PTTA - Protocolo de Transporte Tolerante a Atrasos

Fábio Luiz Pessoa Albini, Anelise Munaretto e Mauro Fonseca

***Resumo*— O presente trabalho consiste na proposta de um protocolo de transporte para redes tolerantes a atrasos e desconexões (PTTA). Este protocolo tem o objetivo de oferecer uma maior taxa de entrega das informações em um menor tempo. Para isso, serão utilizados Códigos Fontanais como técnica de correção de erros. Os resultados mostram as vantagens da utilização do protocolo proposto.**

***Palavras-Chave*— Redes tolerantes ao atraso, técnicas de codificação, códigos fontanais, The ONE.**

Abstract*— *This work consists in a novel transport protocol proposal for Delay-Tolerant Networks and Disruption Tolerant Networks (both DTN). This protocol aims to offer better information delivery rate in less time. For this, fountain codes will be used as an error correcting codes technique. The results shown the achieved advantages using the PTTA - Delay Tolerant Transport Protocol (in portuguese* -Protocolo de Transporte Tolerante a Atrasos*).

Keywords*— *Delay tolerant networks, network coding, fountain codes, The ONE.

I. INTRODUÇÃO

A Internet, com seus protocolos e camadas, é uma maneira eficiente de realizar a comunicação entre diversos dispositivos e em diferentes cenários. Porém, existem alguns tipos de redes que tornam os protocolos da Internet inadequados para a realização da comunicação entre os nós. Alguns exemplos destes ambientes são: comunicação entre dispositivos móveis, rurais, submarinas, redes de sensores, entre outros que compõem os nomeados ambientes desafiadores [1]. Estes ambientes desafiadores, em geral, possuem dificuldade em manter uma comunicação fim-a-fim com baixa latência e perda de pacotes. Devido a estas características, estas redes foram denominadas Redes Tolerantes a Atrasos e Desconexões (*Delay-Tolerant Networks* ou *Disruption Tolerant Networks*, ambas com a sigla DTN) [1] [2].

Os protocolos de transporte mais utilizados em redes conectadas são: UDP (*User Datagram Protocol*) e TCP (*Transmission Control Protocol*). O UDP oferece um serviço sem conexão e sem confirmação para a entrega de dados, sendo assim, sem garantia de entrega. Por outro lado, o TCP proporciona um serviço orientado a conexão e com confirmação, afim de assegurar a entrega das informações. Para obter um serviço de entrega com confiabilidade, usando este protocolo, é preciso existir ao menos um caminho estável entre origem e destino no estabelecimento da conexão para o envio de dados. As redes tolerantes a atrasos e interrupções geralmente são esparsas e desconexas. Nestas redes é comum a inexistência de um caminho estável entre origem e destino, sendo muitas vezes impossível o estabelecimento de conexões para envio dos dados e confirmações [3]. Por este motivo o TCP não é funcional em redes tolerantes a atrasos [1].

A dificuldade neste caso é desenvolver um mecanismo eficaz de transmissão das informações, sem a necessidade de confirmação e conexão entre os hospedeiros (origem e destino). Uma solução seria o uso do UDP que, como descrito anteriormente, é um protocolo de transporte não orientado a conexão, sem confirmação e garantia de entrega das informações e ordenação de pacotes. Mas essa solução não oferece garantia no recebimento das informações, o que para várias aplicações é imprescindível. Outra forma seria realizar o envio sem conexão (como no UDP), mas combinando alguma técnica que verifique os dados faltantes e consiga reaver as informações não recebidas sem o envio de mensagens para a origem. Neste contexto, uma opção é o envio de informações redundantes na rede, para que os dados faltantes possam ser recuperados de uma forma alternativa. Porém, ao enviar informações redundantes é preciso uma política de distribuição das informações na rede, para que estas informações estejam distribuídas da melhor maneira possível. Este problema é provado pelo trabalho [4] como sendo um problema *NP-hard*, o qual afirma que o sucesso da distribuição é dependente da forma com que serão alocadas as réplicas na rede. Para evitar este transtorno de alocação das réplicas os autores de [5] sugerem o uso de uma técnica de códigos corretores de erros chamada *Fountain Codes* [6] (Códigos Fontanais (FC)).

O nome, códigos fontanais, advém da analogia com uma fonte que jorra gotas incessantemente. Estas gotas são as informações codificadas, onde a fonte é o nó de origem. A idéia é que o destinatário que receberá as informações "segure" um recipiente (também chamado copo) próximo à fonte, ou seja, capture gotas até um limite que é o tamanho desse copo, que quando cheio terá a quantidade suficiente de informações para realizar a decodificação de toda a informação original [6].

Ao utilizar os códigos fontanais obtém-se algumas vantagens, sendo elas: a possibilidade quase que infinita de geração de "gotas"; a liberdade de se controlar em tempo real quantas "gotas" serão geradas; a velocidade e baixo uso de processamento, visto que, em geral, este se resume em funções XOR (ou-exclusivo); a independência de quais serão as gotas recebidas para se poder realizar a decodificação; a garantia do recebimento das informações de maneira correta; a segurança das informações, pois os dados são enviados codificados e podem ser criptografados anteriormente; a não necessidade de confirmação do recebimento das "gotas" visto que não importam quais foram perdidas, as informações poderão ser recuperadas com praticamente qualquer combinação delas; não necessita ordenação uma vez que as "gotas" capturadas serão depois processadas para remontar a informação original;

Fábio Luiz Pessoa Albini, Anelise Munaretto¸ Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI), Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, Brasil, E-mails: fabioalbini@yahoo.com, anelise@utfpr.edu.br. Mauro Fonseca, Programa de Pós-graduação em Informática Aplicada (PPGIA), Pontifícia Universidade Católica do Paraná (PUC-PR), Curitiba, Brasil, E-mail: maurofonseca@pucpr.edu.br.

o envio de mensagens *multicast* torna-se muito mais fácil e eficiente, pois as informações podem ficar "jorrando" da origem o tempo que for necessário para a informação ser disseminada; o destino necessita apenas de cerca de 5% [7] de informação redundante (a mais) para decodificar a informação original.

Neste contexto, o objetivo deste artigo é propor um protocolo de transporte para redes tolerantes a atrasos e interrupções, confiável, sem conexão e confirmação. Para isso serão utilizadas técnicas de codificação através dos códigos fontanais. A eficiência da proposta será validada com simulações realizadas no simulador The ONE [8] em um cenário descrito adiante.

O restante deste artigo está organizado da seguinte maneira: a seção II apresenta os trabalhos relacionados, a seção III mostra a formalização do protocolo de transporte criado com a descrição de todos os campos do cabeçalho, o seu diagrama esquemático e o seu funcionamento, a seção IV é composta pela descrição do cenário de avaliação da proposta, a seção V os resultados e discussão dos resultados são apresentados com a avaliação do desempenho e a seção VI traz as conclusões e trabalhos futuros.

II. Trabalhos Relacionados

O uso de códigos corretores de erros em DTNs já tem sido objeto de estudo [9], [4], [5] e [10].

Em [9] o ganho na performance com códigos corretores de erros é comparado à replicação simples, ou seja, o envio de replicas adicionais da mesma mensagem. Para comprovar os benefícios de utilizar os códigos corretores de erros, são realizadas várias simulações, usando vários protocolos de roteamento.

Em [4] é levado em conta o problema de padrões não uniformes de encontros entre os nós. Além disso, é mostrado que existe uma dependência muito forte da probabilidade de entrega das informações com a distribuição das réplicas pelos vários caminhos da rede. Os autores avaliam várias técnicas de alocação dessas réplicas e é provado que o problema de alocação das informações na rede é do tipo *NP-hard*.

Em [5] foi descrito o formalismo estatístico do desempenho das DTNs. Ainda são levados em conta o uso dos códigos corretores de erros e dos códigos fontanais neste formalismo. Este é o primeiro trabalho que os autores possuem conhecimento e comenta a respeito do uso de códigos fontanais em DTN.

Em [10] é proposta a integração dos códigos fontanais com o protocolo *Optimal Probabilistic Forwarding* (OPF). Os dados são codificados e uma regra de encaminhamento é definida para decidir para onde um pacote será enviado.

Como foi possível observar, até o momento, os autores não encontraram na literatura, proposta semelhante a contida neste trabalho, o que caracteriza a sua inovação e singularidade.

III. Formalização do Protocolo

Para evitar equívocos e ambiguidade deve-se definir alguns termos que serão utilizados no decorrer deste documento. Quando o texto se referir a **Dados**, este representa as informações originais a serem enviadas do nó origem ao destino, o termo **Segmento** ou **Gota** se refere às informações originais já codificadas, ou seja, aos fragmentos de dados após sofrerem a codificação que será explicada a seguir e por fim **Pacote** será o termo utilizado para o segmento já encapsulado, ou seja, o segmento em conjunto com os dados do cabeçalho do PTTA - Protocolo de Transporte Tolerante a Atrasos.

A seguir será descrito o formalismo pertinente ao protocolo de transporte proposto.

A. Cabeçalho

Em primeiro lugar, como este protocolo é um protocolo da camada de transporte, ele deve funcionar como multiplexador e demultiplexador para aplicativos. Por isso é extremamente necessário o uso das portas na fonte e destino, para saber a qual aplicativo estes dados são destinados e de qual aplicativo eles estão sendo originados.

Para conseguir uma certa garantia e confiabilidade na informação recebida é necessário um campo *checksum* que possui as informações necessárias para a verificação da integridade dos dados do pacote.

Para se realizar a decodificação das informações é necessário conhecer quais são as partes dos dados originais envolvidos em cada um dos segmentos. Portanto, cada segmento deve carregar consigo alguma forma de identificação dessas partes. Por isso foi criado o campo Mapa de Partes com 32 bits. Entretanto, com 32 bits pode-se representar 32 partes que podem estar envolvidas no segmento atual. Mas podem existir situações onde os dados originais tenham que ser divididos em um número maior que 32 partes. Como podem existir várias partes e o campo reservado no cabeçalho possui um tamanho fixo, pode ocorrer do número de bits disponível para representa-las (32 bits no campo Mapa de Partes) ser muito pequeno, o que ocasionaria conflito nesta representação e impossibilitaria a recuperação dos dados. Dessa forma, é necessário aumentar o número de bits disponíveis para essa informação (Mapa de Partes). Com esse intuito, foi criado o campo Tamanho do Mapa de Partes (com 16 bits) que representa quantos grupos de 32 bits serão utilizados no campo Mapa de Partes, o que aumenta significativamente a representação para se gerar as gotas.

Finalmente, o campo Dados possui realmente o segmento que está sendo carregado da origem para o destino.

A Figura 1 mostra todos esses campos que formam o cabeçalho do protocolo desenvolvido.

Porta da Fonte	Porta do Destino
Checksum	Tamanho do mapa de partes
Mapa de partes	
Dados	

16 bits	16bits

Fig. 1. Figura esquemática com os campos do cabeçalho do protocolo de transporte apresentado.

Porta da Fonte: (16 bits) Número da porta de origem.
Porta do Destino: (16 bits) Número da porta do destino.

Checksum: (16 bits) Este campo contém o complemento de um da soma de todas as palavras de 16 bits pertinentes ao cabeçalho e ao campo Dados. Se um segmento tiver um número ímpar de octetos, o último deverá ser completado com zeros, sendo que estes zeros não serão transmitidos no campo dos dados.

Tamanho do Mapa de Partes: (16 bits) Este campo representa quantos conjuntos de 32 bits formam o campo Mapa de Partes e foram utilizados para a geração dos dados codificados pertinentes a este segmento. Este campo possui um valor inteiro.

Mapa de Partes: (Múltiplo de 32 bits) Corresponde aos dados que foram utilizados para a geração deste segmento, esta informação é necessária para a recuperação das dados originais.

Dados: (Variável) O segmento a ser transmitido.

B. Funcionamento do Protocolo

O Protocolo de Transporte Tolerante a Atrasos possui o seu funcionamento segundo o algoritmo abaixo:

Algoritmo 1: Protocolo de Transporte Tolerante a Atrasos

```
Dividir os dados em n partes;
enquanto Fonte estiver aberta faça
    Selecionar aleatoriamente x partes de dados;
    Realizar a operação XOR entre as partes
    selecionadas;
    Montar o pacote com a porta da fonte, porta do
    destino, tamanho do mapa de partes, mapa de partes
    e o resultado da operação XOR no campo dados;
    Enviar o pacote para a camada inferior;
fim enqto
```

Para facilitar a visualização do processo descrito no algoritmo precedente este foi ilustrado na Figura 2. Nesta é possível visualizar que os dados foram divididos em *n* partes. Estas partes foram combinadas para formar as "gotas" que representam o resultado da operação *XOR* entre os bits das partes combinadas. Esse resultado é utilizado no campo dados do pacote. Pela imagem ainda observa-se que existe uma quantidade muito grande de combinações entre as partes para a geração das gotas.

Para montar o pacote é utilizado o valor da porta de origem e destino, é calculado o *checksum*, é inserido o tamanho do mapa de partes, no exemplo, este campo possui o valor 1 pois, o mapa de partes é composto por apenas um conjunto de 32 bits. Por fim, o mapa de partes é montado sendo este composto pelo valor 1 nos bits que representam as partes utilizadas para a geração da gota que será utilizada neste pacote e o campo dados é composto pelo resultado da operação *XOR* entre as partes envolvidas nesta gota.

IV. Cenário Experimental

Para realizar a avaliação da performance da proposta foi utilizado o simulador The ONE [8] com o cenário de Helsink. Este possuindo 126 nós divididos em ônibus, carros, pedestres

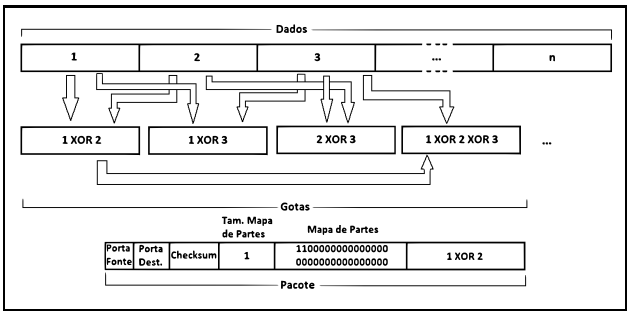

Fig. 2. Ilustração do funcionamento do PTTA.

e *tramways*. Cada um desses tipos compostos pela sua própria velocidade, característica de movimento, caminho permitido para o seu movimento e *buffers*. O caminho permitido para o movimento de cada um dos tipos é importante por existirem ruas com sentido único, onde os pedestres podem se locomover em ambos os sentidos mas os carros não. Ainda existem caminhos que, por exemplo, apenas os *tramways* podem se deslocar. Além disso, todos esses nós operam com uma interface de comunicação com taxas de transmissão de 250KBps.

Em cada simulação havia apenas um nó gerador de mensagens (origem) e todas eram destinadas a um mesmo hospedeiro (destino), simulando a transferência de dados entre dois nós. Estes nós foram selecionados aleatoriamente e a cada simulação esta seleção se deu com base na semente passada por parâmetro, o que permite a reprodução dos resultados. A simulação foi executada até que toda a informação pudesse ser decodificada no receptor ou o tempo limite (aproximadamente 12 horas) da simulação fosse atingido.

Vários tamanhos de dados foram testados. Para simular o envio de pequenos arquivos foi utilizada uma informação de 5MB o que poderia ser uma música em MP3 ou algum MMS. Para uma quantidade intermediária de dados foi utilizado um tamanho de 10MB simbolizando um vídeo pequeno. Para arquivos médios foi utilizada uma informação de 50MB simbolizando um aplicativo, algumas fotos ou até mesmo arquivos pessoais. E, para grandes informações foi utilizado um conjunto de dados de 100MB representando a transferência de arquivos de Backup. Para cada um desses tamanhos foram realizadas 100 simulações com sementes aleatórias distintas. Com isso, todos os resultados apresentados serão médias dos resultados das 100 simulações com intervalos de confiança de 95%.

Os pacotes que trafegavam na rede tiveram seu tamanho estipulados em 500KB. Além disso, foi utilizado o protocolo de roteamento epidêmico [11] conforme implementado no simulador.

V. Avaliação de Desempenho

Para avaliar a proposta do uso do protocolo de transporte desenvolvido, os testes foram divididos em dois passos, no primeiro foi realizada a transferência dos dados sem o uso do protocolo de transporte desenvolvido, também chamados sem FC, ou seja, as informações foram segmentadas e enviadas pela rede utilizando-se do protocolo de rede epidêmico (cada fragmento foi enviado uma única vez, sem TTL - *Time To*

Live). No segundo passo foi efetuada a transferência com o uso do protocolo de transporte desenvolvido, também chamados com FC, onde as informações foram codificadas e em seguida enviadas através da rede sem fechamento da fonte e utilizando-se do mesmo protocolo de rede (epidêmico) do primeiro passo. Foi utilizado o cenário DTN descrito na seção IV. Em todos os testes os parâmetros de entrada eram iguais com relação às configurações da rede.

A Figura 3 apresenta o resultado da quantidade de informações recebidas em função tempo para os diferentes tamanhos de dados testados, os intervalos de confiança foram omitidos por motivo de clareza.

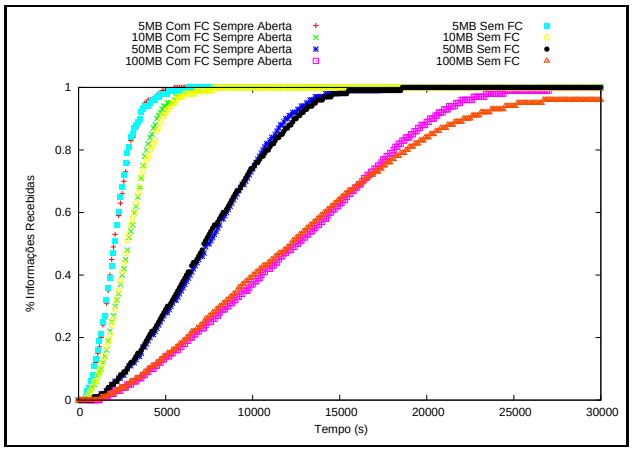

Fig. 3. Quantidade de informações recebidas por tempo de simulação.

Em todos os casos testados, os resultados com o uso do protocolo de transporte proposto obtiveram um melhor resultado em relação aos que não utilizam o protocolo, significando que com as técnicas de códigos fontanais se obteve uma maior eficiência na entrega das informações. Isso ocorre porque ao utilizar o protocolo de transporte descrito não importa quais pacotes foram recebidos para realizar a decodificação das informações, no momento em que o destino receber a quantidade suficiente de segmentos, as informações originais serão recuperadas. Por outro lado, quando não se utiliza do protocolo descrito, as informações originais só podem ser inteiramente recebidas se todas as partes da informação segmentada for recebida. Neste aspecto, no cenário onde não é utilizado o protocolo de transporte proposto, pode ocorrer de um dos fragmentos da informação ficar "preso" em alguma área da rede que venha a ficar incomunicável. Isso faria com que a informação inteira ficasse inutilizada até que este fragmento fosse recebido, como ocorreu nas simulações de números 52, 53, 55, 56, 57, 58, 59, 62 e 63 para os dados com tamanho de 100MB e sem o uso dos códigos fontanais. Isso justifica o motivo da Figura 3 para tamanho de 100MB e sem o uso do protocolo de transporte apresentar um máximo que não atinge 100% no recebimento das informações. É importante comentar que quando se utiliza do protocolo de transporte, este problema não impossibilita o recebimento da informação, pois as informações deste fragmento estarão disponíveis em outros pacotes como redundância.

Ainda na Figura 3 é possível perceber que para arquivos pequenos, como os de 5MB, a diferença no tempo gasto para se entregar as informações nas duas situações testadas é muito menor (cerca de apenas 5%) que a mesma diferença para arquivos maiores como os de 50MB e 100MB que foram de cerca de 10,5% e 11,5%.

Após ter sido realizado o teste sem o fechamento da fonte foram verificados quais eram as últimas mensagens recebidas pelos destinos para se verificar a possibilidade de fechamento da fonte com o objetivo de reduzir a quantidade de pacotes que trafegam na rede. Este resultado é apresentado na Tabela I.

TABELA I

NÚMERO DA ÚLTIMA MENSAGEM RECEBIDA EM MÉDIA E COM INTERVALO DE CONFIANÇA DE 95% NO TESTE COM A FONTE SEMPRE ABERTA.

Tamanho	Mínimo	Médio	Máximo
5MB	46	51	56
10MB	77	83	90
50MB	332	346	360
100MB	643	663	684

A partir dos resultados da Tabela I foram realizados novos testes com o fechamento da fonte utilizando-se o valor máximo do intervalo de confiança por se ter 95% de garantia que a média encontra-se abaixo desse valor. O resultado deste experimento foi praticamente o mesmo, porém com um número de mensagens geradas bem menor. Este resultado pode ser observado na Tabela II.

TABELA II

NÚMERO DE PACOTES CRIADOS EM MÉDIA.

Tamanho	Sempre Aberta	Temporizada	Diferença
5MB	120	56	53,3%
10MB	174	90	48,3%
50MB	500	360	28,0%
100MB	866	684	21,0%

Após, foi realizada a comparação dos resultados obtidos pelos testes utilizando a fonte sempre aberta e a fonte temporizada e o resultado é a Figura 4. Como é possível perceber, a variação no tempo de entrega dos dados foi pequena, quase imperceptível através do gráfico.

Após isso, foi realizado mais um teste utilizando um tempo menor para o fechamento da fonte. Esse tempo foi estipulado com base no limite inferior da média da última mensagem recebida no teste com a fonte temporizada. Novamente, foi utilizada como base a média com intervalo de confiança de 95% e o resultado está apresentado na Tabela III.

TABELA III

NÚMERO DE PACOTES CRIADOS NO SEGUNDO TESTE COM O FECHAMENTO DA FONTE.

Tamanho	Mínimo	Médio	Máximo
5MB	37	40	55
10MB	69	72	75
50MB	309	316	324
100MB	615	626	636

Agora a diferença quanto ao número de mensagens criadas na rede no caso da fonte sempre aberta e neste segundo teste

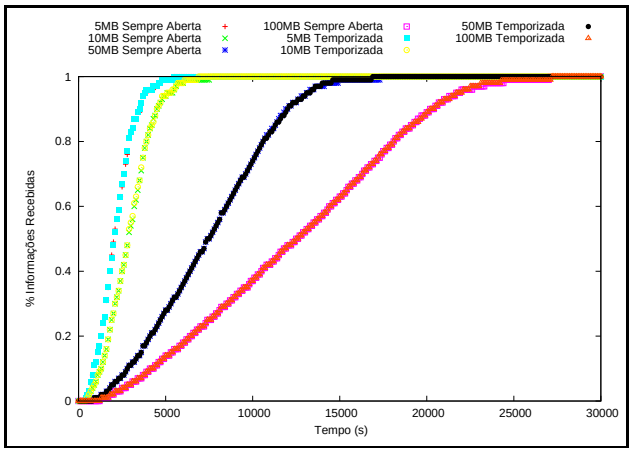

Fig. 4. Comparativo da quantidade de informações recebidas por tempo de simulação onde a fonte está sempre aberta e para a fonte temporizada.

com o fechamento da fonte foi de 69,2%, 60,0%, 38,2% e 29,0% menos mensagens criadas na rede, para os tamanhos 5MB, 10MB, 50MB e 100MB, respectivamente.

Foi verificado que com o fechamento da fonte neste segundo teste o ganho no tempo de entrega das mensagens foi ligeiramente maior como pode ser verificado na Figura 5 com os gráficos para os testes sem códigos fontanais e para esta segunda fonte temporizada, onde a maior diferença é visível nos gráficos de 50MB de dados.

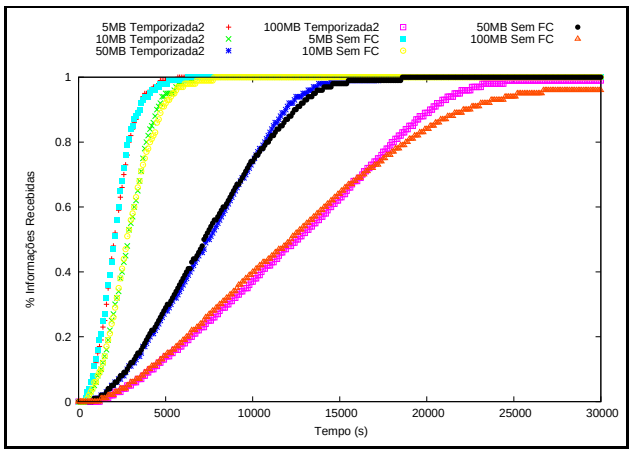

Fig. 5. Informações recebidas por tempo de simulação para o caso sem códigos fontanais e para a segunda fonte temporizada.

Os resultados mostram que o uso do protocolo de transporte é promissor. Isso pode ser verificado pelo ganho obtido no tempo de entrega das informações, como mostrado nas Figuras 5 e 3. Com esse ganho é possível verificar um aumento na taxa de entrega, uma vez que em todos os testes realizados, com o uso dos códigos fontanais, foram entregues 100% das informações originais, já nas simulações sem o uso do PTTA, isso não aconteceu.

VI. Conclusões e Trabalhos Futuros

Neste artigo foi proposto um protocolo de transporte para redes tolerantes a atrasos e interrupções. Este protocolo se utiliza das técnicas dos códigos fontanais para aumentar a taxa de entrega das informações sem o uso de conexão e confirmação no recebimento das informações. Além disso, foram realizados testes com o simulador The ONE para validação e verificação dos impactos no uso desse novo protocolo. Através das simulações foi mostrado que o uso do protocolo demonstra vantagens no recebimento dos dados diminuindo o tempo de entrega. Ainda foi possível concluir que de acordo com os testes realizados, os benefícios são maiores conforme o volume dos dados a serem transmitidos aumenta. As principais vantagens no uso do protocolo criado são: a não necessidade de confirmação; a ausência de reenvios; a flexibilidade de controle na geração das informações codificadas, e; a necessidade de apenas cerca de 5% de informação redundante recebida para se realizar a decodificação. Com os resultados foi verificado que o uso dos códigos fontanais é uma alternativa interessante para solucionar o problema da não existência de confiabilidade em DTNs.

Referências

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2003, pp. 27–34.

[2] C. T. d. Oliveira, D. M. Taveira, and O. C. M. B. Braga, R. B. andDuarte, "Uma proposta de roteamento probabilístico para redes tolerantes a atrasos e desconexões," *XXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, SBRC.*, pp. 735–748, 2008.

[3] Q. Zhang, Z. Jin, Z. Zhang, and Y. Shu, "Network coding for applications in the delay tolerant network (dtn)," in *Proceedings of the 2009 Fifth International Conference on Mobile Ad-hoc and Sensor Networks*, ser. MSN '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 376–380. [Online]. Available: http://dx.doi.org/10.1109/MSN.2009.68

[4] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using redundancy to cope with failures in a delay tolerant network," *SIGCOMM Comput. Commun. Rev.*, vol. 35, pp. 109–120, August 2005. [Online]. Available: http://doi.acm.org/10.1145/1090191.1080106

[5] E. Altman and F. De Pellegrini, "Forward correction and fountain codes in delay tolerant networks," in *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*. IEEE, Apr. 2009, pp. 1899–1907. [Online]. Available: http://dx.doi.org/10.1109/INFCOM.2009.5062111

[6] D. J. C. Mackay, "Fountain codes," *Communications, IEE Proceedings-*, vol. 152, no. 6, pp. 1062–1068, 2005.

[7] Mackay, D. J. C., *Information theory, inference, and learning algorithms*, 1st ed. Cambridge University Press, June 2003.

[8] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. New York, NY, USA: ICST, 2009.

[9] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-coding based routing for opportunistic networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 229–236. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080140

[10] Y. Dai, P. Yang, G. Chen, and J. Wu, "Cfp: Integration of fountain codes and optimal probabilistic forwarding in dtns," in *GLOBECOM*. IEEE, 2010, pp. 1–5.

[11] A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks," 2000.