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 alcance 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 TAL.

5) ProphetRouter: Foi baseado no artigo *Probabilistic Routing in Intermittently Connected Networks* [1].

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* [2].

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

Nesta seção abordaremos os resultados obtidos, fazendo uma análise geral do significado de tais resultados. Na seção III-B apresentaremos tabelas com detalhamento sobre os valores encontrados nos *reports*.

A. CentroSP

Neste cenário não fomos capazes de simular para o protocolo *MaxProp* nos parâmetros desejados. Este protocolo tem uma complexidade um pouco superior e sua simulação é muito mais lenta. Para este cenário, queríamos testar intervalos de 24 horas, como o *MaxProp* estava simulando a uma taxa de 0.16 segundos virtuais por segundo real, demoraria mais de 6 dias para terminar a simulação.

B. Tabelas de resultados

1) USP:

latency_avg	1379.4184 (Prophet)
delivered	2999.0 (FirstContact)
hopcount_avg	1.0 (DirectDelivery)
rtt_med	1217.4 (FirstContact)
relayed	3013.0 (DirectDelivery)
started	3014.0 (DirectDelivery)
buffertime_avg	374.0239 (FirstContact)
created	22467.0 (DirectDelivery)
aborted	1.0 (DirectDelivery)
buffertime_med	87.8 (FirstContact)
sim_time	86400.0 (DirectDelivery)
response_prob	0.0246 (FirstContact)
dropped	19496.0 (FirstContact)
overhead_ratio	0.0 (DirectDelivery)
delivery_prob	0.1327 (FirstContact)
removed	0.0 (DirectDelivery)
rtt_avg	1589.2041 (FirstContact)
hopcount_med	1.0 (DirectDelivery)
latency_med	1064.3 (DirectDelivery)

2) CentroSP:

latency_avg	1379.4184 (Prophet)
delivered	2999.0 (FirstContact)
hopcount_avg	1.0 (DirectDelivery)
rtt_med	1217.4 (FirstContact)
relayed	3013.0 (DirectDelivery)
started	3014.0 (DirectDelivery)
buffertime_avg	374.0239 (FirstContact)
created	22467.0 (DirectDelivery)
aborted	1.0 (DirectDelivery)
buffertime_med	87.8 (FirstContact)
sim_time	86400.0 (DirectDelivery)
response_prob	0.0246 (FirstContact)
dropped	19496.0 (FirstContact)
overhead_ratio	0.0 (DirectDelivery)
delivery_prob	0.1327 (FirstContact)
removed	0.0 (DirectDelivery)
rtt_avg	1589.2041 (FirstContact)
hopcount_med	1.0 (DirectDelivery)
latency_med	1064.3 (DirectDelivery)

3) Parque do Ibirapuera:

latency_avg	44.6416 (MaxProp)	533.0569 (DirectDelivery)	301.88755	172.543226131
delivered	766.0 (DirectDelivery)	1607.0 (MaxProp)	1137.66666667	256.72986235
hopcount_avg	1.0 (DirectDelivery)	70.5813 (FirstContact)	16.3215333333	24.4487584603
rtt_med	NaN	NaN	NaN	NaN
relayed	766.0 (DirectDelivery)	2097237.0 (MaxProp)	986622.666667	945121.963093
started	769.0 (DirectDelivery)	2103173.0 (MaxProp)	990030.333333	948183.444938
buffertime_avg	7.2124 (Epidemic)	183.4433 (DirectDelivery)	89.262375	77.1539670186
created	1624.0 (DirectDelivery)	1624.0 (DirectDelivery)	1624.0	0.0
aborted	3.0 (DirectDelivery)	6896.0 (Prophet)	3360.16666667	3072.78339711
buffertime_med	0.2 (Epidemic)	109.2 (DirectDelivery)	45.95	46.0014401948
sim_time	3600.1 (DirectDelivery)	3600.1 (DirectDelivery)	3600.1	0.0
response_prob	0.0 (DirectDelivery)	0.0 (DirectDelivery)	0.0	0.0
dropped	0.0 (DirectDelivery)	1757166.0 (Prophet)	579484.5	819586.04838
overhead_ratio	0.0 (DirectDelivery)	1795.2241 (Epidemic)	831.517916667	812.137387435
delivery_prob	0.4717 (DirectDelivery)	0.9895 (MaxProp)	0.700533333333	0.158075410977
removed	0.0 (DirectDelivery)	2086185.0 (MaxProp)	369197.5	769303.590939
rtt_avg	NaN	NaN	NaN	NaN
hopcount_med	1.0 (DirectDelivery)	63.0 (FirstContact)	14.3333333333	21.9215773966
latency_med	27.3 (MaxProp)	285.0 (DirectDelivery)	141.533333333	90.219928816

4) Estacionamento:

latency_avg	23.0581 (Epidemic)	398.039 (DirectDelivery)	191.20162	143.277520784
delivered	231.0 (DirectDelivery)	241.0 (Epidemic)	236.6	3.92937654088
hopcount_avg	1.0 (DirectDelivery)	163.7597 (FirstContact)	36.36592	63.8004441409
rtt_med	NaN	NaN	NaN	NaN
relayed	231.0 (DirectDelivery)	153865.0 (Epidemic)	68087.2	68016.4193836
started	231.0 (DirectDelivery)	154153.0 (Epidemic)	68221.4	68087.2953218
buffertime_avg	1.9124 (DirectDelivery)	1.9124 (DirectDelivery)	1.9124	0.0
created	243.0 (DirectDelivery)	243.0 (DirectDelivery)	243.0	0.0
aborted	0.0 (DirectDelivery)	257.0 (Epidemic)	124.4	81.4434773324
buffertime_med	2.0 (DirectDelivery)	2.0 (DirectDelivery)	2.0	0.0
sim_time	7200.0 (DirectDelivery)	7200.0 (DirectDelivery)	7200.0	0.0
response_prob	0.0 (DirectDelivery)	0.0 (DirectDelivery)	0.0	0.0
dropped	0.0 (DirectDelivery)	0.0 (DirectDelivery)	0.0	0.0
overhead_ratio	0.0 (DirectDelivery)	637.444 (Epidemic)	283.17922	282.307896333
delivery_prob	0.9506 (DirectDelivery)	0.9918 (Epidemic)	0.97366	0.0161969873742
removed	0.0 (DirectDelivery)	39825.0 (FirstContact)	7965.0	15930.0
rtt_avg	NaN	NaN	NaN	NaN
hopcount_med	1.0 (DirectDelivery)	117.0 (FirstContact)	26.8	45.2389212957
latency_med	22.0 (Epidemic)	262.0 (DirectDelivery)	125.2	87.8052390236

IV. Conclusões

Concluímos que nada sabemos.

Acknowledgment

The authors would like to thank...

Referências

[1] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/961268.961272

[2] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 252–259. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080143