Universidade Federal do Amazonas
Instituto de Computação
Disciplina: ICC 304 Comunicação sem Fio
Professor: Edjair Mota

Enunciado la Aula Prática Lab 02

Esta aula prática tem por objetivo a experimentação com redes tolerantes a atrasos e desconexões (DTN). Para uma melhor compreensão, as Redes Tolerantes a Atrasos e desconexões (Delay Tolerant Network - DTN) têm sido alvo de diversas pesquisas a fim de obter formas alternativas de montar uma rede totalmente funcional a um custo baixo. Em geral, uma pesquisa envolvendo DTN pode incluir formas de roteamento em uma rede ruidosa e temporariamente conexa até como utilizá-la para incluir digitalmente uma comunidade isolada como, por exemplo, as comunidades ribeirinhas da região Amazônica.

Até alguns anos atrás, computadores, roteadores e alguns equipamentos específicos eram utilizados como nós de uma DTN. Porém, nos últimos anos vem aumentando a popularidade de um mini computador chamado Raspberry Pi, com um sistema operacional embarcado e um poder de processamento que pode chegar a até 1.2GHz em alguns modelos. Portanto, o Raspberry Pi é uma ótima ferramenta para prototipar DTNs.

Resumidamente, a rede que será construída neste laboratório utiliza:

- Um Raspberry Pi modelo B+ como um nó da rede utilizando o Sistema Operacional Raspbian Wheezy com Kernel 4.1.19
- Uma estrutura Ad-Hoc
- Antenas USB com potência de 20 dbm que utilizam o protocolo 802.11bgn e transmitem no máximo 150MiB/s.
- Envio de pacotes utilizando TCP
- Faixa de endereços 192.168.0.0/24
- Berkeley Database 5.3 para armazenamento persistente
- TCL versão 8.5.19 para a interpretação de comandos.

1. Instalando o DTN2

O modelo do Raspberry Pi é o B+ com um sistema operacional Raspbian Wheezy instalado. O programa que será utilizado será o DTN2. Todos os comandos e roteiros aqui são editados e executados via terminal. Para começar, certifique-se que você possui alguma conexão com a internet, pois ela será necessária ao longo do tutorial. Inicialmente, iremos instalar alguns pré-requisitos do DTN2 diretamente do repositório. Antes, é necessário atualizar o repositório de programas do Raspberry Pi. Isso é feito pelos comandos:

sudo apt-get update

sudo apt-get -f install gcc-4.8 g++-4.8 libexpat1-dev libexpat1 expat xsdcxx libxerces-c2-dev libssl-dev hgsvn ecl bison cvs flex gperf texinfo automake libtool libncurses5-dev help2man gawk autoconf -y

Em seguida, é necessário obter algumas bibliotecas e alguns programas que não se encontram no repositório e são necessários para utilizar o DTN2. Agora iremos instalar o Linux-header, utilizado para compilar módulos externos de acordo com o kernel atual. Para verificar a versão do kernel atual, basta executar o comando:

uname -r

A versão usada neste laboratório é a 4.1.19+. Apesar de não ser obrigatório, se você quiser adquiri-la, execute o comando abaixo:

sudo rpi-update f406502f5628d32e6ca5dadac34ff7ca59f8e27f

O Raspberry Pi precisa ser reiniciado depois desse comando para trocar para o novo kernel. O Linux-header se encontra no link [1]. Baixe a versão que você precisa e instale utilizando o *dpkg*. Por exemplo:

wget https://www.niksula.hut.fi/~mhiienka/Rpi/linux-headers-rpi/linux-headers-4.1.19+_4.1.19+-2_armhf.deb

sudo dpkg -i linux-headers4.1.19+_4.1.19+-2_armhf.deb

Agora, é recomendado criar um diretório para concentrar os arquivos do DTN2 e seus demais pré-requisitos.

```
mkdir -p $HOME/insDTN2
cd $HOME/insDTN2
```

O próximo requisito que será instalado é o interpretador TCL. A versão utilizada aqui é a 8.5.19, que foi a versão mais alta que compilou sem erros. As demais versões são encontradas no link [2]. Ele é utilizado para ler os comandos feitos no *prompt* do DTN2 e também ler as configurações do arquivo *dtn.conf* que será descrito mais a frente. Os comandos a seguir baixam, descompactam, compilam e instalam o TCL.

```
wget https://sourceforge.net/projects/tcl/files/Tcl/8.5.19/tcl8.5.19-src.tar.gz
tar -xzvf tcl8.5.19-src.tar.gz
cd tcl8.5.19/unix
export CXX="g++-4.8"
export CC="gcc-4.8"
export ac_cv_func_strtod=yes
export tcl_cv_strtod_buggy=1
./configure
make
sudo make install
```

Com o fim da instalação do tcl, o próximo requisito é o banco de dados Berkeley, da Oracle. A versão mais alta que compilou sem erros foi a 5.3. As demais versões podem ser encontradas no link [3]. Para evitar inconveniências como a necessidade de registros e busca, é possível baixar a versão 5.3 diretamente do link [4]. O banco de dados Berkeley serve para manter os pacotes armazenados localmente por um longo período até sua transmissão ou até o seu tempo de vida acabar. Os comandos a seguir baixam, descompactam, compilam e instalam o Banco de Dados Berkeley 5.3:

```
cd $HOME/insDTN2
git clone https://www.github.com/MatheusOliveira2015/db/
```

cp db/db-5.3.28.tar.gz $PWD

tar -xzvf db-5.3.28.tar.gz

cd db-5.3.28/build_unix

../dist/configure

make

sudo make install

Por fim, o ultimo requisito para instalar o DTN2 é o oasys. O oasys é utilizado para informar aos arquivos de instalação do DTN2 quais são os programas e bibliotecas que serão utilizados pelo DTN2 e onde os mesmos estão localizados e suas respectivas versões. O comando usado a seguir baixa a versão mais recente do oasys, que no momento é a 1.6.0. Demais versões podem ser encontradas no link [5]. Os comandos a seguir baixam, compilam e instalam o oasys:

cd $HOME/insDTN2

hg clone http://hg.code.sf.net/p/dtn/oasys

cd oasys

export CXX="g++-4.8"

export CC="gcc-4.8"

./configure -C

make

sudo make install

Vale observar que o oasys será procurado dentro da pasta do DTN2 ou no diretório superior. Atente-se a isso mesmo que este tutorial já lhe garanta que o oasys estará no mesmo diretório que a pasta DTN2. Por fim, os comandos a seguir baixam, compilam e instalam o DTN2.

cd $HOME/insDTN2/

hg clone http://hg.code.sf.net/p/dtn/DTN2

cd DTN2

./configure -C

make

sudo make install

sudo cp $HOME/insDTN2/tcl8.5.19/unix/libtcl8.5.so /usr/lib/

sudo cp $HOME/insDTN2/db-5.3.28/build_unix/.libs/libdb-5.3.so /usr/lib/

Pronto, o DTN2 terminou de ser instalado. Para ver as aplicações DTN instaladas, você pode escrever no terminal *dtn* e apertar o TAB duas vezes para ver se aparecem várias opções como *dtnd, dtnping, dtnrecv, dtnperf-server,* etc. Para verificar se o DTN2 está funcionando, primeiro crie um banco de dados para os pacotes utilizando o comando:

sudo dtnd --init-db

e crie o processo *dtnd* com:

sudo dtnd

Se tudo deu certo, ele irá ao *prompt* do dtnd. Em seguida, abra um outro terminal na mesma máquina e execute:

dtnping localhost

Uma mensagem similar abaixo deve aparecer:

source_eid [dtn://raspberrypi.dtn/ping.2557]

dtn_register succeeded, regid 11

PING [dtn:// raspberrypi.dtn/ping] (expiration 30)...

20 bytes from [dtn:// raspberrypi.dtn/ping]: 'dtnping!' seqno=0, time=181 ms

20 bytes from [dtn:// raspberrypi.dtn/ping]: 'dtnping!' seqno=1, time=310 ms

Aperte CTRL+C para parar o *dtnping* e escreva *exit* no *prompt* do dtnd e aperte ENTER para encerrar o processo. Com isso, a instalação do DTN2 no Raspberry Pi está finalizada. Caso tenha acontecido algo errado, verifique com cuidado se você executou os passos corretamente. Caso o erro persista, entre em contato.

2. Configurando a rede Ad-Hoc

Vocês irão criar uma rede ad hoc, utilizando a sub-rede 192.168.0.0/24. Para configurar um Raspberry Pi a fim de se tornar um nó dessa rede, basta editar o arquivo */etc/network/interface* e resetar a interface modificada. Antes de qualquer alteração, é importante sempre ter um *backup* do arquivo modificado. Neste caso, basta executar:

sudo cp /etc/network/interfaces /etc/network/interfaces.backup

Ao abrir o arquivo */etc/network/interfaces*, você pode se deparar com um conteúdo similar ao texto da Figura 1:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
allow-hotplug ra0
iface ra0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Figura 1 – Conteúdo do arquivo /etc/network/interfaces

Em boa parte dos arquivos, a interface *ra0* não irá aparecer, pois ele é referente a uma interface com chipset Ralink 7601 que não possui um módulo nativo no Raspbian Wheezy. No Raspbian Jessie esse módulo já vem instalado, porém ele é identificado como *wlan0* também. Por conta disso, as configurações feitas na interface *ra0* podem ser feitas na interface *wlan0*, *wlan1*¸*etc*, se o mesmo oferecer serviços ad hoc.

A Figura 2 mostra como ficará a configuração do primeiro nó da rede. Como se pode perceber, o canal utilizado é o 1, o *ESSID* da rede é RPiDTN e o modo de operação, obviamente, é ad-hoc. Feche e salve o arquivo. Em cada Raspberry Pi da rede esse arquivo deve ser modificado como a Figura 2 variando apenas o IP, mantendo o IP dentro da sub-rede.

Após isso, a interface deve ser reiniciada executando:

sudo ifdown ra0

sudo ifup ra0

Para desfazer essas alterações, basta apagar o arquivo, recuperar o backup e reiniciar novamente a interface. Os comandos para isso são:

sudo cp -f /etc/network/interfaces.backup /etc/network/interfaces

sudo ifdown ra0

sudo ifup ra0

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
allow-hotplug ra0
iface ra0 inet static
        address 192.168.0.1
        netmask 255.255.255.0
        wireless-mode Ad-Hoc
        wireless-essid RPiDTN
        wireless-channel 1
iface default inet dhcp
```

Figura 2 – Conteúdo do arquivo /etc/network/interfaces modificado

Com isso, a configuração do nó Ad-Hoc está finalizada. Note que um nó não precisa de senha para se conectar na rede. Uma rede Ad-Hoc criptografada está fora do escopo deste laboratório.

3. Configurando o DTN2

Vamos começar concentrando todos os arquivos referentes à DTN em um diretório só. neste caso, */home/pi/dtn*. É necessário criar um diretório para o banco de dados Berkeley e outro diretório para guardar pacotes em trânsito na rede (*payload*). Os comandos para isso são:

```
sudo mkdir -p /home/pi/dtn                #pasta contendo arquivos DTN
sudo mkdir -p /home/pi/dtn/bundles    #pacotes em payload
sudo mkdir -p /home/pi/dtn/db          #Banco de Dados Berkeley
```

Agora, copiar o arquivo */etc/dtn.conf* para a pasta */home/pi/dtn/dtn.conf*:

```
sudo cp -f /etc/dtn.conf  /home/pi/dtn
```

O arquivo *dtn.conf* contém todas as configurações do nó local da rede DTN. Como já comentado, cada linha desse arquivo, com excessão de linhas começadas com o caractere '#', é interpretada pelo tcl e manda o comando para o processo *dtnd*. Antes de começar as edições, considere que a pequena rede Ad-Hoc DTN segue o modelo apresentado na Figura 3.

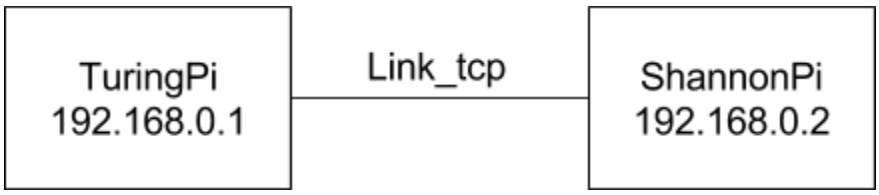

Figura 3 – DTN Ad-Hoc com 2 nós

É hora de editar o arquivo */home/pi/dtn/dtn.conf*. Uma dica: utilize o *nano* no terminal para editar o */home/pi/dtn/dtn.conf* e utilize o CTRL+W para encontrar os comandos abaixo. Vamos começar setando os diretórios para *payload* e o Banco de Dados Berkeley. Encontre a linha:

```
storage set dbdir $dbdir/db
```

e troque por

```
storage set dbdir /home/pi/dtn/db
```

Encontre a linha:

storage set payloaddir $dbdir/bundles

e troque por

storage set payloaddir /home/pi/dtn/bundles

Agora é hora de definir um EID para identificar os pontos finais da DTN. Seguindo o modelo da Figura 3, temos apenas 2 EIDs: TuringPi e p ShannonPi. Encontre a linha:

route local_eid "dtn://[info hostname].dtn"

Vale notar que o comando em colchetes *info hostname* pega o *hostname* do Raspberry Pi local, que por padrão é *raspberrypi,* e coloca no lugar. Então se deixarmos essa linha nessa forma, o EID desse nó ficará *dtn://raspberrypi.dtn*. Se você mudar o *hostname* desse Raspberry Pi, não será necessário mudar essa linha. Mas no caso deste tutorial, nós mudaremos o EID diretamente nesse arquivo. Então em cada dispositivo essa linha ficará:

No RPi 192.168.0.1 → *route local_eid "dtn://TuringPi.dtn"*

No RPi 192.168.0.2 → *route local_eid "dtn://ShannonPi.dtn"*

Novamente seguindo o modelo da Figura 3, é preciso criar os *links* em cada nó da rede. No geral, para criar um *link* adicione a linha abaixo:

link add NomeDoLink IPDestino TipoDeLink CamadaDeConvergência

No RPi 192.168.0.1 adicione a linha:

link add link_tcp 192.168.0.2 ALWAYSON *tcp*

No RP2 192.168.0.2 adicione a linha:

link add link_tcp 192.168.0.1 ALWAYSON *tcp*

Na seção Referências → Configuração → Comandos Links → *"link add"* do *link* [6] você encontra mais detalhes sobre os parâmetros do *link add*.

Para o *link* ser percebido pelos demais nós, é necessário que cada nó crie um agente para anunciar que o nó local foi ligado à DTN. Para isso adicione as linhas abaixo em cada Raspberry Pi:

discovery add NomeDoAgente ip local_addr=IPLocal port=PortaOrigem/Destino unicast=false continue_on_error=true

discovery announce NomeDoLink NomeDoAgente tcp cl_addr=IPLocal cl_port= PortaLocal interval=1

Por exemplo, no RPi 192.168.0.1:

discovery add AgenteTuring ip local_addr=192.168.0.1 port=9556 unicast=false continue_on_error=true

discovery announce link_tcp AgenteTuring tcp cl_addr=192.168.0.1 cl_port=9557 interval=1

Por exemplo, no RPi 192.168.0.2:

discovery add AgenteShannon ip local_addr=192.168.0.2 port=9556 unicast=false continue_on_error=true

discovery announce link_tcp AgenteShannon tcp cl_addr=192.168.0.2 cl_port=9557 interval=1

Verifique se a porta 9556 e a 9557 não estão sendo utilizadas por outro processo. Outro cuidado que você deve tomar é que não pode haver *links* com nomes duplicados. Se algum nó da rede trocar de IP, crie um novo *link* para substituir os *links* conectados a ele.

Precisamos adicionar uma rota final para que os pacotes sejam enviados adicionando a linha:

route add eid_destino/ NomeDoLink*

No RPi 192.168.0.1:

route add dtn://TuringPi.dtn/ link_tcp*

No RPi 192.168.0.2:

route add dtn://ShannonPi.dtn/ link_tcp*

Agora as aplicações DTN sabem qual *link* utilizar para enviar seus pacotes. Não se esqueça do caractere '*' no final do EID, pois é ele vale como um coringa para toda palavra depois do caractere '/'.

Por fim, é necessário que todos os nós DTN estejam sincronizados, pois o período de expiração dos pacotes se refere a sua data de criação e a data no Raspberry Pi. Certifique-se que você possui uma conexão com a internet e execute em cada Raspberry Pi o comando:

sudo ntptime

Com isso terminamos de editar o */home/pi/dtn/dtn.conf* . Essas modificações são o suficiente para os dispositivos se comunicarem na rede. Para testar a comunicação execute os comandos no RPi 192.168.0.1 :

sudo dtnd -c /home/pi/dtn/dtn.conf --init-db

sudo dtnd -c /home/pi/dtn/dtn.conf

No RPi 192.168.0.2 execute os mesmos comandos e abra um segundo terminal e execute:

dtnping dtn://TuringPi.dtn/ping

Se os pacotes foram recebidos pelo Turing Pi então você está com uma comunicação funcional. Realize para todos os nós que você criar. Para mais detalhes sobre comandos TCL para o *dtn.conf* visite o link [6] na seção de Referência→ Configuração.

4. Aplicações do DTN2

4.1 dtnd

O *dtnd* é o processo principal que controla os outros programas da DTN. Ao executá-lo sem parâmetros, ele utiliza a configuração padrão definida no arquivo */etc/dtn.conf* . Se a opção –d for omitida, ele inicia um *prompt* que permite que você execute os comandos TCL para configurar o seu nó DTN.

Uso: dtnd [opção]

[opção]:

-h = Mostra todas as opções

-d = Executa o dtnd como um *daemon*. Deve ser usado com o -o

-c <arquivo.conf> = o *dtnd* utiliza a configuração do arquivo <arquivo.conf>.

-t = Limpa o Banco de Dados.

-o <arquivo.log> = grava a saída do *dtnd* no <arquivo.log>.

--init-db = Cria um Banco de Dados no local definido pelo <arquivo.conf>

Ex:

```
#Roda o dtnd como daemon e registra a saída no dtn.log
sudo dtnd -d -o dtn.log

#Cria um Banco de Dados na pasta definida no /home/pi/dtn.conf
sudo dtnd -c /home/pi/dtn.conf --init-db
```

Note que a primeira vez que você rodar o *dtnd* , ele deve vir com a opção --init-db para criar o Banco de Dados. Se você iniciar o *dtnd* com a opção –d, ele só pode ser encerrado utilizando algum comando do terminal. Uma maneira de contornar isso é executar o comando

```
ps ax|grep dtnd
```

A saída deve ser algo assim:

```
2690 ?        Ssl    0:02 dtnd -c dtn.conf -d -o dtn.log
 2712 pts/0    S+     0:00 grep --color=auto dtnd
```

Para encerrar o *daemon*, basta executar

```
sudo kill 2690
```

Para sair do *dtnd* no *prompt*, basta digitar *exit* e apertar ENTER ou apertar CTRL+C

4.2 dtnping

O *dtnping* é mais simples que o *ping* tradicional.

Uso: *dtnping [-c count] [-i interval] [-e expiration] eid*

-h = Mostra todas as opções

-c count = Envia *count* pacotes

-i interval = Envia os pacotes em intervalos de *interval* segundos

-e expiration = Os pacotes enviados expiram em *expiration* segundos

eid/ping = EID do destino

Ex:

#Pinga o EID TuringPi
dtnping dtn://TuringPi.dtn/ping

#Pinga o EID TuringPi em intervalos de 3 segundos
dtnping -i 3 dtn://TuringPi.dtn/ping

4.3 dtnperf-server e dtnperf-client

O *dtnperf-server* e *dtnperf-client* criam um fluxo de dados entre dois nós da DTN. A camada de transporte utilizada depende do link entre os nós definida no *dtn.conf*.

Uso: *dtnperf-server* [opção]

Opção:

-h = Mostra todas as opções

-d <*dir*> = Coloca o tráfego recebido na pasta <*dir*>

-L <*file.log*>= Cria um log <*file.log*>

Uso: *dtnperf-client -d* <*dest_eid*> *[-t* <*sec*> | *-n* <*num*>*] [Opção]*

-d <*dest_eid*> = cria um fluxo com o *dest_eid* que deve estar rodando o servidor

-t <*sec*> = Mantém o fluxo por <sec> segundos

-n <*num*> = Envia um tráfego <num [BKM]> bytes.

[Opção]

-h = Mostra todas as opções

-L <*file.log*> = Cria um log <*file.log*>

-p <*num*> = Envia agregados de <*num*[BKM]> bytes

-w <*windows*> = Faz com que o tráfego mantenha até no máximo <windows> agregados em *payload*.

Ex:

#Envia 30KBytes para TuringPi do ShannonPi com log

#No TuringPi
dtnperf-server

#No ShannonPi
dtnperf-client -d dtn://TuringPi.dtn -n 30K -L dtnperf.log

#Gera um tráfego durante 10 segundos entre o TuringPi do ShannonPi

#No TuringPi
dtnperf-server

#No ShannonPi
dtnperf-client -d dtn://TuringPi.dtn -t 10

Para encerrar o *dtnperf-server*, aperte CTRL+C.

5. Experimento

5.1 Objetivo geral

Estimar a capacidade de transmissão de agregados em DTNs

5.2 Escopo do Experimento

O experimento consiste em fazer medições de vazão útil de agregados e potência recebida (RSSI) entre duas instâncias DTN2, uma aplicação DTN de referência, executadas em Raspberries Pi conectados utilizando redes sem fio. A Comunicação funciona seguindo o modelo da Figura 4.

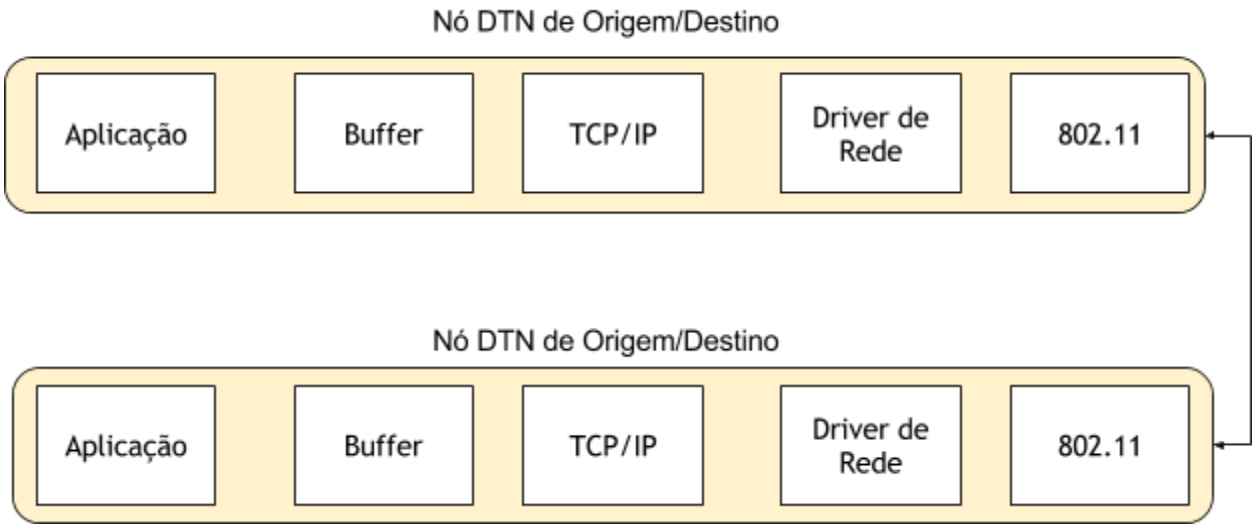

Figura 4 - Modelo de Conectividade

5.3 Experimento - Vazão x Tamanho do Agregado e Vazão x Potência

Este experimento consiste em manter um tráfego entre os Raspberries a fim de obter a vazão da rede variando o tamanho de cada agregado:

1. Configure o arquivo *dtn.conf* e inicie seu *dtnd* em cada RPi conforme mostrado nos exemplos da seção 4. Recomenda-se não utilizar a opção *-d*. Dessa forma, você pode visualizar o *prompt* do *dtnd*.

2. Inicie no RPi Servidor o dtnperf-server
    a. *sudo dtnperf-server -D*

3. Mantenha o RPi cliente a uma distância de:
    a. 5m
    b. 10m
    c. 15m

4. Utilize o comando *iwlist* para obter a potência do recebida do RPi Cliente
    a. *iwlist ra0 scan |grep Signal*

5. Inicie no RPi Cliente o dtnperf-client com o comando

    a. *sudo dtnperf-client -d eid_dest -n payload -p payload -e 10*

6. Repita o passo 5 variando o valor do *payload* com os seguintes valores
    a. 50K
    b. 100K
    c. 250K
    d. 500K
    e. 1M
    f. 2,5M
    g. 10M

7. Repita cada teste 5 vezes a fim de obter uma média para cada distância e tamanho de agregado.

8. Os dados finais estarão contidos todos no RPi Cliente. Obtenha a vazão de cada experimento na região que mostra o GOODPUT.

9. Caso você perceba que o programa demora excessivamente a parar, olhe o *prompt* do *dtnd* e veja se aparece um aviso que a quantidade de agregados em *payload* excedeu o limite do Banco de dados. Neste caso, pressione CTRL+C em ambos RPi, limpe o banco de dados e resete-o:
    a. *cd /home/pi/dtn*
    b. *sudo rm db/* bundles/**
    c. *sudo dtnd -c dtn.conf --init-db*

5.4 Resultados

Os resultados devem ser apresentados em forma de gráficos como os exemplos mostrados nas Figuras 5 e 6.

Lembrem-se de apresentar o que cada eixo do gráfico representa e suas respectivas unidades de medidas. Discuta os resultados, faça uma análise, proponha experimentos adicionais que poderiam melhorar a vazao.

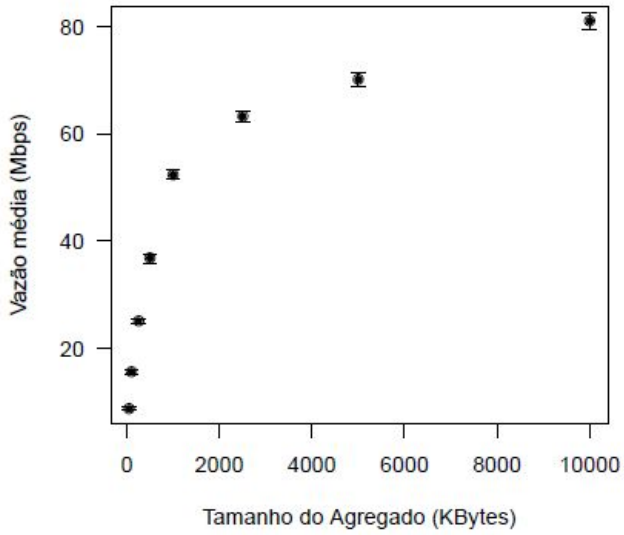

Figura 5 - Vazão medida em função do tamanho dos agregados

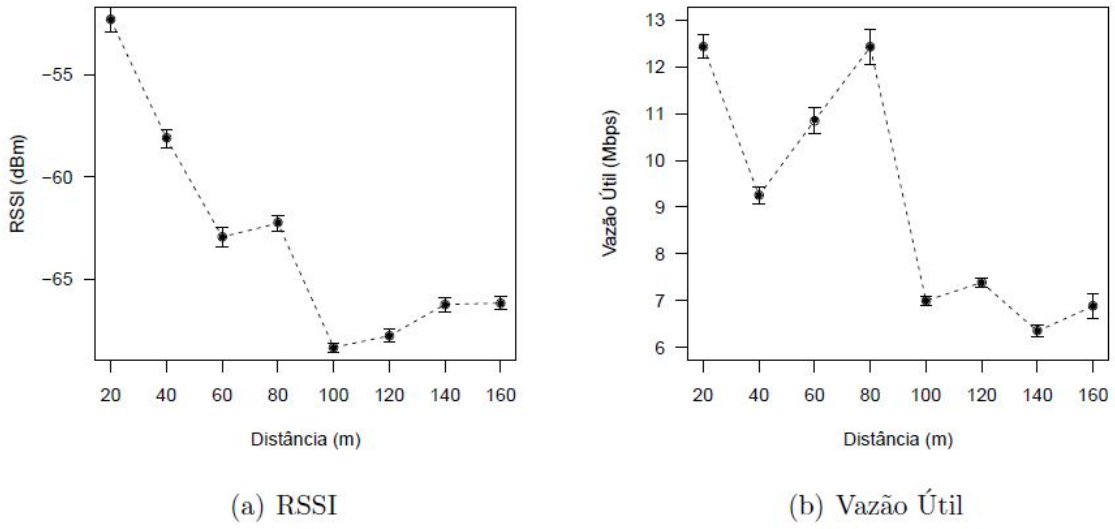

Figura 6 - Exemplos de medições da potência e da distância