DCC023 – Redes de computadores

Trabalho Prático 2 – SNMP-DCC

Professores: Daniel Fernandes Macedo, Marcos Augusto M. Vieira

Monitor: Lucas Augusto Maia da Silva

Data de entrega: 02/12/2015

Trabalho em grupos de dois alunos (ou individual).

Valor do trabalho: 15 pontos

1. Introdução

SNMP (Simple Network Management Protocol) é um protocolo utilizado para gerenciar máquinas em uma rede IP. Este protocolo possui grande usabilidade para administradores de sistemas (Sysadmin), profissão que gerencia os recursos computacionais de um centro de dados (data center). Neste trabalho vocês irão criar uma versão simplificada deste protocolo para obter informações específicas de máquinas conectadas em uma rede utilizando os protocolos UDP e IPv6.

2. Programas a serem desenvolvidos

O trabalho prático consistirá na implementação de dois programas, o agente e o monitor. Ambos vão receber parâmetros de entrada como explicado a seguir:

```
agente  <ipv6> <porta>

monitor <porta>
```

O primeiro programa, o agente, irá conectar no monitor definido pelo endereço IPv6 e porta passados por parâmetro. Após recebida as informações o agente imprime na tela as informações solicitadas e termina a sua execução.

Já o monitor irá receber requisições utilizando a pilha IPv6/UDP e enviará informações sobre a máquina em que ele estiver executando, a sua execução é terminada utilizando o comando CTRL + C. O monitor deve ser capaz de receber várias conexões em série. A porta a ser escutada é dada pelo parâmetro único do programa.

3. Protocolo SNMP-DCC

O protocolo SNMP-DCC irá recuperar as seguintes informações da máquina em que está hospedado:

1 – Nome da máquina

2 – Hora/ Up Time (tempo em que a máquina está ligada)

3 – Endereço IPv6

4 – Endereço IPv4

5 – Espaço em disco

O protocolo utilizará endereços IPv6 e o protocolo de transporte UDP. São definidos dois tipos de mensagem, Request e Answer. Mensagens do tipo Request são enviadas no sentido agente -> monitor, e mensagens do tipo Answer são enviadas no sentido monitor -> agente. Request são mensagens de apenas 7 bytes, contendo a string "request". Elas são utilizadas para identificar a solicitação de informações sobre uma máquina. As mensagens do tipo Answer possuem no máximo 1024 bytes e são utilizadas para transportar as informações solicitadas pelo agente. As duas mensagens são codificadas em ASCII.

3.1. Implementação

O primeiro passo para implementar o trabalho prático é habilitar o protocolo IPv6 na máquina em que será executado o código. Para testar se sua máquina suporta IPv6 utilize o seguinte comando no terminal:

```
ping6 –c5 ::1
```

Ping6 é análogo ao ping, voltado para o protocolo IPv6. O endereço ::1 é o localhost ou 127.0.0.1 para endereços IPv6. Caso o seu computador não passe nesse teste, o seguinte link contém mais informações para ativação do IPv6:

http://ipv6.br/post/habilitando-ipv6-no-linux/

Para implementação da biblioteca sockets utilizando IPv6 e UDP basta seguir os tutoriais providos pelos links e pelo livro disponibilizado no Moodle.

A coleta de dados será feita via chamadas de sistema, para isso a linguagem C oferece a função *system(char * command).* A partir dessa função é possível chamar os programas executados no terminal. Para obter a saída de cada chamada, basta redirecionar a saída do programa para um arquivo de texto e depois enviar o seu conteúdo para o agente. Por exemplo o comando *hostname* é utilizado para retornar o nome do computador. A seguinte chamada pode ser feita utilizando a função system:

```
system("hostmane >> output.txt");
```

Assim, o nome da máquina será salvo no arquivo output.txt. Para coletar os outros dados solicitados pelo programa, serão utilizados os seguintes comandos:

```
system("w | head –1 >> output.txt");

system("ifconfig | grep \"inet6 addr\">> output.txt" );

system("ifconfig | grep \"inet addr\">> output.txt");

system("df –kh >> output.txt");
```

O programa *w* é utilizado para retornar hora e há quanto tempo a máquina está ligada. O programa *ifconfig* é responsável por retornar informações sobre as interfaces de rede do computador. Por último, o programa *df* é responsável por retornar a utilização do espaço em disco da máquina. A resposta a ser enviada para o agente é uma string de 1024 bytes, contendo as informações solicitadas pelo agente. Caso a informação gerada pela saída dos comandos ultrapasse 1024 bytes, o programa deve truncar os dados lidos em 1024 bytes.

4. Entrega do código

O código e a documentação devem ser entregues em um arquivo Zip (não pode ser RAR nem .tgz nem .tar.gz) no Moodle, contendo um arquivo **readme.txt** com o nome dos integrantes, e um arquivo PDF da documentação. Incluam todos os arquivos (.c, .h, makefile, não incluam executáveis ou arquivos objeto) EM UM ÚNICO DIRETÓRIO. Um makefile deve ser fornecido para a compilação do código.

Parte desse trabalho envolve o aprendizado de como construir um makefile e utilizar a ferramenta Make. Este makefile, quando executado sem parâmetros, irá gerar os dois programas, *agente* e *monitor*, EXATAMENTE com esses nomes. Submissões onde os programas não seguem as especificações de parâmetros e nomes, makefiles não funcionando ou arquivos necessários faltando não serão corrigidos. Programas que não compilarem também não serão corrigidos. O julgamento do trabalho será feito em um computador rodando o SO Linux.

5. Documentação

A documentação, além de ser entregue com o Zip junto ao código, deve ser entregue impressa ao professor até o dia seguinte à entrega, na sala de aula. Caso não tenhamos aula no dia seguinte ao prazo para envio, iremos combinar antecipadamente qual é a forma mais conveniente de entrega da documentação. Trabalhos que forem entregues no Moodle mas sem a documentação impressa **não serão** corrigidos.

O texto da documentação deve ser breve, de forma que o corretor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deve conter os seguintes itens:

- Sumário do problema a ser tratado
- Uma descrição sucinta dos algoritmos e dos tipos abstratos de dados, das principais funções, e procedimentos e as decisões de implementação
- Decisões de implementação que porventura estejam omissos na especificação
- Testes, mostrando que o programa está funcionando de acordo com a especificação, seguidos da sua análise

- Conclusão e referências bibliográficas

6. Avaliação

A avaliação do trabalho será composta pela execução dos programas desenvolvidos e pela análise da documentação. Os seguintes itens serão avaliados:

- A qualidade do código (código bem organizado, com comentários explicativos, variáveis com nomes intuitivos, modularidade, etc).

- Execução correta do código em entradas de testes, a serem definidas no momento da avaliação. As entradas de teste irão exercitar a funcionalidade completa do código e testar casos especiais ou de maior dificuldade de implementação, mas que devem ser tratados por um programa correto.

- Aderência ao protocolo especificado. Iremos usar ferramentas de captura do tráfego da rede (*tcpdump*, *wireshark*) e iremos analisar o código para verificar se a comunicação está implementada da forma definida na documentação.

- Conteúdo da documentação, que deve conter os itens mencionados anteriormente.

- Coerência e coesão da documentação (apresentação visual e organização, uso correto da língua, qualidade textual e facilidade de compreensão).

Como mencionado anteriormente, trabalhos fora da especificação (por exemplo, sem makefile, que não gerem programas com os nomes especificados, que não compilem ou não possuam os parâmetros esperados) não serão corrigidos. Trabalhos fora da especificação tomam muito trabalho do corretor, que deve entender como compilar e rodar **cada programa avaliado**, tempo esse que deveria ser empregado para avaliar a execução e corretude do código.

Qualquer elemento que não esteja especificado neste documento, mas que tenha que ser inserido para que o protocolo funcione, deve ser descrito na documentação de forma explícita.

7. Desconto de nota por atraso

Os trabalhos poderão ser entregues até a meia-noite do dia especificado para a entrega. O horário de entrega deve respeitar o relógio do sistema Moodle, ou seja, a partir de 00:01 do dia seguinte à entrega no relógio do Moodle, os trabalhos já estarão sujeitos a penalidades.

A fórmula para desconto por atraso na entrega do trabalho prático ou resenha é:

$\text{Desconto} = 2^{d-1}/0.32\ \%$

onde *d* é o atraso em dias úteis. Note que após 5 dias úteis, o trabalho não pode ser mais entregue.

8. Referências

Programação em C:

- http://www.dcc.ufla.br/~giacomin/Textos/tutorialc.pdf
- ftp://ftp.unicamp.br/pub/apoio/treinamentos/linguagens/c.pdf
- http://www.cs.cf.ac.uk/Dave/C/CE.html
- http://www2.its.strath.ac.uk/courses/c/
- http://www.lysator.liu.se/c/bwk-tutor.html

Uso do programa make e escrita de Makefiles:

- http://comp.ist.utl.pt/ec-aed/PDFs/make.pdf
- http://haxent.com.br/people/ruda/make.html
- http://informatica.hsw.uol.com.br/programacao-em-c16.htm
- http://www.gnu.org/software/make/manual/make.html
- http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/