Trabalho Prático Arquiteturas Emergentes de redes*

João Neves[1 [PG47321]], Tiago Ribeiro[2 [PG47701]], and Luís Magalhães[3 [PG47415]]

Universidade do minho

Abstract. O presente trabalho foi proposto na unidade curricular de Arquiteturas Emergentes de Redes e tem como objetivo a implementação de um jogo multiplayer tem por base de comunicação um protocolo de DTN (delay tolerant network), capaz de correr em nodos moveis.

Keywords: DTN · Multiplayer · Redes · Arquitetura

* Supported by organization Universidade do minho.

1 Introdução

O presente projeto foi desenvolvido em duas fases distintas. A primeira fase de desenvolvimento consistiu na concepção de um jogo multiplayer, baseado numa arquitetura cliente-servidor e que fosse capaz de utilizar Ipv6 para a comunicação entre estes componentes.

A segunda fase,consistiu em , a partir do trabalho da primeira fase, criar um modulo DTN (delay tolerant Network), capaz de suportar o jogo previamente desenvolvido, e completamente desacoplado deste.Para além disso, este modulo teria de ser capaz de suportar a comunicação entre o servidor e clientes presentes em nodos moveis com conexão intermitente.

De seguida será apresentado em detalhe a especificação, implementação e alguns testes referentes ao trabalho desenvolvido, de forma a justificar todas as opções tomadas.

2 Especificação dos protocolos

2.1 Arquitetura da aplicação

Tendo em conta o carácter instável das condições da rede em questão, foi criado um protocolo que separa a camada aplicacional da camada de transporte, ficando no meio das duas e gerindo a comunicação entre estas. Esta camada DTN surge com o intuito de, independentemente das condições da rede em questão, fornecer funcionalidades que permitam assegurar a troca de informação, ainda que a conexão seja intermitente, ou mesmo inexistente.

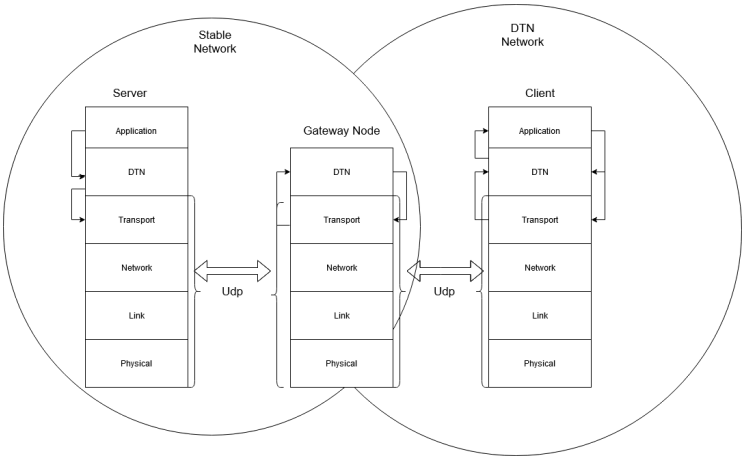

Fig. 1. Arquitetura da Solução

Por baixo segue-se a topologia de teste utilizada e que testará a capacidade do modulo DTN implementado.

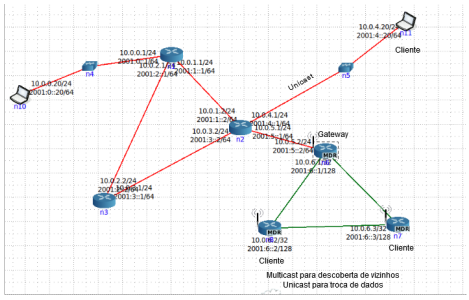

Fig. 2. Topologia utilizada

2.2 primitivas de comunicação

Servidor → Client O protocolo Dtn desenvolvido, tem como um dos seus objetivos diferenciar o tráfego da rede fixa do tráfego dos nodos moveis. Com isso em mente, foi desenvolvido um modulo dtn especifico para o servidor, capaz de receber dados de clientes fixos e moveis, e ter a capacidade de distinguir os dois e dar tratamentos diferentes a estes, tratamento estes explicados mais á frente.

Client → Servidor Sendo um protocolo bidirecional, o programa dos clientes tem também um modulo Dtn associado, capaz de enviar e receber dados de múltiplas fontes.Caso o cliente esteja ligado à rede fixa, o transporte de dados será feito diretamente com o servidor, caso contrario, o cliente terá obrigatoriamente que comunicar com um nodo edge da rede (gateway), isto é, que interliga a rede móvel à rede fixa.

Gateway O gateway é um auxiliar ao protocolo DTN, funcionando como um OTT node, que tem como objetivo efetuar a ligação entre o cliente movel e o servidor, sendo fundamental para a implementação dos conceitos associados a um algoritmo DTN. Primeiramente, assim que executado, o gateway, que tem como input o ip do server, envia um pedido para se conectar ao servidor, funcionando como um relay node, ou bundle node.De seguida, o gateway espera pela chegada de mensagens quer do lado do servidor quer do lado do cliente, reenviando-as para o seu destinatário, servidor ou cliente móvel.

Client → Cliente A comunicação entre clientes moveis distingue-se em dois processos. Primeiramente temos a descoberta de vizinhos, para isso, cada cliente, de uma forma constante, envia por multicast pedidos de pacotes recentes, de seguida caso um dos clientes contenha dados mais recentes ou tenha inputs em cache, estas mensagens sao trocadas de forma a que caso um dos nodos passe pelo gateway, a informação consiga la chegar de uma forma mais rapida.

2.3 interações

Servidor ↔ Gateway Tendo em conta o posicionamento do gateway e do servidor na topologia (rede fixa), podemos considerar que existe uma conectividade continua entre estes dois componentes.

Gateway → Client O modulo DTN tem a sua maior importância assim que é necessário enviar dados a um nodo móvel, tendo em conta o carácter intermitente da conexão, o gateway tem a capacidade de guardar os dados em cache caso não seja possível enviar ao destinatário.Com isto, assim que o gateway volte a ter conexão, os dados São reenviados. Para além disto e visto que esta solução não será, possivelmente, suficiente para uma aplicação com informação em constante

movimento, este nodo tem também a funcionalidade de transferir os dados em cache para qualquer utilizador que entre em contacto com ele, com o intuito de os reenviar ate ao cliente desejado (store carry and forward).

Client → Client Através do Protocolo de neighbors, os clientes trocam entre si dados do servidor que contenham em cache, assim como dados de input deles mesmos, com o intuito de eventualmente passa-los ao gateway para serem reenviados ao servidor.

Client → Gateway Através do caching dos inputs dos clientes, é possível que os clientes que não tem conexão estabelecida com o gateway, troquem dados de input de forma a que, assim que um deles contacte com o gateway, os dados sejam transferidos.

2.4 Formato das mensagens Protocolares

DTN Packet Este PDU é responsavel por encapsular os dados do jogo, assim fornecer informações de forma a controlar o algoritmo DTN.

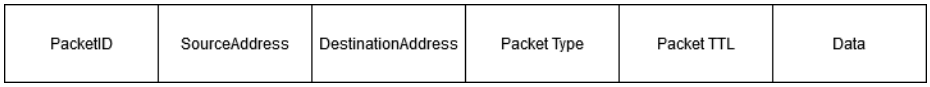

Fig. 3. DTN message

PacketID Este parâmetro define o numero de sequencia do pacote gerado, este numero é importante na medida em que permite ao modulo DTN distinguir os pacotes que já tenham chegado ao seu destino, daqueles que ainda estão a "navegar na rede" e ainda nao tenham chegado à rede fixa

SourceAddress Representa a origem do pacote, é importante conhecer a origem do pacote, para não confundir pacotes de clientes diferentes.

DestinationAddress Representa o destino do Pacote

Packet Type Representa o tipo de pacote:

1. **inteiro 0** → Representa pacotes provenientes do servidor ou de um cliente da rede fixa.
2. **inteiro 1** → Representa um pacote que tenha saído de um cliente Móvel.
3. **inteiro 2** → Representa um pacote com um pedido de inicio de conexão (gateway)
4. **inteiro 3** → Representa um pacote proveniente de um cliente sem conectividade.

Packet TTL Representa o tempo de vida de um pacote, assim que este valor atinge 0 o pacote é descartado, por razoes de controlo de trafego, defeniu-se os pacotes do client com um tempo de vida inferior aos pacotes do Servidor.

Data Representa o payload do pacote, contendo a informação proveniente da aplicação

Multicast Packet De seguida temos a constituição das mensagens multicast, enviadas de forma continua ao longo do decorrer da aplicação. Estas mensagens contem simplesmente o id do ultimo pacote recebido pelo cliente que enviou Multicast, assim como o seu próprio Address.

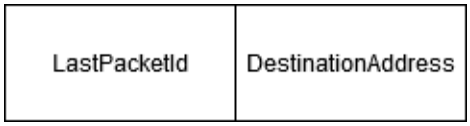

Fig. 4. Multicast message

3 Implementação

3.1 Servidor DTN

Classe DTN aplicada ao Servidor da aplicação De forma a modificar a comunição do servidor, para este comunicar pelo algoritmo implementado, foi instanciado um objeto da classe DTN no codigo do Servidor, de forma a que, ao inves de enviar e receber mensagens de um socket udp, o servidor passa a enviar e receber diretamente do protocolo DTN implementado.

Thread Receive Esta Thread contem um buffer (BufferR) responsavel por armazenar os pacotes que chegam ao servidor,sendo prosteriormente pedidos pela aplicação.Esta Thread contem um socket UDP constantemente á escuta por pacotes referentes ao modulo DTN (DTN Packet).

Thread send Esta Thread tem como objetivo enviar os dados provenientes da aplicação.Para isto, utilizou-se um buffer (BufferE) que contem os dados provenientes da aplicação e que deverão ser enviados para o cliente(no caso de rede fixa) ou para o gateway(rede movel).De forma a haver distinção entre clientes, utilizou-se uma lista de nodos moveis ativos (mobile Nodes).

Mobile nodes O modulo DTN do servidor mantém um registo de todos os nodos moveis conectados, de forma a saber quais pacotes mandar pelo gateway ativo e quais mandar diretamente para um cliente.

Gateways O modulo DTN do servidor tem a capacidade de receber pedidos de conexão de Gateways e guarda-los numa lista de gateways ativos, de forma a enviar e receber destes.No entanto, esta funcionalidade nao foi totalmente implementada do lado do cliente, de forma a que so é possivel a existencia de um Gateway.

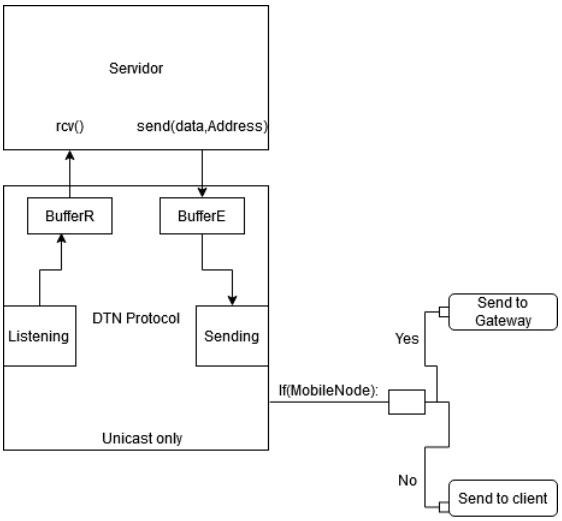

Fig. 5. Arquitetura Servidor Com modulo DTN

3.2 Gateway

Thread Receive Thread responsável pela receção de pacotes, quer dos clientes moveis quer do servidor.Após a receção de um pacote, é feita a verificação do tipo do pacote, caso tenha vindo do Cliente, é imediatamente retransmitido para o servidor, caso seja proveniente do Servidor, este é armazenado em buffer para ser reenviado por outra Thread.

Thread Send/Forwarding Strategy Thread responsavél pelo envio de pacotes a nodos com conexão intermitente, esta thread tem como objetivo enviar os dados provenientes do servidor, e em caso de nao ser possivel, mante-los em buffer ate ser possivel enviar.De forma a aumentar a probabilidade de fazer chegar os pacotes ao cliente destino, esta componente da aplicação, envia os pacotes em cache a todos os clientes que se conectem ao Gateway.Esta transferência permite que os pacotes possam, provavelmente, através da NeighborStrategy chegarem ao cliente em questão.

BufferR Buffer responsavel por guardar os pacotes que chegam do servidor.

Neighbor Buffer Responsavel por guardar os pacotes que tentaram ser enviados pelo gateway, mas nao conseguiram por nao haver conexao ao Cliente.

3.3 Protocolo de vizinhança

Algoritmo de procura e transmissão de pacotes entre vizinhos
ipv6 Group *ff15:7079:7468:6f6e:6465:6d6f:6d63:6173´*

Thread Receive Esta componente é responsavel por receber pedidos multicast de outros clientes.Estes pedidos multicast despertam duas funcionalidades desta Thread, a primeira, representa a resposta a este pedido,isto é, caso o cliente que recebeu o pedido contenha no seu neighborBuffer pacotes com um id mais recente aquele recebido pelo multicast e caso o Address seja o mesmo, é efetuada a troca de informação.A segunda funcionalidade representa a o envio de inputs em cache(caso exista) do cliente q recebeu o pedido, de forma a aumentar a probabilidade de receção de pacotes por parte do servidor.

Thread Send Esta Thread destina-se ao envio, em multicast, de um pacote do tipo multicast.Este envio é realizado de forma constante e a partir do momento em que o cliente recebe a primeira mensagem do servidor, esta Thread tem como objetivo inundar a rede de pedidos de pacotes, de forma a que, quando este cliente se desconecta, outro poderá receber os pedidos e responder ou não com pacotes.

ClientCache Representa o buffer onde sao guardados os pacotes gerados pelo proprio cliente que nao conseguem ser entregues.

Neighbor Buffer Representa o buffer que contem dados de outros Clientes, dados estes passados pelo gateway por nao conseguir entregar ao destino.

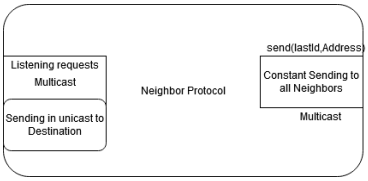

Fig. 6. Neighbor Protocol

3.4 Cliente DTN

Classe DTN aplicada ao Cliente da aplicação De forma a modificar a comunição do Cliente, para este comunicar pelo algoritmo implementado, foi instanciado um objeto da classe DTN no codigo do Cliente, de forma a que, ao invés de enviar e receber mensagens de um socket udp, o Cliente passa a enviar e receber diretamente do protocolo DTN implementado.

Thread Receive Esta thread é responsavel pela receção de pacotes e diferenciação dos mesmos, isto é, esta thread recebe todo o tipo de pacotes e realiza o parse dos mesmos.Estes pacotes podems ser provenientes do servidor e dirigidos à aplicação do cliente em questao, assim como podem ser pacotes provenientes de um client vizinho, isto é packets contendo input de outro utilizador, ou packets provenientes do gateway, que foram despejados no cliente em questao para se possivel serem transmitidos a esse mesmo cliente (Store carry and forward)

Thread send Esta thread é responsável pelo envio de mensagens provenientes da aplicação do cliente, estas mensagens sao direcionadas ao gateway.

inputViz InputViz é um buffer desenhado com o objetivo de guardar em buffer os inputs que nao conseguiram ser enviados ao Gateway por falta de conexao, este buffer é utilizado para , assim que exista uma conexao com um nodo vizinho, os dados sejam enviados com o intuito de prosteriormente chegarem ao gateway.

Neighbor Buffer Neighbor Buffer como o nome indica é um buffer que guarda os dados provenientes do gateway e que nao sao relacionados com o cliente em questao. Estes dados sao prosteriormente transmitidos assim que é recebido um pedido multicast por eles.

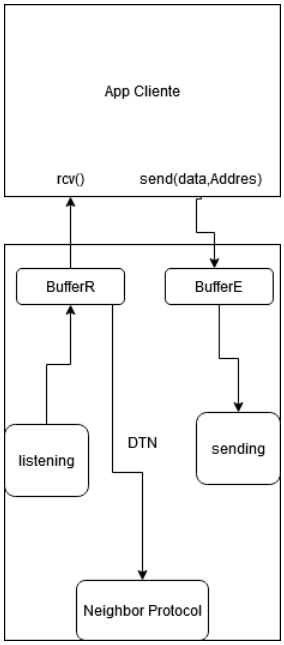

Fig. 7. Arquitetura Cliente Com modulo DTN

3.5 Bibliotecas de funções

Como bibliotecas de funcoes, decidimos utilizar o Pickle, responsavel por encapsular e transformar dados em bytes, para prosteriormente serem enviados. Para além desta, fizemos tambem uso de um script multicast.Este script continha um exemplo de envio e receção em ipv6 multicast. Este script foi modificado e reutilizado no modulo DTN, no processo de descoberta de vizinhos.

4 Testes e resultados

De forma a testar modulo DTN, começamos por correr o jogo na topologia anteriormente preparada.Após o inicio do jogo, o jogador movel com o ip *2001:6::2* afastou-se do gateway perdendo conectividade.

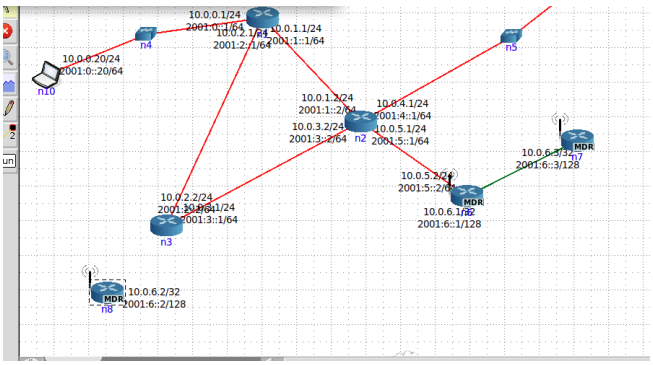

Fig. 8. Topologia com nodo desconectado

Com isto, podemos ver, através da analise da figura 9 , que os dois terminais apresentam countdowns diferentes, o que representa que o que tem o countdown mais alto se apresenta desconectado.

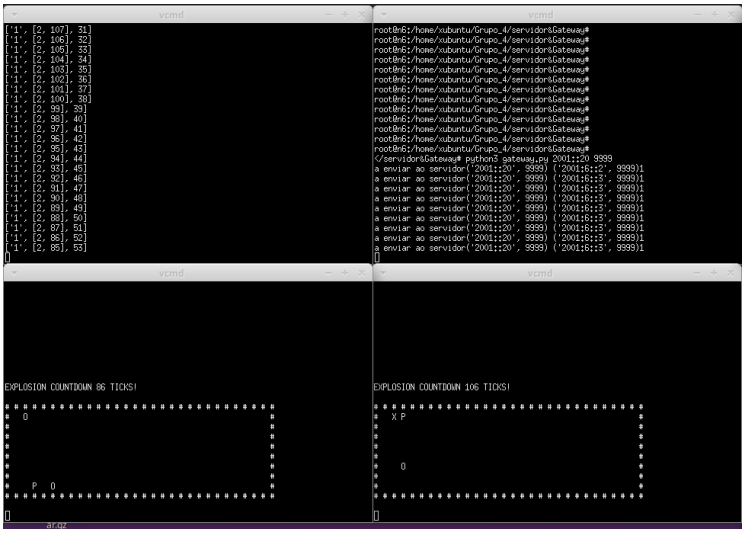

Fig. 9. Clientes moveis

4.1 Servidor → Cliente

Enquanto o utilizador *2001:6::2* se encontra desconectado, todas as mensagens que chegam ao gateway para este mesmo cliente, serão guardadas em cache. Caso esteja outro cliente ligado ao gateway, as mensagens do cliente desconectado serão passadas para este. Assim que o cliente com mensagens do servidor em cache se aproxima do cliente desconectado, estas mensagens São transmitidas, assim como as mensagens de input do cliente desconectado para o cliente que se aproximou.

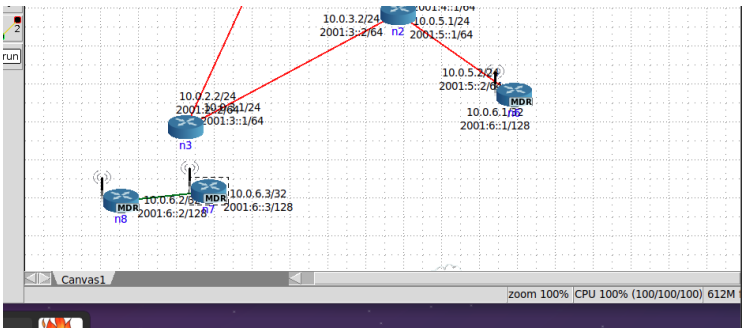

Fig. 10. Aproximação de clientes

Fig. 11. Troca de mensagens

4.2 Cliente → Servidor

Quando o cliente *2001:6::3* que foi entregar as mensagens do servidor ao Cliente regressa e volta a conectar-se ao gateway, as mensagens de input que o Cliente desconectado lhe enviou (*2001:6::2*) São reenviadas ao gateway.Atualizando a peca do jogador desconectado.

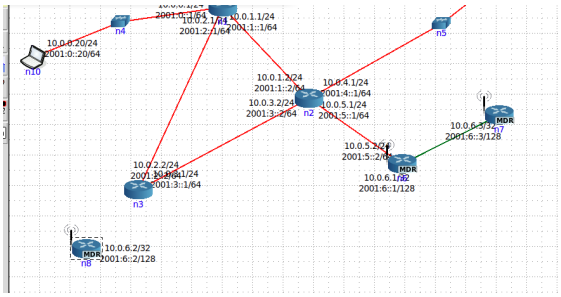

Fig. 12. Mensagens Input *2001:6::2* reenviadas por Outro Cliente

Como podemos ver na figura 13, o terminal da direita representa o utilizador desconectado, que tem a sua peca no topo do tabuleiro, por outro lado, o terminal da esquerda que se encontra atualizado com o servidor, apresenta a peca do utilizador da direita mais abaixo, o que significa que o utilizador desconectado se moveu, ainda que em nenhum momento se tenha conectado diretamente ao servidor.

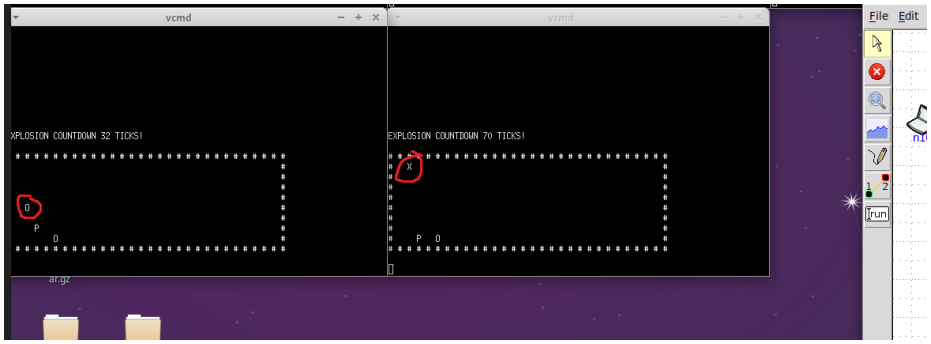

Fig. 13. Atualização das mensagens do client desconectado

5 Conclusões e trabalho futuro

O desenvolvimento deste projeto permitiu-nos aumentar a capacidade de interpretação, desenvolvimento e implementação de um algoritmo DTN, aplicada a um jogo multiplayer.

Como trabalho futuro, gostaríamos de salientar a falta de escalabilidade presente no projeto, onde a eficiência de resposta está dependente da aproximação dos clientes ao gateway, assim como, a existência de 1 gateway, proporciona no caso de um numero elevadíssimo de mensagens, algum delay.Com isto,para um trabalho futuro seria expectável a permissão de mais do que um gateway de forma a aumentar a probabilidade de entrega e talvez dividir a carga de trabalho presente no gateway em dois.Para além disto, caso o jogo tivesse um numero maior de utilizadores, seria bom implementar uma metrica de distribuição de pacotes em cache pelo gateway, de forma a minimizar o trafego da rede, tentando manter sempre a probabilidade de entrega o mais alta possível.

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. pickle Homepage, https://docs.python.org/3/library/pickle.html.
3. Delay-Tolerant Networking Architecture, https://datatracker.ietf.org/doc/html/rfc4838.
4. DTN Nasa website, https://www.nasa.gov/directorates/heo/scan/engineering/technology/disruption_tolerant_netwo