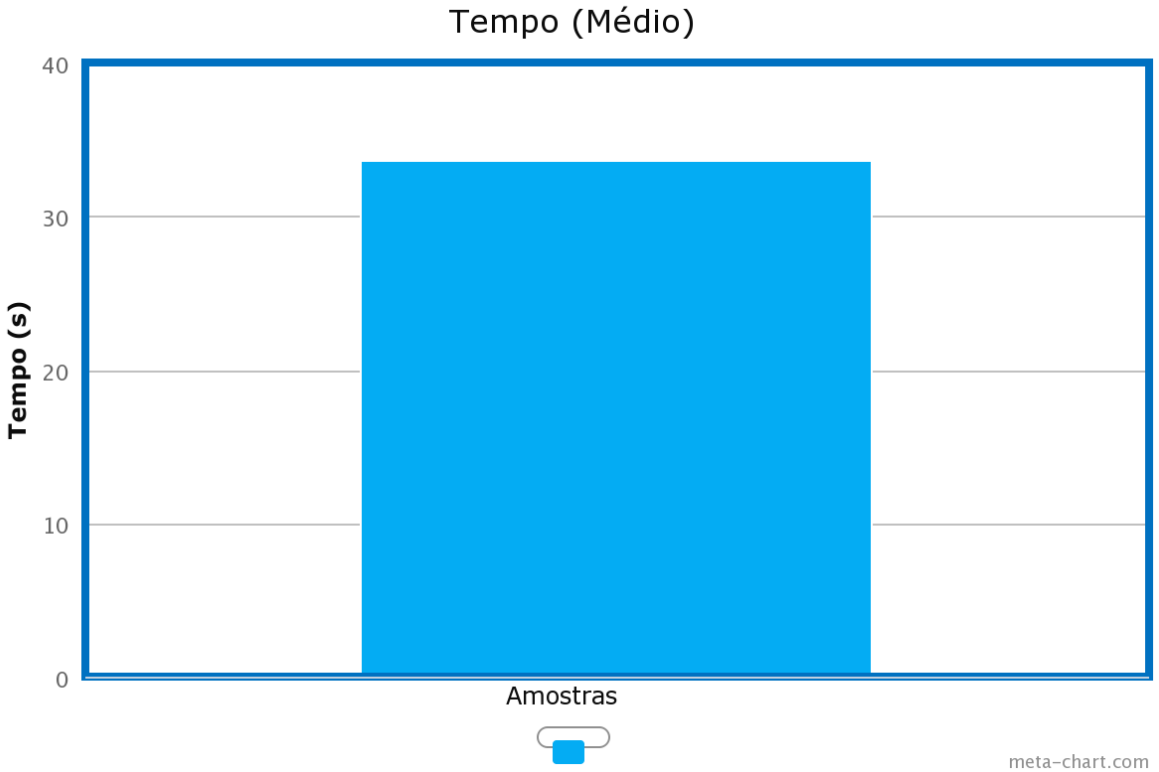


$$\text{Média} = 77.3667$$

$$\text{IC} = [77.18370188 \ ; \ 77.54969812]$$

Testes realizados com o simulador

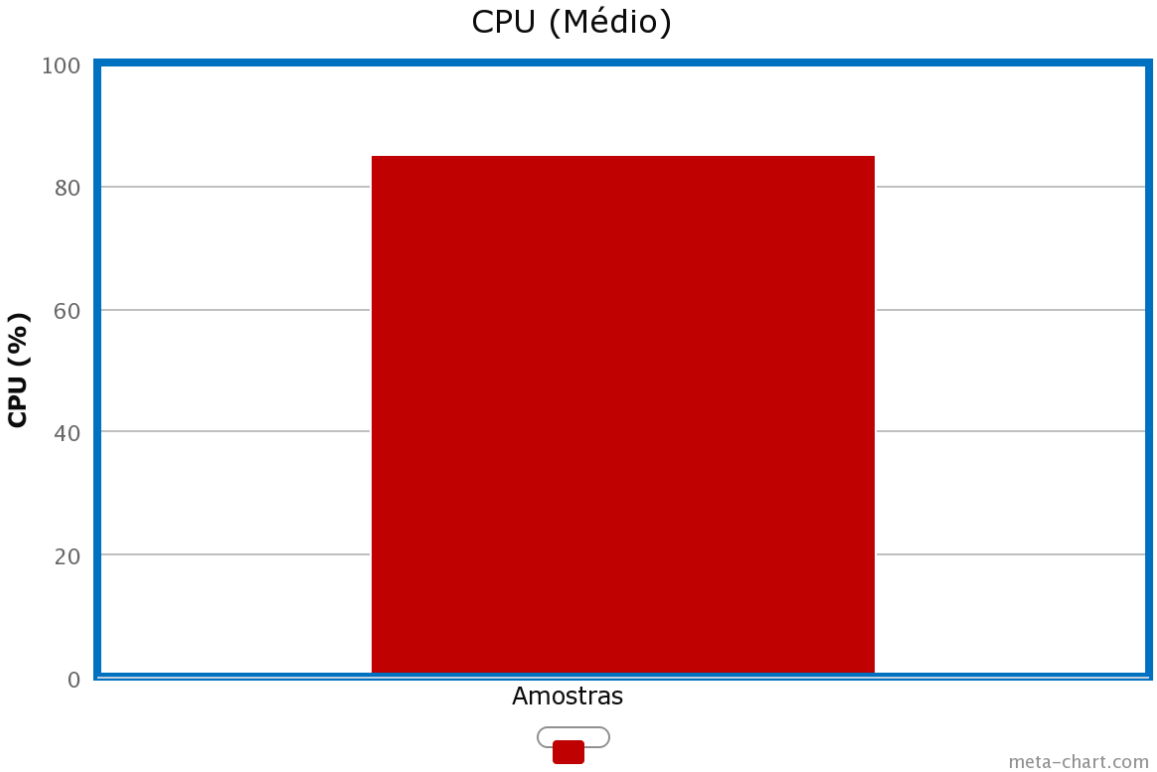
Cenário intermediário - Tempo



$$\text{Média} = 33.615$$

$$\text{IC} = [33.570252673 \ ; \ 33.659747327]$$

Testes realizados com o simulador
Cenário intermediário - CPU

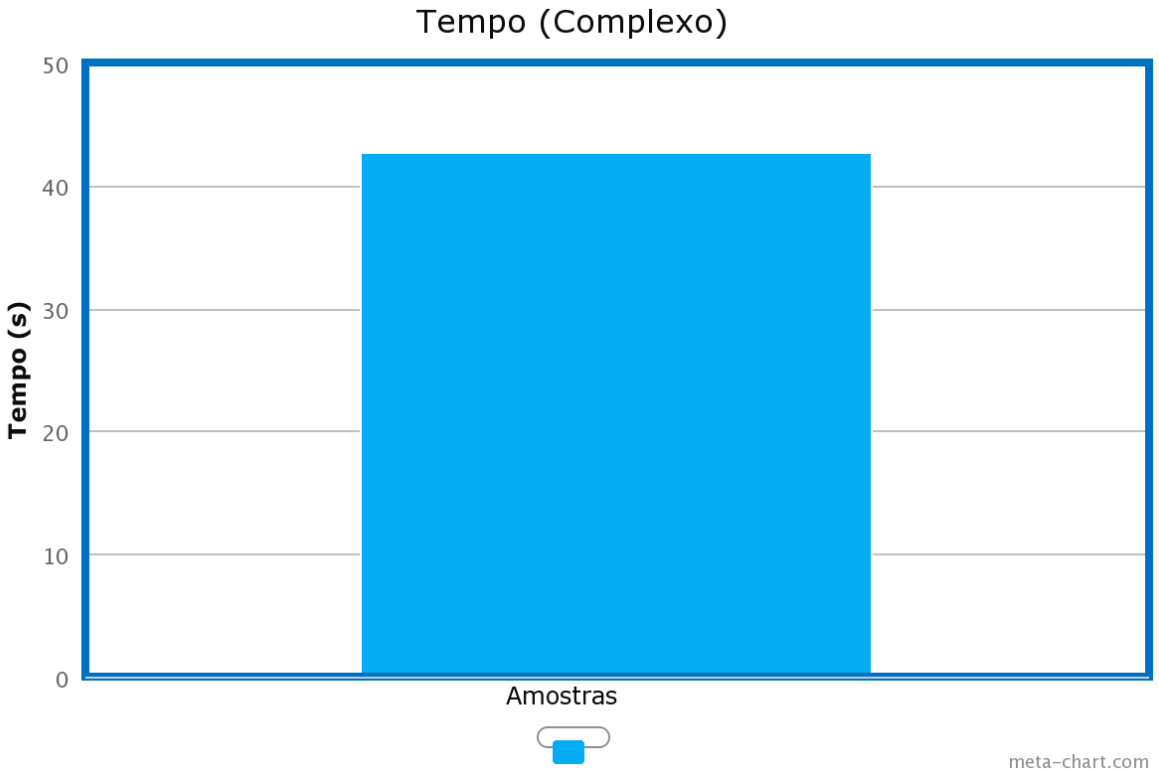


$$\text{Média} = 85$$

$$\text{IC} = [85 \ ; \ 85]$$

Testes realizados com o simulador

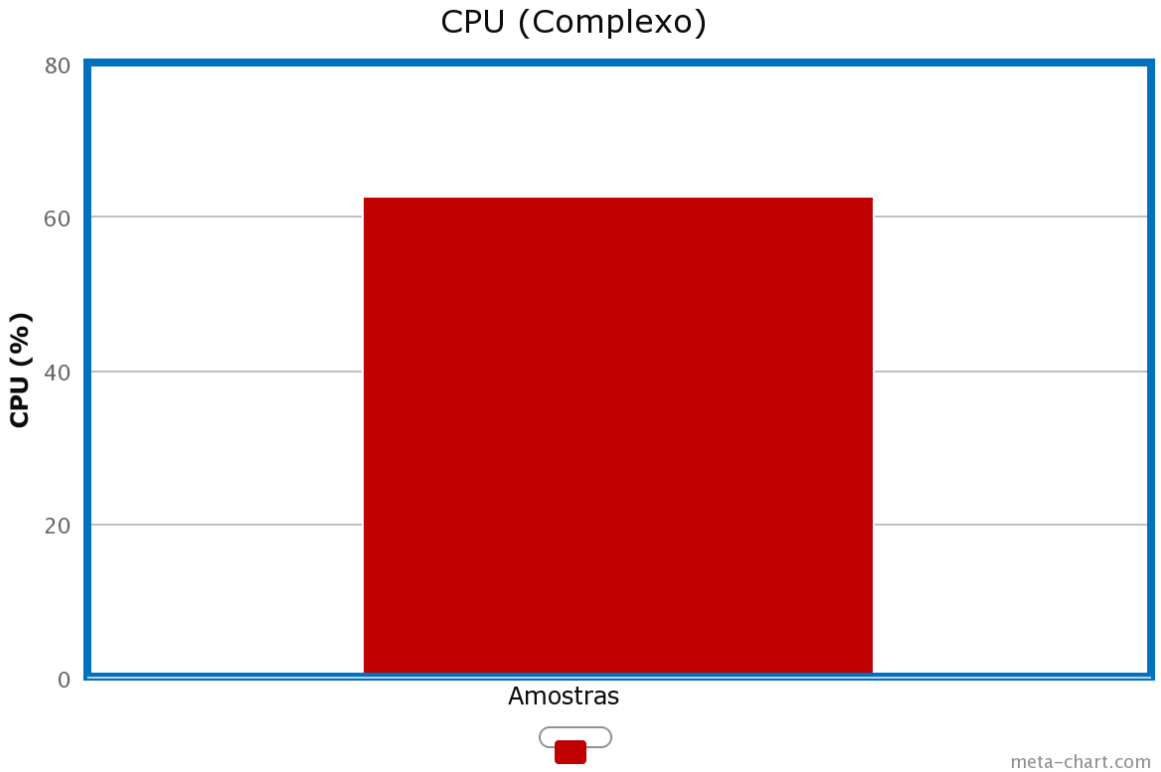
Cenário complexo - Tempo



$$\text{Média} = 42.7793$$

$$\text{IC} = [42.693827598 \ ; \ 42.864772402]$$

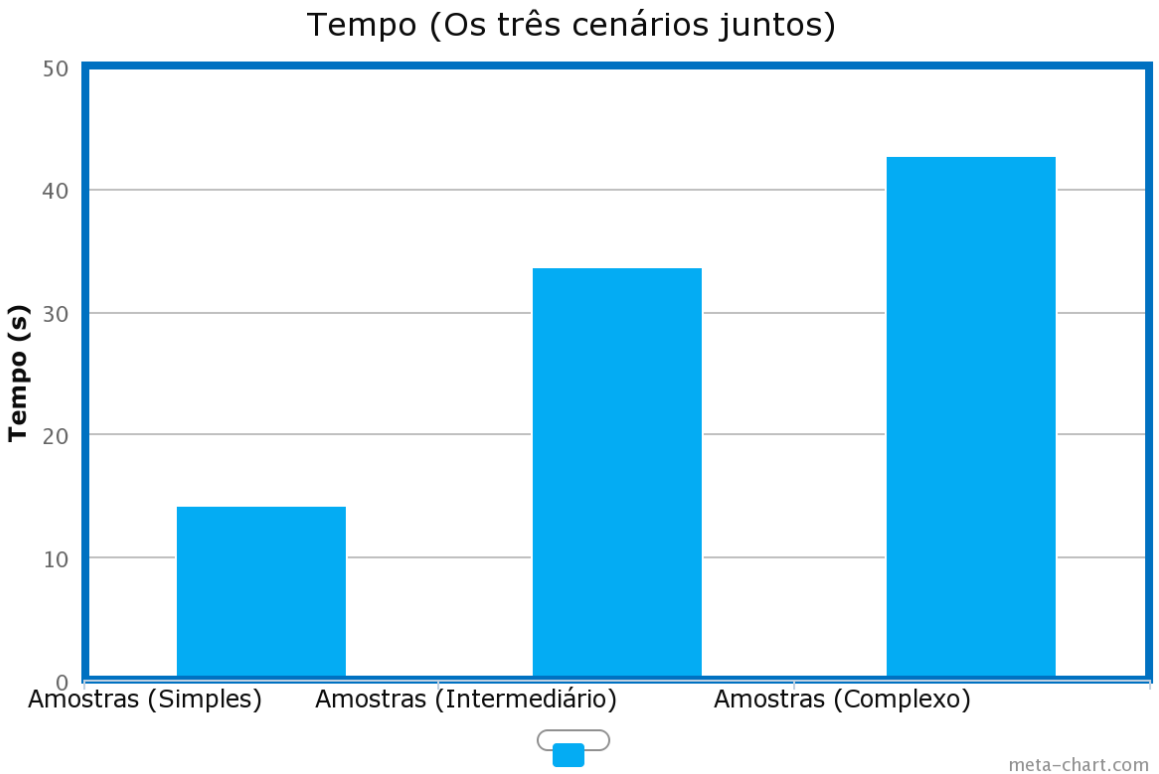
Testes realizados com o simulador
Cenário complexo - CPU



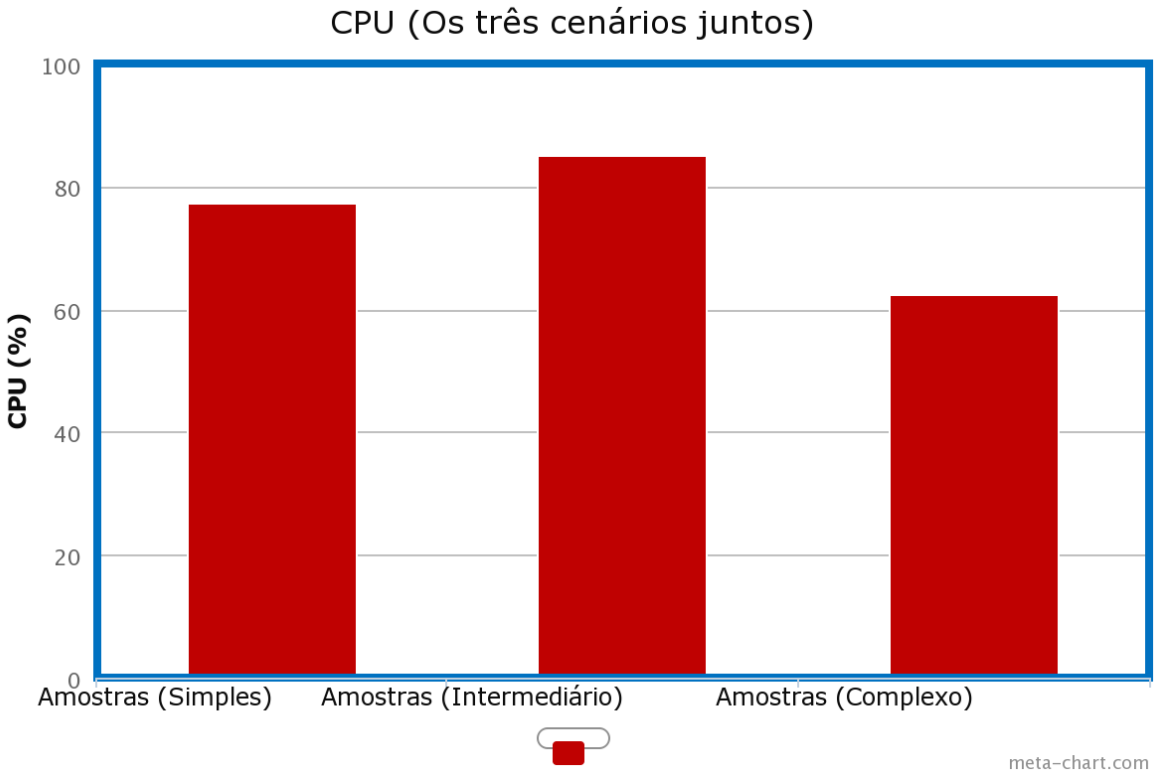
$$\text{Média} = 62.5667$$

$$\text{IC} = [62.378521825 \ ; \ 62.754878175]$$

Testes realizados com o simulador
Os três cenários de tempo



Testes realizados com o simulador
Os três cenários de CPU



- Observamos uma variância crescente no *tempo* conforme a complexidade do cenário
- Não é possível inferir nada sobre a variância no tempo de CPU
- O experimento mais complexo possui uma rede distribuída e disparos mais intervalados, o que explica o baixo consumo de CPU em comparação com os demais

- Máquina real:
  - Intel i5 2500K 3.3GHz, 8GB de RAM
  - HD Seagate 7200RPM 1TB
  - Sistema Operacional: Arch Linux x86_64 (kernel 3.17.2-1)

Conteúdo

- O simulador: como funciona?
- Testes realizados com o simulador
- Dificuldades encontradas

Dificuldades encontradas e pontos positivos

Dificuldades:

- Planejamento: encontrar a estrutura ideal que comporte os seis protocolos requisitados
- Concorrência: encontrar a forma ideal para lidar com todos os pacotes circulando pela rede simultaneamente
- Integração entre todas as classes do programa

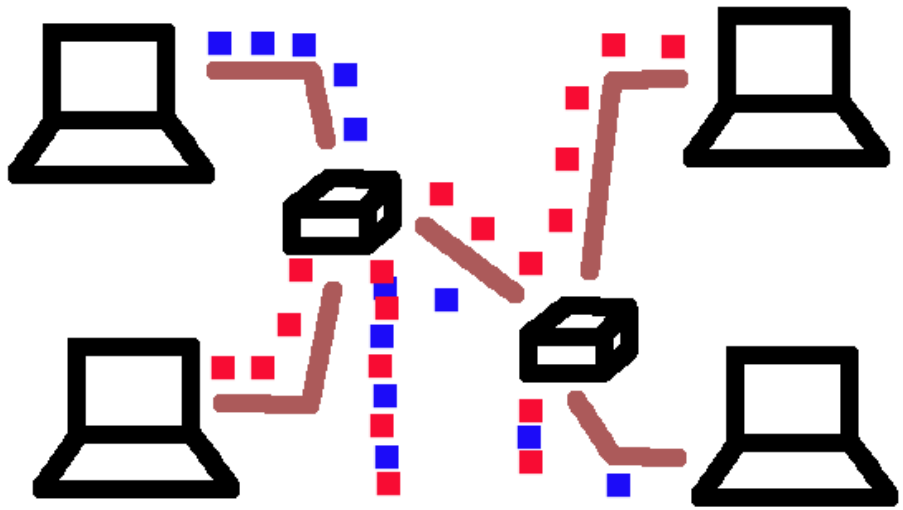
Dificuldades encontradas e pontos positivos

Pontos positivos:

- Aprendizado: mais conhecimento sobre o funcionamento das camadas e como elas interagem
- Menos problemas para planejar a estrutura do programa graças à bagagem do EP2

Conteúdo

- O simulador: como funciona?
- Testes realizados com o simulador
- Dificuldades encontradas e pontos positivos



Obrigado! :)