MAC0463/5743 - Computação Móvel
Exercício Programa 3 - The One

Diogo Haruki Kykuta
nUSP: 6879613
Instituto de Matemática e Estatística
Universidade de São Paulo
Email: haruki@linux.ime.usp.br

Fernando Omar Aluani
nUSP: 6797226
Instituto de Matemática e Estatística
Universidade de São Paulo
Email: rewasvat@linux.ime.usp.br

***Resumo*—Neste trabalho, apresentamos alguns resultados de simulações usando o *The One* e motivações, assim como uma análise superficial dos resultados obtidos.**

I. Introdução

O *The One* é um simulador de DTNs. Neste trabalho, apresentaremos resultados de simulações em quatro cenários diferentes, com padrão de movimentação e interfaces de comunicação fixadas para cada cenário e variando o protocolo de roteamento.

A. Motivação

Escolhemos inicialmente 4 cenários que nos parecessem interessantes. Para cada cenário, escolhemos alguns protocolos de roteamento que gostaríamos de testar seu desempenho. Juntando todos os protocolos escolhidos tivemos uma lista de seis protocolos de roteamento, e rodamos simulações dos cenários para cada protocolo na lista. Listamos a seguir os cenários, os protocolos escolhidos e uma breve explicação de suas escolhas.

1) CentroSP: Julgamos esse cenário interessante por ser o único de nossos cenários que fazia sentido simular para um tempo grande (24 horas). Afinal, São Paulo não para.

- DirectDeliveryRouter: Pela curiosidade de ver o funcionamento desse protocolo num ambiente com muitas pessoas se movimentando.
- EpidemicRouter: Queríamos ver o funcionamento de um protocolo epidêmico nesse ambiente onde há uma grande movimentação de pessoas.

2) USP: Não poderíamos deixar de incluir nossa segunda casa nos cenários. Este é o cenário "USP2"disponibilizado.

- SprayAndWaitRouter e ProphetRouter: Escolhas aleatórias de protocolos amplamente usados baseados em artigos entre as possibilidades presentes no *The One*.
- EpidemicRouter: Queríamos ver o funcionamento de um protocolo epidêmico nesse ambiente onde há uma grande movimentação de pessoas.

3) Estacionamento: Nos interessamos pelas muitas vagas imóveis e gostaríamos de ver como as transmissões eram feitas nesse caso e poder comparar com os demais cenários desse trabalho, que possuem nós se movimentando intensamente.

- FirstContactRouter: parecia fazer sentido transmitir para o primeiro que eu encostar num estacionamento com vagas imóveis.

4) ParqueIbirapuera: O cenário do Parque do Ibirapuera parece bem completo, tendo 6 grupos com características bem interessantes. Tanto que, para ser viável, não pudemos usar um tempo de simulação maior, mas apenas de 1 hora, pois este cenário era bem complexo e sua execução, bem mais demorada.

- ProphetRouter e MaxPropRouter: Escolhas aleatórias de protocolos amplamente usados baseados em artigos entre as possibilidades presentes no *The One*.

B. Fundamentação

Segue uma explicação sobre os protocolos de roteamento usados:

1) DirectDeliveryRouter: É um protocolo ingênuo que só transmite uma mensagem a um nó se o nó for o destino da mensagem. Dessa forma, as mensagens fazem um único *hop* (salto) para chegar onde devem chegar, mas levam mais tempo para sair do nó de origem pois devem esperar o nó de origem e o de destino estarem em alcançe para conseguir se comunicar.

2) EpidemicRouter: É um protocolo ingênuo baseado na "enchente"de mensagens na rede. Um nó continuamente copia e transmite uma mensagem para contatos que ainda não possuam uma cópia dessa mensagem. Dessa forma é meio garantido que as mensagens cheguem onde devem chegar, mas isso consome muitos recursos e pode causar muito congestionamento na rede.

3) FirstContactRouter: É um protocolo ingênuo de roteamento que transmite uma mensagem para o primeiro nó que ele encontra. Isso confere algumas propriedades estranhas para o protocolo, pois o funcionamento dele é altamente dependente do protocolo de comunicação e dos padrões de movimento dos nós.

4) MaxPropRouter: Foi baseado no artigo *MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks por John Burgess et al.*

Esse protocolo é baseado em replicação: quando um nó é descoberto, todas mensagens que aquele nó não tem serão possivelmente copiadas e transferidas. A diferença do MaxProp está em como ele determina quais menssagens deverão ser transmitidas antes e quais deverão ser canceladas antes.

Para tal, ele mantém uma fila ordenada baseada na destinação de cada mensagem, ordenado pela probabilidade estimada de um caminho transitivo futuro para aquele destino. Para obter probabilidades de caminho estimado, cada nó mantém um vetor de tamanho $N-1$ (onde N é o número de nós na rede) consistindo da probabilidade do nó encontrar cada outro nó na rede. Esse vetor é inicializado com uma probabilidade igual para cada nó, e ao encontrar outro nó, a probabilidade daquele nó aumenta em 1 e o vetor é normalizado para a somatória das probabilidades equivaler a 1.

Quando nós se encontram, eles transmitem seus vetores para o outro nó, e é usando o seu vetor e o vetor conhecido de outros nós que um nó calcula o provável caminho mais curto para o destino, e baseado nisso também é calculado o custo do caminho que é usado para determinar a ordem em que as mensagens são transmitidas e canceladas.

Fora essa funcionalidade padrão descrita acima, o MaxProp permite diversas modificações e adições de modo a melhorar sua eficiência. O MaxProp implementado pelo *The One* implementa uma extensão baseada no artigo *Time Scales and Delay-Tolerant Routing Protocols por Karvo and Ott*. Essa extensão altera como o vetor de probabilidade de um nó encontrar os outros é atualizado ao começar uma nova conexão. Essa atualização é baseada num parâmetro α, com o valor padrão $\alpha = 1$ (dessa forma, ele se comporta como o algoritmo original).

5) ProphetRouter: Foi baseado no artigo *Probabilistic routing in intermittently connected networks by Anders Lindgren et al.*

Esse protocolo tenta explorar a não-aleatoridade de encontros no mundo real mantendo um conjunto de probabilidade de entregar com sucesso uma mensagem para cada destino na DTN. Ele usa um algoritmo adaptativo para determinar a probabilidade de um nó conseguir entregar uma mensagem, e altera as probabilidades a cada encontro de acordo com algumas regras:

- Se a probabilidade para um destino não é conhecida, é assumida como zero;
- Quando um nó A encontra outro nó B, a probabilidade de B é aumentada;
- Quando um nó A encontra outro nó B, a probabilidade de todos destinos diferentes de B são "envelhecidas";
- Quando um nó A encontra outro nó B, eles trocam probabilidades, usando a propriedade transitiva das probabilidades para atualizar os valores de destinos D para qual B tem um valor na suposição que A provavelmente vai encontrar B de novo;

O protocolo então só copia uma mensagem e transmite para outro nó se aquele nó tiver uma probabilidade maior de entregar a mensagem.

6) SprayAndWaitRouter: Foi baseado no artigo *Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks por Thrasyvoulos Spyropoulus et al.*

Esse protocolo tenta misturar os benefícios de protocolos de roteamento baseados em replicação de mensagem com protocolos baseados em encaminhamento. Ele é composto por duas fases:

- Fase de *Spray*: responsabilidade do nó de origem, essa é a fase inicial e consiste em fazer N cópias da mensagem e transmitir cada cópia para um nó diferente.
- Fase de *Wait* (Espera): é como fica o nó de origem depois de terminar a Fase de *Spray*, ou os nós ao receberem uma mensagem. Eles simplesmente esperam encontrar o destino da mensagem para entregá-la diretamente.

Existem dois métodos de protocolo: o padrão e o binário. A única diferença entre eles é como as N cópias da mensagem são distribuídas durante a Fase de *Spray*. Pelo que vimos, a implementação desse protocolo no *The One* aceita ambos métdos, mas nós só usamos o padrão.

II. Metodologia dos testes

Escolhemos primeiro quatro cenários e listamos protocolos de roteamento que achamos interessantes, como explicado em I-A. Decidimos, então, simular cada cenário para cada todos os protocolos que apareceram em qualquer das listas. Assim, testamos cada cenário para os seguintes protocolos: DirectDeliveryRouter, EpidemicRouter, FirstContactRouter, MaxPropRouter, ProphetRouter e SprayAndWaitRouter.

III. Resultados obtidos

Mais mimimi teste USP (*Prophet*), USP (*SprayAndWait*) e Parking (*MaxProp*)

IV. Conclusões

Concluímos que nada sabemos.

latency_avg	1379.4184 (Prophet)	1648.8626 (FirstContact)	1470.90068333	90.4342641804
delivered	2999.0 (FirstContact)	8966.0 (SprayAndWait)	5914.83333333	2210.33937188
hopcount_avg	1.0 (DirectDelivery)	8.6519 (FirstContact)	3.34506666667	2.45038987419
rtt_med	1217.4 (FirstContact)	3130.9 (MaxProp)	2433.7	706.744734681
relayed	3013.0 (DirectDelivery)	353485.0 (Epidemic)	210891.5	126577.764175
started	3014.0 (DirectDelivery)	354338.0 (Epidemic)	211490.333333	126884.584733
buffertime_avg	374.0239 (FirstContact)	3879.7628 (DirectDelivery)	2198.27053333	1120.85311105
created	22467.0 (DirectDelivery)	25126.0 (SprayAndWait)	24022.5	1086.62439846
aborted	1.0 (DirectDelivery)	874.0 (MaxProp)	598.833333333	314.735559196
buffertime_med	87.8 (FirstContact)	3574.1 (DirectDelivery)	1920.98333333	1202.58914072
sim_time	86400.0 (DirectDelivery)	86400.0 (DirectDelivery)	86400.0	0.0
response_prob	0.0246 (FirstContact)	0.1886 (SprayAndWait)	0.120616666667	0.0584497623795
dropped	19496.0 (FirstContact)	370886.0 (Epidemic)	172287.833333	134360.01799
overhead_ratio	0.0 (DirectDelivery)	61.5892 (FirstContact)	35.3758	22.5849459047
delivery_prob	0.1327 (FirstContact)	0.3568 (SprayAndWait)	0.242616666667	0.0830614314161
removed	0.0 (DirectDelivery)	187705.0 (FirstContact)	56506.6666667	80599.2441893
rtt_avg	1589.2041 (FirstContact)	3137.3417 (MaxProp)	2533.37816667	630.717832815
hopcount_med	1.0 (DirectDelivery)	5.0 (FirstContact)	2.5	1.25830573921
latency_med	1064.3 (DirectDelivery)	1345.2 (Epidemic)	1242.3	99.2917082809

ACKNOWLEDGMENT

The authors would like to thank...

REFERÊNCIAS

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.