



## Universidade Federal do Amazonas Instituto de Computação

### IEC684– SIMULACAO DE SISTEMAS

DANNY FABIANO FARIAS COSTA	20901592
JULIA CAROLINE DA SILVA PACHECO	20901704
MARCOS ANTONIO BATISTA DE OLIVEIRA	20901561
RAYLSON GAMA BRANDAO	20901792
RODRIGO LEITE COELHO	20410148

## TRABALHO PARCIAL II

### 1. Introdução

O presente trabalho descreve o modelo de mobilidade com o simulador para redes oportunistas ONE, implementando e avaliando políticas de gerência de buffer para redes tolerantes a atrasos e desconexões utilizando três modelos de mobilidade.

O modo como as redes são desenvolvidas nos últimos anos tem evoluído sobre diversos aspectos. Estas redes ditas do futuro possuem uma gama de características que não tinham relevância no modo clássico como as redes eram concebidas, como interrupção sucessiva ou mobilidade variada, por exemplo.

### 2. Arquitetura DTN

#### Redes Regionais

Na arquitetura DTN é usado o conceito de regiões que define a área de atuação de redes com características de comunicação homogêneas. A rede DTN é construída como uma rede *overlay* em cima das redes regionais, que são redes diferentes, com características distintas. A rede DTN provê interoperabilidade entre as redes regionais, servindo como uma espécie de *buffer* para acomodar as diferentes latências entre as redes e possíveis falta de conectividade. Com isso a DTN também pode resolver os problemas de mobilidade e bateria limitada de dispositivos sem-fio.

#### Comutação de Mensagens

A comutação de mensagens não foi desenvolvida agora, ela é uma das formas mais simples de se trocar mensagens e é usada até hoje em muitos lugares como, serviços de correio, e-mail e correio de voz. Uma mensagem numa rede DTN pode ser um bloco de inteiro de dados de um aplicativo ou um segmento dele. A comutação de mensagens da rede DTN é do tipo *store-and-forward*, ou seja, ela primeiro recebe toda a mensagem, guarda em um *buffer* e depois encaminha para o próximo nó ou destino. A figura abaixo ilustra melhor esse processo.

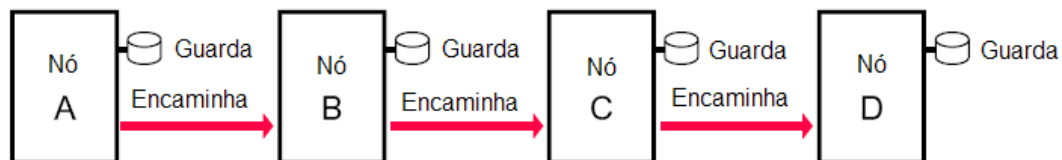


Figura 1: Comutação de Mensagens Store-and-Foward (\*)

A comutação de mensagens embora seja muito simples, exige que os responsáveis por encaminhar as mensagens - os roteadores e *gateways* - tenham de possuir grande capacidade de armazenamento. Isso porque as mensagens têm que ser armazenadas por completo até que seja possível encaminhá-las. Esse tipo de armazenamento é dito persistente em oposição ao armazenamento feito em roteadores convencionais da Internet, que guardam os pacotes em *chips* de memória de curto prazo, por tempos da ordem de milissegundos, até que os pacotes sejam encaminhados. Os roteadores DTN precisam de armazenamento persistente pelos seguintes motivos:

- A comunicação com o próximo salto pode ficar indisponível por um grande período de tempo.
- Um dos nós da comunicação pode enviar dados muito mais rápido que o outro.
- O receptor da mensagem encaminhada pode detectar um erro e pedir a retransmissão dela.

Por armazenar as mensagens por inteiro, a técnica de comutação de mensagens pode avisar ao próximo salto os requisitos de banda e tamanho livre necessário em memória para armazenar a mensagem nos nós intermediários.

### Contatos Oportunistas

Contatos oportunista são contatos que acontecem sem estarem previamente combinados. Suponha que uma pessoa deseje falar algo com outra, mas por algum motivo não consegue se comunicar com ela. Se, por sorte essas pessoas se encontrarem fisicamente no centro da cidade, elas vão ter a oportunidade de se comunicarem, e como existe o interesse de uma delas de fazê-lo, então ela o fará. Isto é considerado um contato oportunista, já que nenhuma das pessoas havia marcado um encontro com a outra. Esse é um exemplo que pode acontecer entre seres humanos, no entanto, hoje em dia está cada vez mais comum este

tipo de contato acontecer entre dispositivos eletrônicos sem-fio. Imagine que uma pessoa programou seu PDA (*Personal Digital Assistant*) para fazer o *download* de um arquivo. Porém no momento em que o indivíduo programou o seu PDA ele não tinha como acessar a Internet. Dessa maneira, uma vez que o PDA está programado, assim que ele tiver a oportunidade de se conectar com um ponto de acesso e baixar o arquivo, ele o fará. A figura abaixo ilustra outros tipos de contato oportunistas entre dispositivos móveis.

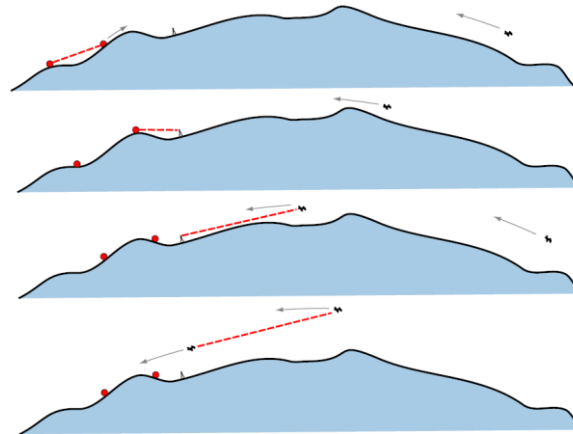


Figura 2: Contatos Oportunistas (\*)

### Funcionamento DTN

A partir de agora será mostrado o funcionamento das redes DTN. O tipo de mais básico de transmissão de mensagens numa rede DTN é o envio de mensagem sem a necessidade de qualquer reconhecimento ou classe de serviço. A fonte apenas envia a sua mensagem para o próximo nó. Outros tipo de classe de serviço são:

- Transferência em Custódia
- Confirmação de recepção da mensagem - confirmação para a fonte ou responsável pela retransmissão da mensagem, de que a mensagem foi recebida pela aplicação de destino.
- Notificação de aceitação de custódia - notificação para a fonte ou responsável pela retransmissão da mensagem, toda vez que um nó aceitar ser o novo custodiante de uma mensagem.
- Notificação de encaminhamento - notificação para a fonte ou responsável pela retransmissão da mensagem, toda vez que a mensagem for encaminhada.
- Prioridade de entrega - muito volume de dados (pouca prioridade), normal (média prioridade) e importantes (alta prioridade).
- Autenticação - Indica o método de autenticação, se for especificado algum (*e.g.* assinatura digital) este será usado para verificar a autenticidade da fonte e a integridade dos dados.

Para resolver o problema de diferença entre as latências das regiões, as redes DTN realizam a terminação dos protocolos de transporte, que ocorre na camada de empacotamento dos roteadores e *gateways*. Já que os protocolos de transporte (*e.g.*, TCP) são os responsáveis pela garantia da confiabilidade na comunicação fim-a-fim, e é a confiabilidade na comunicação, exatamente um dos desafios encontrados nos ambientes aonde as redes DTN se propõem a situar. Com isso, a camada de empacotamento age como substituta da camada de transporte na função de garantir a comunicação fim-a-fim. Essa medida, de terminar os protocolos de transporte, faz com que as camadas inferiores à camada de empacotamento que se situam em redes de baixa latência fiquem isoladas de seus pares homólogos em outras redes com alta latência. Resolvendo assim o problema de atraso.

A camada de empacotamento suporta sozinha a troca de mensagens fim-a-fim. As mensagens são tipicamente entregues separadamente, de um nó para o próximo, independentemente das outras mensagens, exceto mensagens de resposta opcionais. Mesmo que as mensagens sejam quebradas em fragmentos, como já foi dito acima que pode acontecer, elas viajam independentemente uma da outra.

### 3. Implementação das políticas de gerência

A primeira parte do trabalho era a construção dos algoritmos de descarte de mensagens no ONE. As políticas especificadas para o grupo foram:

- a) Evict Most forwarded first – A mensagem que foi enviada um número máximo de vezes é descartada;
- b) Drop tail – A primeira mensagem na fila do buffer é escolhida para ser retirada.

#### 3.1. Algoritmos

##### DropTail

```
01  protected Message getDropTailMessage(boolean excludeMsgBeingSent) {
02      Collection<Message> messages = this.getMessageCollection();
03      Message first = null;
04      Object[] queueMessages = messages.toArray();
05      for (int i=0; i<queueMessages.length; i++) {
06
07          if (excludeMsgBeingSent && isSending(((Message)
08              queueMessages[i]).getId())) {
09              continue; // skip the message(s) that router is sending
10          }
11          else{
12              first = (Message)queueMessages[i];
13              break;
14          }
15      }
```

```

16         return first;
17     }

```

Este algoritmo retorna a primeira mensagem na fila do buffer. O problema que tivemos de imediato com ele é que a fila de mensagens do ONE é uma variável da classe `Collection` de `Message`, que composta por vários elementos, mas sem serem indexados. A ideia do algoritmo é justamente pegar a mensagem de índice 0, dentro da coleção. Implementamos isso, transferindo as referências dos objetos `Message` para para um `Array` de `Objects`. Fazendo isso, com um método da própria classe `Collection` (`toArray()`), conseguimos transformar a fila de mensagens num vetor e pegar a posição 0 do mesmo. Uma consideração importante deve ser destacada: a linha 07 testa se a mensagem começou a ser enviada, ou seja, caso o algoritmo selecione a primeira mensagem e ela está sendo enviada, o mesmo a ignora e condena a segunda ou a terceira ou a quarta mensagem presente na fila, e assim por diante, para ser descartada.

### Evict Most forwarded first

```

01     protected Message getEvictMostForwardedFirstMessage(boolean
           excludeMsgBeingSent) {
02         Collection<Message> messages = this.getMessageCollection();
03         Message mostForwarded = null;
04         int i;
05         for (Message m : messages) {
06
07             if (excludeMsgBeingSent && isSending(m.getId())) {
08                 continue; // skip the message(s) that router is sending
09             }
10             else if (mostForwarded == null ) {
11                 mostForwarded = m;
12             }
13             else if (m.getForwardCount() >= Message.MAX_FORWARDED) {
14                 mostForwarded = m;
15                 break;
16             }
17             else if (mostForwarded.getForwardCount() <
m.getForwardCount()) {
18                 mostForwarded = m;
19             }
20         }
21     }
22
23     return mostForwarded;
24 }

```

Neste tivemos que pensar um pouco mais. Vamos ao que o algoritmo propõe: retorna a mensagem presente no buffer que tiver o maior número de envios registrados. Para isso tivemos que modificar mais classes além da `ActiveRouter`, porque precisávamos guardar e manipular a quantidade de vezes a mensagem fora enviada. Portanto tomamos uma decisão de projeto: modificar a classe `Message`, incluindo os seguintes membros:

```

01     /** Máximo número de vezes que uma mensagem é enviada */
02     public static final int MAX_FORWARDED = 10;
03     /** Número que vezes que a mensagem foi enviada */

```

```
04     private int forwardCount;
05     /** Retorna o número de envios atual */
06     public int getForwardCount() {
07         return forwardCount;
08     }
09     /** Atribui um valor ao número de envios, caso necessário */
10     public void setForwardCount(int forwardCount) {
11         this.forwardCount = forwardCount;
12     }
13
14     /** Chamada de método para incrementar o número de envios
15      * quando do início do processo de envio de mensagem
16      */
17     public void incForwardCount() {
18         this.forwardCount++;
19     }
```

O algoritmo funciona testando cada uma das mensagens presentes no buffer. A primeira mensagem encontrada é condenada na primeira iteração, por um simples detalhe de lógica, para que haja com o que comparar as demais. A partir da segunda iteração, o algoritmo passa a testar se o número de envios das mensagens se iguala ou ultrapassa à constante MAX\_FORWARDED, membro da classe Message e que define o número máximo de envios até que a mensagem seja descartada. A primeira mensagem encontrada que satisfaça esta condição é escolhida pelo algoritmo. Caso ele não encontre, a mensagem de maior número é descartada, ou seja, a que tiver número de envios mais próximo do máximo. Especificamos a constante MAX\_FORWARDED como sendo igual a 10, pelo fato de que, como são muitas mensagens que figuram num mesmo instante no simulador (apesar de este número ter variação muito alta), é provável o algoritmo de envio dará menos atenção aos elementos do buffer, porque são muitos, logo o número de envios de cada elemento (mensagem) será reduzido, assim: —————. O denominador desta razão não é maior que o numerador, portanto teremos um número racional que será discretizado se tornando inteiro.

## 4. Modelo de Mobilidade

A segunda parte do trabalho é a definição do modelo de mobilidade. O nosso problema real é: em uma rede DTN, quando ocorrer overflow no buffer dos nós qual mensagem será escolhida para ser descartada.

### My Tracks

O My Tracks é um aplicativo que permite gravar e compartilhar trilhas de GPS, com estatísticas. Ele grava seus caminhos com o GPS e mostra estatísticas em tempo real enquanto você caminha, pedala, corre ou participa de outras atividades ao ar livre. Os caminhos podem ser compartilhados, enviados ao Google Planilhas, Google Meus Mapas ou Twitter. (Android Market, 2011)

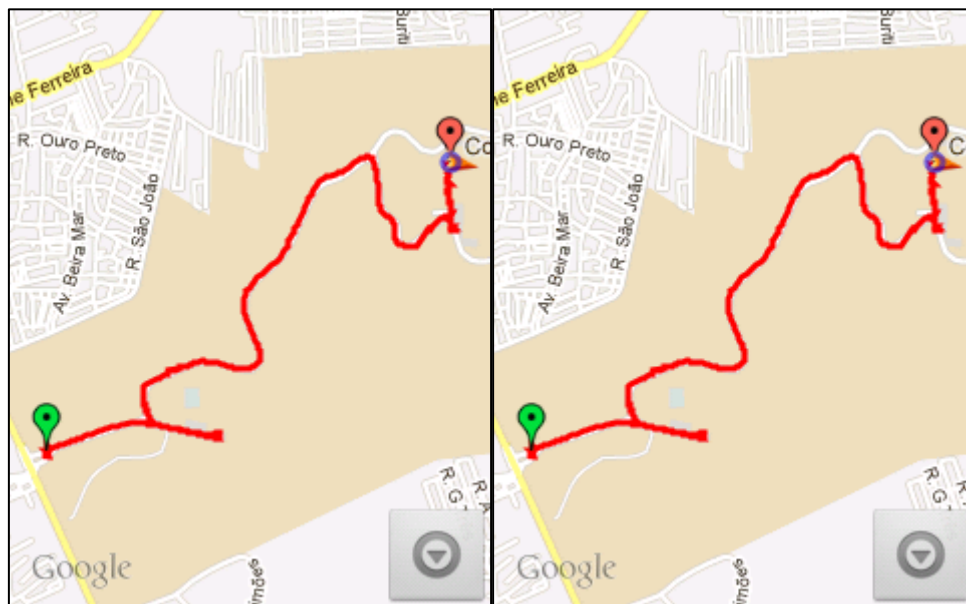


Figura 3. Da esquerda pra direita: percurso gravado no GPS pelo MyTracks, Altimetria e Estatísticas do percurso (dados da Android Market, 2011).

#### Informações sobre o modelo:

- O ambiente de simulação que escolhemos foi o campus da Universidade Federal do Amazonas;
- Coletamos dados com o aplicativo My Tracks durante 8 dias úteis. Esses dados foram utilizados para compor um trace de mobilidade que será usado como parâmetro de entrada no simulador ONE;
- Ao todo foram realizadas 25 coletas;

A seguir algumas imagens das coletas:





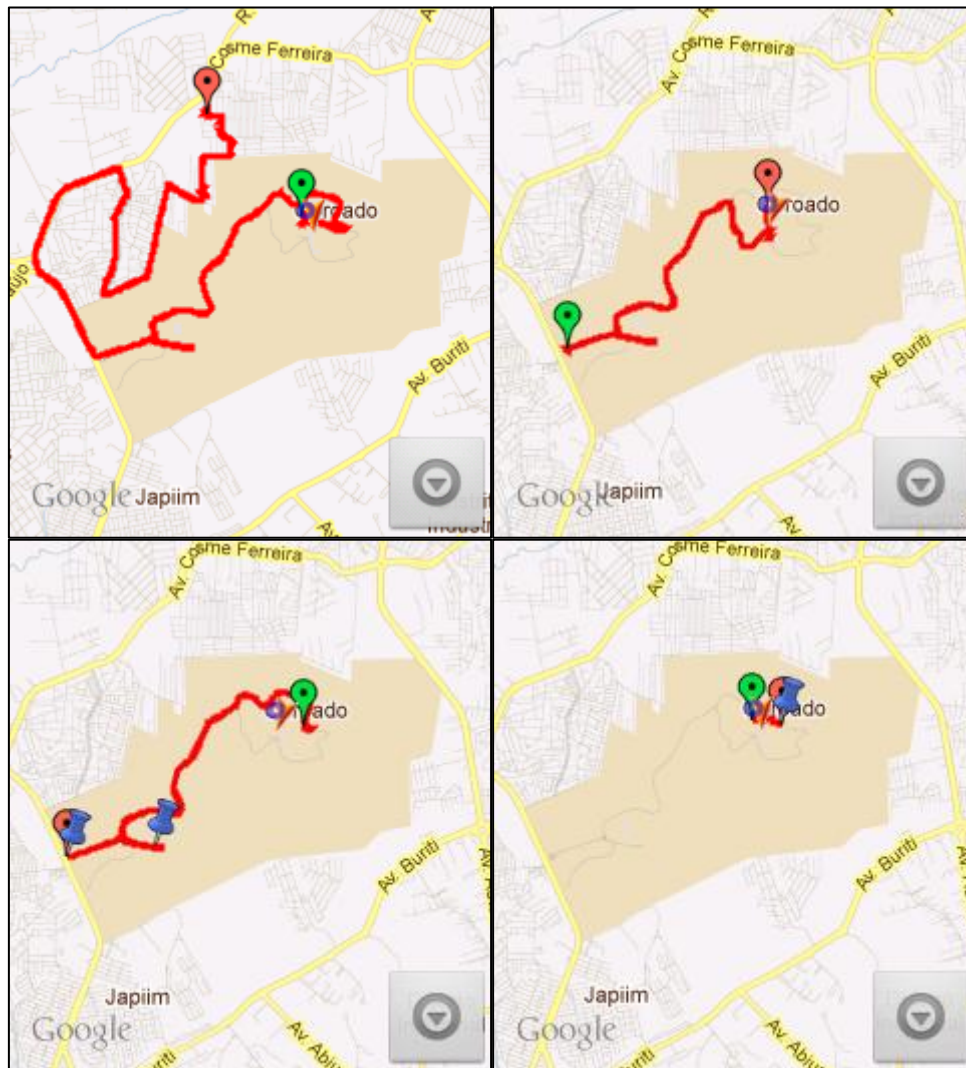


Figura 4: Screenshots do trace de seis das vinte e cinco coletas realizadas.

Parâmetros de configuração do simulador:

```
01 # Common settings for all groups
02 Group.movementModel = ExternalMovement
03 ExternalMovement.file = TrabalhoParcial2_MARCOS.ANTONIO.gpx.one
04 Group.router = EpidemicRouter
05 Group.bufferSize = 5M
```

## 5. Aplicação de V&V ao modelo

Duas fases muito delicadas no processo de desenvolvimento de modelos de simulação correspondem à verificação e à validação desses modelos.

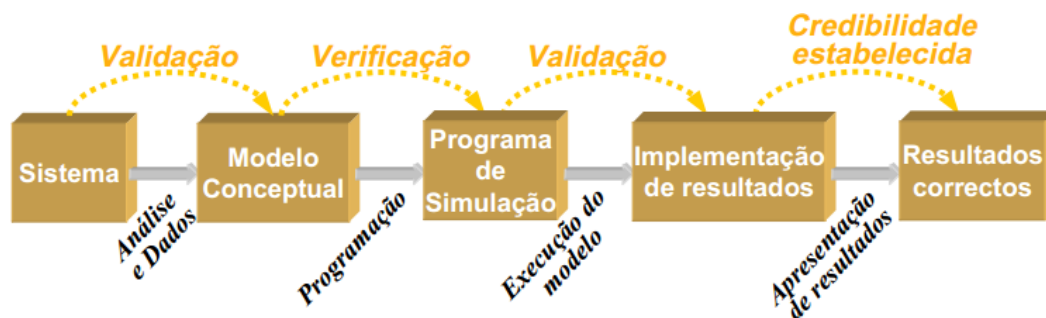
Verificação do modelo é a etapa onde o modelador checa se o modelo desenvolvido corresponde ao idealizado. Neste ponto é verificado se o modelo foi construído corretamente. Nesta fase busca-se fazer testes exaustivos no simulador. O modelador precisa se convencer de que o simulador está correto e rodando bem.



Na verificação de um modelo, deve-se variar os valores dos parâmetros de entrada (inclusive utilizando as fronteiras do intervalo de valores) e analisar se os resultados são coerentes. Nesse momento é possível sentir a força da simulação, que agiliza a etapa de testes, tornando-os viáveis em termos de tempo e dinheiro.

A validação é a etapa onde será checado se o modelo desenvolvido representa bem o sistema real. É a busca da resposta para a pergunta: foi desenvolvido o modelo correto?. A ideia é passar confiança ao usuário, mostrando que qualquer experimento com o modelo irá gerar resultados que coadunam com a realidade do sistema estudado.

A validação é normalmente conseguida executando o modelo e comparando seus resultados com os oriundos do sistema real. Se os resultados da simulação se aproximarem dos valores reais, dentro de um nível de confiança desejado, o simulador será validado.



Implementamos verificação com três técnicas:

Técnica 1: Escrever e debugar em módulos e submódulos;

Técnica 2: Utilizar mais de uma pessoa para revisar o código, pois aquele que codificou um subprograma ou submódulo não é um bom crítico;

Técnica 7: (Análise do fluxo de dados) Pode ser usada para descobrir anomalias do código do modelo, tais como variáveis indefinidas ou não referenciadas.

## 6. Uso das ferramentas

### My Tracks

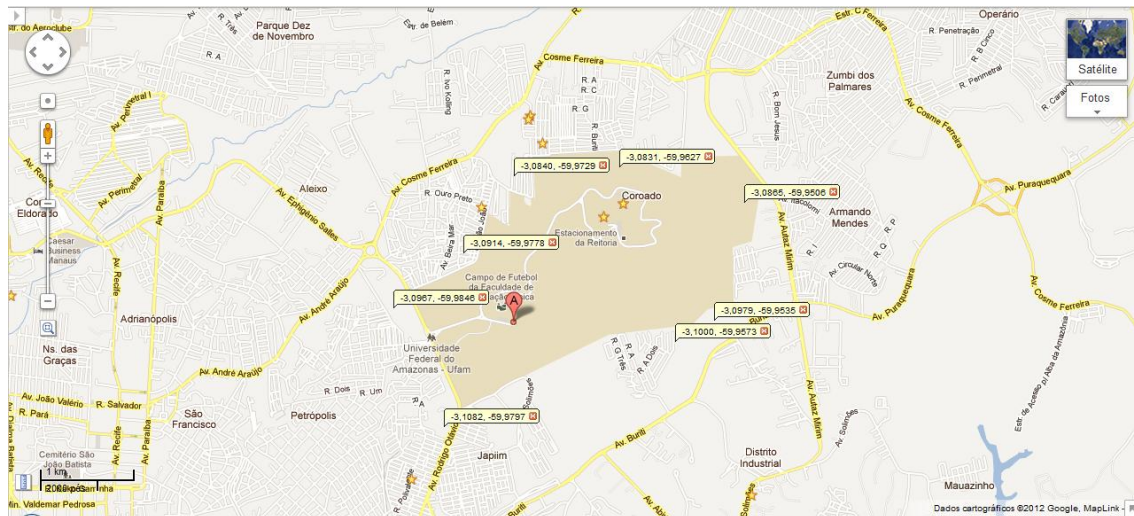
Com a utilização do my tracks não houve nenhum problema. Talvez o único problema que mereça ser citado seja que enquanto o programa estava em execução, a bateria do smartphone se esgotava rapidamente. Gerando problema para dois dos nossos colegas que instalaram o software em seus celulares pessoais.

### BonMotion

Seguindo o tutorial disponibilizado pelo monitor, não houve muito mistério em conseguir chegar a um resultado.

A principio houve estranheza ao utilizar algum software que desse as coordenadas (usado google maps), pois o foco do software não era mostrar coordenadas exatas, mais sim, nome de ruas, locais

próximos, entre outros. Para conseguir as coordenadas foi necessário utilizar algumas “gambiarras”. Como mostrado abaixo:



Apesar de não esta, no mapa, marcando exatamente o perímetro da ufam, as coordenadas mostradas são de fato algumas pontos que marcam a fronteira da ufam.

Uma vez coletado os pontos, bastou pegar a maior latitude e longitude e a menor latitude e longitude.

Com os dados escolhidos foi possível montar a instrução que se faltava:

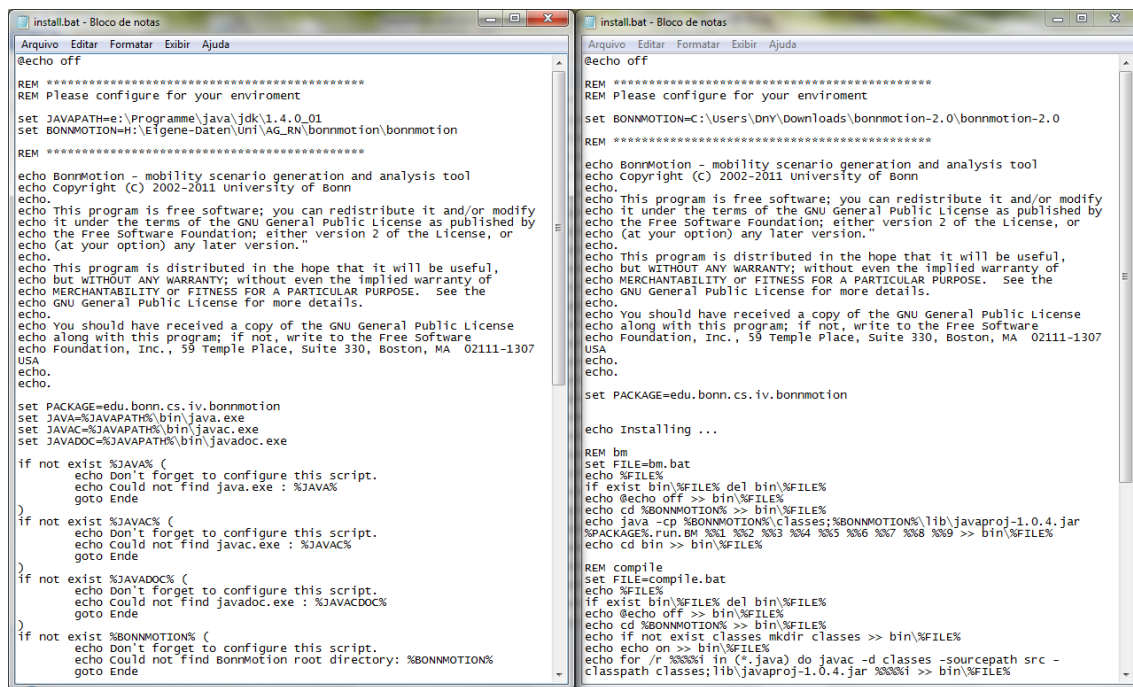
```
<bounds minlat="-3.1082" minlon="-59.9845" maxlat="-3.0836" maxlon="-59.9507"/>
```

Quando terminado de colocar a instrução acima em todas as coletas. Bastou seguir o manual e conseguimos gerar o arquivos de saída com extensão “.one”.

Gerado os primeiros resultados, notou-se que seria muito custoso criar um programa para fazer o que era pedido. Para facilitar a vida decidi instalar o bonmontion no Windows, pois é de conhecimento próprio a manipulação de arquivos usando resquício do DOS (linguagem batch).

A primeiro momento foi necessário instalar o JDK e coloca-lo na variáveis de sistema.

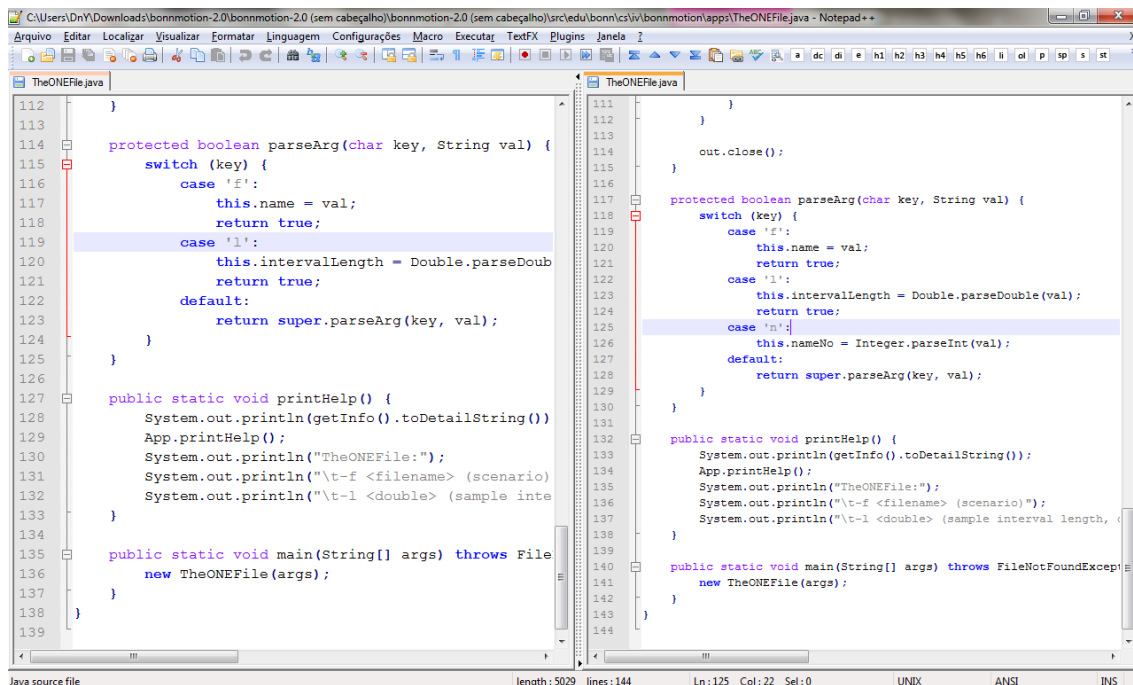
Quando tentei rodar pela primeira vez o bonmontion, notei q ele exigia que modificasse o arquivo install.bat. Mas só realizando as modificações que o tutorial próprio do bonmontion sugeria não era suficiente. Então foi modificado parte do arquivo para que pudesse ser compilado na minha maquina. Como mostrado a seguir:



Como era (imagem esquerda) e como ficou(imagem direita).

Uma vez compilado e rodando, decidi modificar o arquivo fonte para que o programa facilitasse na hora de ordenar os arquivos.

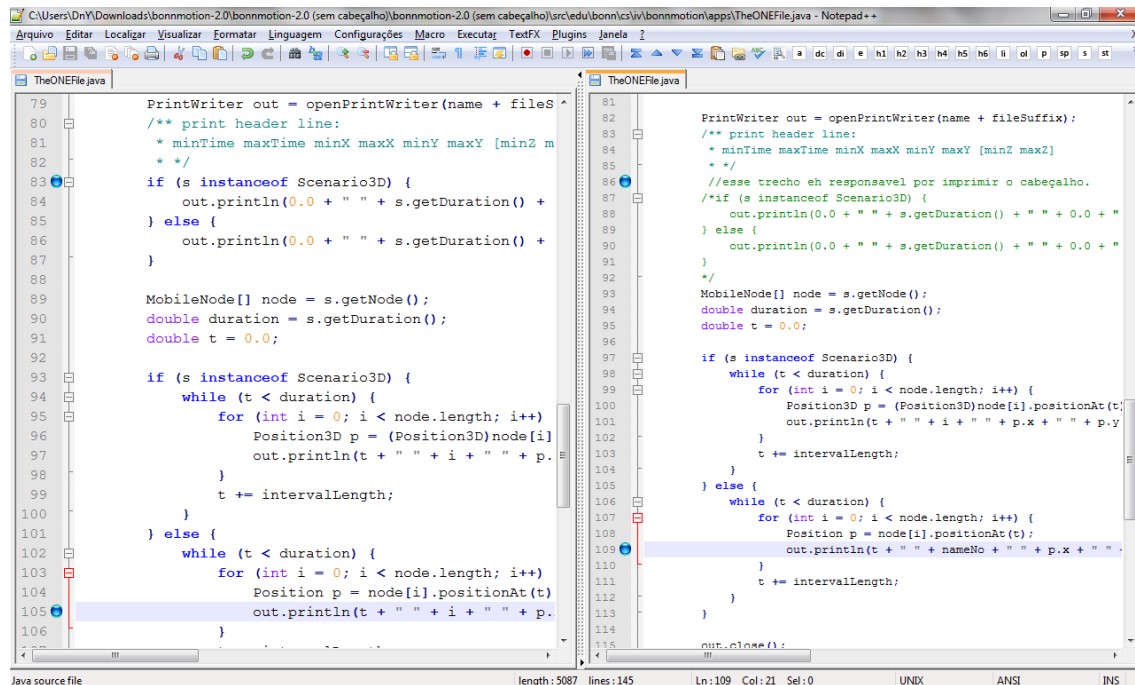
O arquivo modificado foi o TheONEFile.java, como mostra a imagem:



Como era (imagem esquerda) e como ficou(imagem direita).

Foi adicionado mais um parâmetro(-n ) para que pudesse suportar a escolha do nome do no que será impresso.

Após feito isso, foi feito mais uma modificação para que o cabeçalho de cada arquivo não fosse impresso, imprimindo assim somente o caminho do no, foi também modificado para que imprimissem em vez do valor padrão 0 (zero), o valor determinado pelo parâmetro citado anteriormente. Como mostra a imagem abaixo:

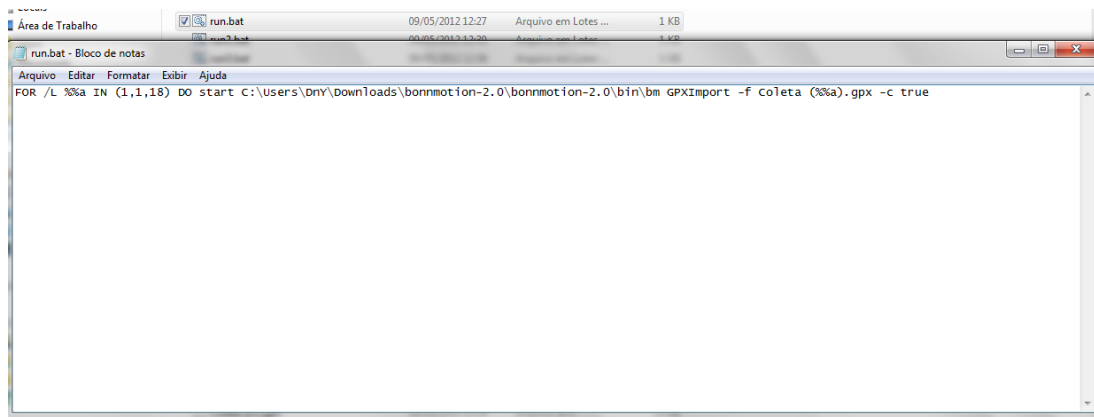


Como era (imagem esquerda) e como ficou (imagem direita).

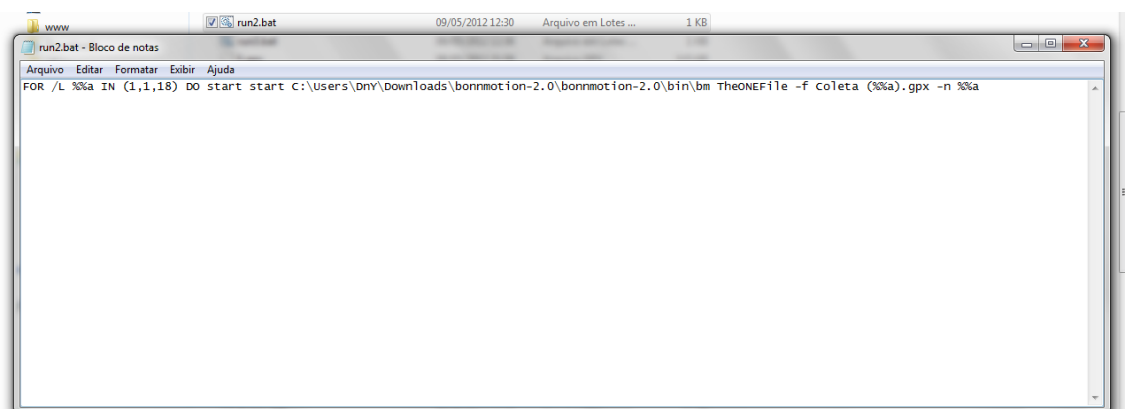
Como próximo passo foi padronizado os nomes dos arquivos de entrada (arquivos com extensão “.gpx”). Figura abaixo:

<input checked="" type="checkbox"/>	Coleta (1).gpx	08/05/2012 21:58	Arquivo GPX	115 KB
<input checked="" type="checkbox"/>	Coleta (2).gpx	08/05/2012 20:33	Arquivo GPX	89 KB
<input checked="" type="checkbox"/>	Coleta (3).gpx	09/05/2012 12:29	Arquivo GPX	12 KB
<input checked="" type="checkbox"/>	Coleta (4).gpx	09/05/2012 12:29	Arquivo GPX	45 KB
<input checked="" type="checkbox"/>	Coleta (5).gpx	09/05/2012 12:29	Arquivo GPX	8 KB
<input checked="" type="checkbox"/>	Coleta (6).gpx	09/05/2012 12:29	Arquivo GPX	34 KB
<input checked="" type="checkbox"/>	Coleta (7).gpx	09/05/2012 12:29	Arquivo GPX	8 KB
<input checked="" type="checkbox"/>	Coleta (8).gpx	09/05/2012 12:29	Arquivo GPX	43 KB
<input checked="" type="checkbox"/>	Coleta (9).gpx	08/05/2012 15:10	Arquivo GPX	1 KB
<input checked="" type="checkbox"/>	Coleta (10).gpx	09/05/2012 12:29	Arquivo GPX	12 KB
<input checked="" type="checkbox"/>	Coleta (11).gpx	09/05/2012 12:29	Arquivo GPX	6 KB
<input checked="" type="checkbox"/>	Coleta (12).gpx	09/05/2012 12:29	Arquivo GPX	7 KB
<input checked="" type="checkbox"/>	Coleta (13).gpx	09/05/2012 12:29	Arquivo GPX	39 KB
<input checked="" type="checkbox"/>	Coleta (14).gpx	09/05/2012 12:29	Arquivo GPX	40 KB
<input checked="" type="checkbox"/>	Coleta (15).gpx	09/05/2012 12:29	Arquivo GPX	31 KB
<input checked="" type="checkbox"/>	Coleta (16).gpx	09/05/2012 12:29	Arquivo GPX	52 KB
<input checked="" type="checkbox"/>	Coleta (17).gpx	09/05/2012 12:29	Arquivo GPX	12 KB
<input checked="" type="checkbox"/>	Coleta (18).gpx	09/05/2012 12:29	Arquivo GPX	44 KB

Após isso foi criado o mini-programa batch run.bat, que aplica a instrução “bm GPXImport” para todos os arquivos gerados pelo mytrack e já com os nomes padronizados:



Depois foi criado o run2.bat, que roda para todos os arquivos a instrução “bm TheONEFile”, onde também determina qual será o nome do no, determinado pelo parâmetro “-n”:



Como saída tem-se esses arquivos:

<input checked="" type="checkbox"/>	Coleta (2).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (3).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (4).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (5).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (6).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (7).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (8).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (10).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (11).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (12).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (13).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (14).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (15).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (16).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (17).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (18).gpx.params	09/05/2012 12:30	Arquivo PARAMS	1 KB
<input checked="" type="checkbox"/>	Coleta (1).gpx.one	09/05/2012 12:34	Seção do Microso...	184 KB
<input checked="" type="checkbox"/>	Coleta (2).gpx.one	09/05/2012 12:34	Seção do Microso...	1.669 KB
<input checked="" type="checkbox"/>	Coleta (3).gpx.one	09/05/2012 12:34	Seção do Microso...	32 KB
<input checked="" type="checkbox"/>	Coleta (4).gpx.one	09/05/2012 12:34	Seção do Microso...	57 KB
<input checked="" type="checkbox"/>	Coleta (5).gpx.one	09/05/2012 12:34	Seção do Microso...	51 KB
<input checked="" type="checkbox"/>	Coleta (6).gpx.one	09/05/2012 12:34	Seção do Microso...	242 KB
<input checked="" type="checkbox"/>	Coleta (7).gpx.one	09/05/2012 12:34	Seção do Microso...	274 KB
<input checked="" type="checkbox"/>	Coleta (8).gpx.one	09/05/2012 12:34	Seção do Microso...	191 KB
<input checked="" type="checkbox"/>	Coleta (10).gpx.one	09/05/2012 12:34	Seção do Microso...	115 KB
<input checked="" type="checkbox"/>	Coleta (11).gpx.one	09/05/2012 12:34	Seção do Microso...	22 KB
<input checked="" type="checkbox"/>	Coleta (12).gpx.one	09/05/2012 12:34	Seção do Microso...	166 KB
<input checked="" type="checkbox"/>	Coleta (13).gpx.one	09/05/2012 12:34	Seção do Microso...	33 KB

Com essas informações:

The image shows two side-by-side screenshots of a Notepad++ window. The left window displays a file named '2.gpx.one' containing a list of coordinates and timestamps. The right window displays a file named 'Coleta (3).gpx.one' containing a more extensive list of coordinates and timestamps. The status bar at the bottom indicates 'length: 32050 lines: 716'.

Como era (imagem esquerda) e como ficou(imagem direita).

Como terceiro programa foi criado o run3.bat, que junta todos os resultados em um único arquivo chamado “saída.one”:

The image shows a Notepad++ window with the file 'run3.bat'. The script content is as follows:

```
FOR /L %a IN (1,1,18) DO copy /b "saída.one"+"Coleta (%a).gpx.one" "saída.one"
```

The status bar at the bottom indicates 'run3.bat' was created on '09/05/2012 12:36' and is '1 KB' in size.

Resultado: .one:



```

38478 697.0 3 5515.980646568909 5747.5394150019165
38479 698.0 3 5516.093150290102 5750.421589674963
38480 699.0 3 5516.205654011294 5753.303764348012
38481 700.0 3 5516.318157732487 5756.185939021059
38482 701.0 3 5513.539042454213 5758.029561973468
38483 702.0 3 5510.759927175939 5759.873184925877
38484 703.0 3 5507.980811897665 5761.716807878285
38485 704.0 3 5505.201696619391 5763.560430830694
38486 705.0 3 5503.101819046463 5767.15347452322
38487 706.0 3 5501.001941473533 5770.746518215747
38488 707.0 3 5498.902063900605 5774.339561908273
38489 708.0 3 5498.6055075276645 5777.176313558547
38490 709.0 3 5498.308951154724 5780.013065208821
38491 710.0 3 5498.012394781783 5782.849816859094
38492 711.0 3 5497.7158384088425 5785.686568509368
38493 712.0 3 5497.419282035902 5788.523320159642
38494 713.0 3 5496.900318320841 5793.487637612736
38495 714.0 3 5496.381354605779 5798.45195506583
38496 0.0 4 420.7802733127028 1853.3393005331745
38497 1.0 4 420.63127396330236 1854.7585200241535
38498 2.0 4 420.48227461390195 1856.1777395151323
38499 3.0 4 420.3332752645016 1857.596959006111
38500 4.0 4 420.1842759151012 1859.0161784970899
38501 5.0 4 420.03527656570077 1860.4353979880689
38502 6.0 4 419.88627721630036 1861.8546174790476
38503 7.0 4 419.73727786689994 1863.2738369700267
38504 8.0 4 419.5882785174996 1864.6930564610054
38505 9.0 4 419.4392791680992 1866.1122759519842
38506 10.0 4 419.29027981869876 1867.5314954429632

```

Note que junta o final de um arquivo com o início do outro

Por último todos esses dados foram ordenados usando a ajuda do Microsoft Excel:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
4	0	4	420,7803	1853,339															
5	0	5	4863,653	6091,228															
6	0	6	4635,763	5747,36															
7	0	7	4819,481	6117,893															
8	0	8	4652,078	5878,321															
9	0	10	4920,243	5678,506															
10	0	11	4791,9	5896,717															
11	0	12	4773,164	5917,596															
12	0	13	5468,796	5874,305															
13	0	14	544,4241	2174,394															
14	0	15	4854,537	6227,971															
15	0	16	4658,458	5718,31															
16	0	17	4915,546	6079,769															
17	0	18	5384,034	5784,541															
18	1	1	#NOME?	5090,904															
19	1	2	4719,638	-1,66E+88															
20	1	3	4907,157	6219,462															
21	1	4	420,6313	1854,759															
22	1	5	4862,803	6091,817															
23	1	6	4648,133	5732,949															
24	1	7	4819,487	6117,908															
25	1	8	4654,28	5880,653															
26	1	10	4923,843	5681,457															
27	1	11	4787,325	5894,295															
28	1	12	4773,171	5917,598															
29	1	13	5472,941	5870,939															
30	1	14	544,4254	2174,479															

Concluído a ordenação dos valores, é necessário criar o cabeçalho para o arquivo final que será entregue, para isso é necessário imprimir somente os cabeçalhos e ignorar os dados, ou seja, o inverso do que foi feito até agora, para isso foi necessário modificar novamente o arquivo TheONEFile.java:



```

85  */
86  //esse trecho eh responsavel por imprimir o c
87  /*if (s instanceof Scenario3D) {
88      out.println(0.0 + " " + s.getDuration() +
89  } else {
90      out.println(0.0 + " " + s.getDuration() +
91  }
92  */
93  MobileNode[] node = s.getNode();
94  double duration = s.getDuration();
95  double t = 0.0;
96
97  if (s instanceof Scenario3D) {
98      while (t < duration) {
99          for (int i = 0; i < node.length; i++)
100              Position3D p = (Position3D)node[i]
101              out.println(t + " " + i + " " + p.x
102              }
103              t += intervalLength;
104          }
105      } else {
106          while (t < duration) {
107              for (int i = 0; i < node.length; i++)
108                  Position p = node[i].positionAt(t)
109                  out.println(t + " " + nameNo + " "
110                  }
111                  t += intervalLength;
112              }
113          }
114      }
115  }
116  }

```

Java source file length: 5029 lines: 144 Ln: 20 Col: 1 Sel: 0 UNIX ANSI INS

Como era (imagem esquerda) e como ficou(imagem direita).

E da mesma forma como foi feito anteriormente, os programas run.bat, run2.bat e run3.bat foram executados em sequencia, como resultado se obteve esse resultado:

```

1 0.0 4944.0 0.0 8453.617238426581 0.0 8116.158936453634
2 0.0 32836.0 0.0 8331.324723841622 0.0 7987.7943758935435
3 0.0 715.0 0.0 8453.617238426581 0.0 8116.158936453634
4 0.0 1275.0 0.0 8453.617238426581 0.0 8116.158936453634
5 0.0 1148.0 0.0 8453.617238426581 0.0 8116.158936453634
6 0.0 5408.0 0.0 8453.617238426581 0.0 8116.158936453634
7 0.0 6126.0 0.0 8453.617238426581 0.0 8116.158936453634
8 0.0 4272.0 0.0 8453.617238426581 0.0 8116.158936453634
9 0.0 2518.0 0.0 8453.617238426581 0.0 8116.158936453634
10 0.0 482.0 0.0 8453.617238426581 0.0 8116.158936453634
11 0.0 3622.0 0.0 8453.617238426581 0.0 8116.158936453634
12 0.0 714.0 0.0 8453.617238426581 0.0 8116.158936453634
13 0.0 1576.0 0.0 8453.617238426581 0.0 8116.158936453634
14 0.0 10508.0 0.0 8453.617238426581 0.0 8116.158936453634
15 0.0 10807.0 0.0 8453.617238426581 0.0 8116.158936453634
16 0.0 1876.0 0.0 8453.617238426581 0.0 8116.158936453634
17 0.0 2238.0 0.0 8453.617238426581 0.0 8116.158936453634
18
19

```

Onde com a ajuda novamente do Microsoft Excel, chegou-se nesses valores:

	0	4944	0	8453,617	0	8116,159
	0	32836	0	8331,325	0	7987,794
	0	715	0	8453,617	0	8116,159
	0	1275	0	8453,617	0	8116,159
	0	1148	0	8453,617	0	8116,159
	0	5408	0	8453,617	0	8116,159
	0	6126	0	8453,617	0	8116,159
	0	4272	0	8453,617	0	8116,159
	0	2518	0	8453,617	0	8116,159
	0	482	0	8453,617	0	8116,159
	0	3622	0	8453,617	0	8116,159
	0	714	0	8453,617	0	8116,159
	0	1576	0	8453,617	0	8116,159
	0	10508	0	8453,617	0	8116,159
	0	10807	0	8453,617	0	8116,159
	0	1876	0	8453,617	0	8116,159
	0	2238	0	8453,617	0	8116,159
max=	0	32836	0	8453,617	0	8116,159

Após criado o cabeçalho, ele foi anexado no início do arquivo gerado anteriormente, gerando esse arquivo final:

```

1 0 32836 0 8453.617238 0 8116.158936
2 0 1 NaN 5091.347524
3 0 2 4719.63897 -1.66E+88
4 0 3 4907.720172 6233.873886
5 0 4 420.7802733 1853.339301
6 0 5 4863.652716 6091.22758
7 0 6 4635.763098 5747.359845
8 0 7 4819.481465 6117.89307
9 0 8 4682.077994 5878.321143
10 0 10 4920.242719 5678.5061
11 0 11 4791.900183 5896.71668
12 0 12 4773.163802 5917.595767
13 0 13 5468.795607 5874.305087
14 0 14 544.4241235 2174.394032
15 0 15 4854.537129 6227.970838
16 0 16 4658.458273 5718.310392
17 0 17 4915.546198 6079.769107
18 0 18 5384.03426 5784.540718
19 1 1 -Infinity 5090.904496
20 1 2 4719.637551 -1.66E+88
21 1 3 4907.156782 6219.461962
22 1 4 420.631274 1854.75852
23 1 5 4862.802889 6091.816551
24 1 6 4648.139455 5732.948707
25 1 7 4819.487415 6117.907529
26 1 8 4654.279786 5880.652882
27 1 10 4923.842755 5681.456946
28 1 11 4787.325005 5894.295373
29 1 12 4773.171121 5917.598195
30 1 13 5472.940811 5870.939127
31 1 14 544.4253934 2174.478633
32 1 15 4855.463922 6227.960423
33 1 16 4658.458941 5718.353732
34 1 17 4916.554128 6078.924376
35 1 18 5386.233012 5780.002023
36 2 1 -Infinity NaN 461467

```

## 7. Conclusão

A rede DTN é uma rede de comutação de mensagens, diferentemente da Internet que é baseada em comutação de pacotes.

Em resumo, esta segunda parte do trabalho está em pegarmos os dados coletados por algum programa de manipulação de arquivos de GPS como o google earth, o GPSTabel ou o track maker, e depois tracar o limite superior e mínimo do mapa de coleta, latitude e longitude, e colocarmos esses valores dentro de arquivos .gpx gerado.

Tendo a equipe coletado os dados, com esses .gpx gerados, usamos o Bonnmotion para criarmos através de dois comandos simples e os .gpx, alguns arquivos, dos quais usamos o arquivo .one gerado para o próximo passo.

Com todos os .one em mãos, agora teremos que junta-los, em um arquivo .one apenas. Essa parte leva um grande trabalho, pois o arquivo grande deve seguir um padrão, utilizando cada coordenada gerada obtida. Nossa equipe usou o Calc para manipulação das colunas, e padronizar o arquivo global necessário. Iremos então, com todos os dados jogá-los no simulador ONE.

A validação do modelo é extremamente importante, pois os simuladores normalmente tendem a parecerem reais e, tanto o modelador como o usuário passam a acreditar nele.

## Referências

COUTINHO, Gustavo L. Delay Tolerant Networks (DTN). UFRJ, 2006. Disponível em [http://www.gta.ufrj.br/grad/06\\_2/gustavo/arquitetura.htm](http://www.gta.ufrj.br/grad/06_2/gustavo/arquitetura.htm). Acesso em: 09/05/2012.

COSTA, Miguel A. B. da. Simulação de Sistemas Parte 1: Introdução à Simulação. São Carlos, 2002. Disponível em: [http://www.simucad.dep.ufscar.br/dn\\_sim\\_doc01.pdf](http://www.simucad.dep.ufscar.br/dn_sim_doc01.pdf). Acesso em: 10/05/2012.

LINDGREN, Anders. PHANSE, Kaustubh S. Evaluation of Queueing Policies and Forwarding Strategies for Routing in Intermittently Connected Networks. Lulea University of Technology. Lulea, Suécia, ????

GUEDES, Raphael M. Escassez de Recursos em Redes Tolerantes a Atrasos e Interrupções. (Dissertação de Mestrado). COPPE UFRJ, 2009.

MOTA, Edjair. Simulação de Sistemas. (Notas de Aula). UFAM, 2012.