QoN++: Roteamento por Qualidade dos Nós com redução de Overhead em Redes DTN.

William Castro da Rosa, Edjair de Souza Mota, José Ferreira de Rezende e Celso Barbosa Carvalho

***Resumo* – Este artigo apresenta o QoN++, uma extensão do algoritmo QoN-ASW que gerencia informações contidas em *buffers* de dados dos nós de uma rede tolerante a atrasos e desconexões (DTN) e ajusta a estratégia de encaminhamento de dados de acordo com a qualidade dos nós da rede. O QoN++ pode aumentar as taxas de entrega ao custo de um pequeno acréscimo no atraso médio. Em nossa proposta, destaca-se a redução da sobrecarga em relação à quantidade de mensagens transmitidas na rede, que pode passar dos 50% em alguns casos.**

***Palavras-chave* – Qualidade dos nós, sobrecarga, roteamento, DTN**

***Abstract* – This paper presents QoN++, an extension of the QoN-ASW algorithm that manages information contained in data buffers of nodes in a delay and disruption-tolerant network (DTN) and adjusts the data relay strategy according to the quality of the network nodes. QoN++ can increase delivery rates at the cost of a small increase in the average delay. In our proposal, it is highlighted the reduction of overhead related to the amount of messages transmitted on the network, which can exceed 50% in some cases.**

***Keywords* – Quality of nodes, overhead, routing, DTN**

I. INTRODUÇÃO

A evolução de sistemas embarcados permite processar e compartilhar dados entre dispositivos conectados em rede. A Internet das Coisas (*Internet of Things* – IoT) mescla tecnologias de análise em tempo real, sensoriamento, aprendizagem de máquina e gerenciamento de recursos de rede, oferecendo soluções a partir do uso de dispositivos inteligentes. Dentre as aplicações de IoT, mencionam-se: interconexão de redes de sensores sem fio [1]; arquitetura de cidades inteligentes [2]; monitoramento médico remoto [3] e sua redução de gastos financeiros [4]; IIoT (*Industrial Internet of Things*) [5] e seu potencial para gerar trilhões de dólares em produtos internos brutos [6]; controle de irrigação e fertilização [7]; controle de tráfego de veículos [8]; entre outras.

Muitos dispositivos inteligentes são móveis, logo a conectividade é intermitente, característica típica das redes tolerantes a atrasos e desconexões (*Delay and Disruption-Tolerant Networks – DTN*). Nas DTNs, o estabelecimento de rotas fim-a-fim de protocolos como AODV [9] e DSR [10] é ineficaz. A abordagem adotada para contornar a instabilidade das conexões é conhecida como armazena e repassa (*store and forward*), na qual os dispositivos de uma rede, chamados de nós, armazenam temporariamente os dados recebidos na memória e os encaminham a outros nós em futuros contatos. Os critérios de encaminhamento variam entre os algoritmos de roteamento.

Alguns algoritmos são tradicionais na literatura especializada. Dentre eles, cita-se o Epidêmico [11], PRoPHET [12] e *Spray and Wait* (SnW) [13]. Em geral, algoritmos de roteamento mais recentes são modificações ou extensões de modelos mais antigos. Destaca-se o novo QoN - ASW [14], que aplica o conceito de qualidade dos nós (*Quality of Nodes - QoN*) e apresenta um melhor balanço entre taxa de entrega, atraso médio e sobrecarga (*overhead*) na entrega de mensagens.

O QoN-ASW é uma modificação do SnW em que um dispositivo utiliza funções de probabilidade, similares às do PRoPHET, para calcular o número de cópias de uma mensagem a ser transmitida entre nós da rede. Tais funções se baseiam no histórico de contatos entre nós e têm como objetivo aumentar a taxa de entrega de mensagens enquanto limita o encaminhamento excessivo de mensagens.

O objetivo deste artigo é apresentar uma proposta, denominada QoN++, que adapta a estratégia de encaminhamentos do QoN-ASW em função do valor de QoN e que utiliza o histórico de mensagens entregues no gerenciamento de *buffer*. O algoritmo proposto foi implementado e avaliado a partir de experimentos com a ferramenta *ONE* [15]. Os resultados das simulações mostram um aumento da taxa de entrega de até 10% e redução da sobrecarga de até 65%, com pequeno acréscimo no atraso médio quando comparado ao algoritmo QoN-ASW.

Para apresentar nossa proposta, o artigo é dividido em seções conforme a seguir: i) a Seção II descreve o algoritmo QoN-ASW; ii) a Seção III apresenta a nossa proposta, o QoN++; iii) a Seção IV mostra os resultados e desempenhos; por fim, a Seção IV apresenta as conclusões.

II. ALGORITMO QoN-ASW

O algoritmo QoN-ASW utiliza *slots* de tempo de mesma duração. Quando dois nós (*i,j*) se encontram em um *slot*, *i* e *j* trocam mensagens entre si, calculam suas capacidades de repasse de mensagens no referido *slot* e atualizam suas probabilidades de encontros diretos e transitivos [14].

Na Eq. 1, a capacidade estimada do nó *i* no *slot k* ($EMHC_i^k$), de duração *TS,* é o número estimado de cópias de mensagens repassadas por *i* no referido *slot*, onde m_{cur} é o número de cópias de mensagens que *i* repassou com sucesso até o instante do cálculo, m_x é o número de cópias repassadas nos *slots x* anteriores ao *slot k* atual (*x* varia de 1 a *k-1*) e Δt é o tempo transcorrido desde o fim do último *slot*.

$$EMHC_i^k = \begin{cases} \frac{m_{cur}}{\Delta t} \,.\, TS, & k = 1 \\ \frac{m_{cur} - \sum_{x=1}^{k-1} m_x}{\Delta t} \,.\, TS, & k > 1 \end{cases} \quad (1)$$

Na Eq. 2, a capacidade do nó *i* no *slot k* (MHC_i^k) é o número normalizado de cópias que *i* repassou com êxito. A partir do segundo *slot* ($k > 1$), MHC_i^k é uma ponderação entre a capacidade do *slot* anterior (MHC_i^{k-1}) e a capacidade estimada do *slot* atual ($EMHC_i^k$), ajustada pelo fator $\alpha \in [0,1]$.

$$MHC_i^k = \begin{cases} EMHC_i^k, & k = 1 \\ \alpha . MHC_i^{k-1} + (1-\alpha) . EMHC_i^k, & k > 1 \end{cases} \quad (2)$$

Na Eq. 3, a função $U_{MHC(i)}$ do nó *i* denota sua utilidade relativa para entregar a mensagem ao destino, em comparação ao nó *j*, com quem *i* entrou em contato.

$$U_{MHC(i)} = \frac{MHC_i^k}{MHC_i^k + MHC_j^k} \quad (3)$$

Na Eq. 4, a força de conectividade $FC_{(i,j)}$ indica a razão entre o tempo de conexão total entre os nós (*i,j*) e o tempo total de simulação. O parâmetro *n* indica o número de conexões entre (*i,j*) no decorrer do tempo. Para cada conexão individual *l*, são registrados os instantes de início (S^l_{ij}) e de fim (E^l_{ij}). A diferença entre os instantes equivale à duração da conexão *l* entre (*i,j*). O somatório destas diferenças representa o tempo em que (*i,j*) estiveram conectados durante a simulação, de tempo total *T*.

$$FC_{(i,j)} = \frac{\sum_{l=1}^{n} \left(S_{ij}^l - E_{ij}^l \right)}{T} \quad (4)$$

O QoN-ASW utiliza funções de probabilidade semelhantes às do PRoPHET, com atualizações por contato direto, por contatos via nós intermediários (transitividade) e por envelhecimento [12]. Apenas a função de contato direto (Eq. 5) apresenta modificações na fórmula:

$$P_{(i,j)} = P_{(i,j)old} + \left(1 - P_{(i,j)old}\right) . P_{init} . \lambda^{FC_{(i,j)}} \quad (5)$$

$FC_{(i,j)}$ é utilizada em conjunto com o parâmetro λ, que corresponde à influência da força de conectividade sobre a probabilidade do nó *i* entregar uma cópia de mensagem ao nó *j*. P_{init} é o valor inicial fixo da probabilidade, configurado como P_{init} = 0,7 [14]. $P_{(i,j)old}$ é o valor da probabilidade após um período de envelhecimento e $P_{(i,j)}$ é o novo valor da probabilidade de contato direto após o cálculo.

Na Eq. 6, a função utilidade da probabilidade de entrega do nó *i*, $U_{P(i)}$, é uma relação entre a probabilidade de entrega de *i* ao destino *d* e a soma das probabilidades de entrega dos nós *i* e *j* ao referido destino.

$$U_{P(i)} = \frac{P_{(i,d)}}{P_{(i,d)} + P_{(j,d)}} \quad (6)$$

Portanto, após a etapa inicial de contato entre os nós *i* e *j*, eles devem calcular MHC_i^k, MHC_j^k e $P_{(i,j)}$.

Em seguida, o nó *i* verifica as mensagens contidas em seu *buffer* que não estejam presentes no *buffer* do nó *j*. Para cada mensagem, destinada a um nó de destino *d*, são analisadas algumas possibilidades. Se o nó *j* for o nó de destino da mensagem, esta é entregue diretamente e eliminada do *buffer* de *i*.

Caso contrário, verifica-se o número de cópias da mensagem. Se este número for maior que 1, o nó *i* calcula a sua Qualidade (*QoN*) a partir das funções utilidade U_{MHC} e U_P. O fator η pondera a importância das duas funções.

$$QoN_i = \eta . U_{MHC(i)} + (1 - \eta) . U_{P(i)} \quad (7)$$

O valor de QoN_i determina o número de cópias repassadas a *j* e mantidas no *buffer* de *i*. Quando o algoritmo opera neste modo, diz-se que ele está no modo *Spray*. Para uma dada mensagem m_l, o número de cópias mantidas no *buffer* de *i* é dado por Li_{new}, enquanto o número de cópias repassadas a *j* é dado por Lj_{new}.

$$L_{i_{new}}(m_l) = \left\lfloor QoN_i . L_{i_{old}}(m_l) \right\rfloor \quad (8)$$

$$L_{j_{new}}(m_l) = L_{i_{old}}(m_l) - L_{i_{new}}(m_l) \quad (9)$$

No entanto, se o número de cópias for igual a 1, o algoritmo entra no modo *Wait*. A probabilidade de contato com o nó destino *d* será usada como critério para determinar se a mensagem ficará armazenada no *buffer* de *i* ou *j*. Se *j* apresentar maior probabilidade de contato com *d* ($P_{(j,d)} > P_{(i,d)}$), *i* repassará a sua cópia a *j*. Caso contrário, *i* manterá sua cópia.

De forma semelhante, *j* realiza os mesmos procedimentos mencionados para o nó *i*. Assim, os nós *i* e *j* trocam mensagens entre si, utilizando recursos do PRoPHET e do SnW.

III. ALGORITMO QoN++

Criamos duas características adicionais em nossa proposta QoN++. A primeira consiste em divulgar a entrega e remover dos *buffers* dos demais nós da rede, as mensagens entregues aos nós de destino. Isso é realizado através do compartilhamento de tabelas de entregas entre contatos, sendo continuamente atualizadas a cada encontro. A segunda é o ajuste da estratégia de encaminhamento das cópias de mensagens em função do valor de QoN.

Tal ajuste está baseado na ideia de que um nó com uma qualidade muito baixa em relação a seus contatos pode não ser a melhor escolha para encaminhar a mensagem até o destino final, sendo útil apenas como um intermediário para futuros encaminhamentos para nós com qualidade mais elevada. Portanto, nós com essa finalidade não necessitam dispor de um grande número de cópias consumindo recursos de *buffer*. Desse modo, nós com elevada qualidade maximizam as chances de manter o maior número possível de cópias.

O Algoritmo 1 ilustra o QoN++. *F(i)* é uma tabela mantida pelo nó *i* com todas as mensagens cujas entregas finais já tenham sido informadas a *i*. Se o nó *i* já teve contato prévio com um dado nó *j*, *i* precisa saber se houve alterações em *F(j)*. Portanto, chamamos de $F(i,j)_{new}$ as informações mais recentes de *F(j)*, que estão sendo fornecidas de *j* para *i*. $F(i,j)_{old}$ são informações antigas de *F(j)*, a qual *i* teve acesso em algum momento prévio. Se *i* e *j* estão se encontrando pela primeira vez, $F(i,j)_{old}$ é nula.

Algoritmo 1 - QoN++

01: **se** nó *i* encontrar o nó *j*
02: **se** ($F(i,j)_{old}$ == nula) **ou** ($F(i,j)_{new}$!= $F(i,j)_{old}$)
03: **para** $\forall m_l \in F(j)$ e $m_l \notin F(i)$
04: *i* adiciona m_l a *F(i)*
05: **se** ($m_l \in i$)
06: *i* remove m_l de seu *buffer*
07: atualize MHC_i^k e MHC_j^k
08: atualize $P_{(i,j)}$
09: **para** $\forall m_l \in i$ e $m_l \notin j$

10: $L_{i_{old}}(m_l)$ = número de cópias de m_l
11: d = nó destino de m_l
12: **se** $(d == j)$
13: i transmite m_l a j
14: i adiciona m_l a $F(i)$
15: i remove m_l de seu *buffer*
16: **senão se** $(L_{i_{old}}(m_l) > 1)$
17: calcule QoN_i
18: **se** $(QoN_i < 0{,}3)$
19: $L_{i_{new}}(m_l) = 1$
20: $L_{j_{new}}(m_l) = L_{i_{old}}(m_l) - 1$
21: **senão se** $(QoN_i \geq 0{,}3)$ **e** $(QoN_i < 0{,}6)$
22: $L_{i_{new}}(m_l) = \lfloor QoN_i \; . \; L_{i_{old}}(m_l) \rfloor$
23: $L_{j_{new}}(m_l) = L_{i_{old}}(m_l) - L_{i_{new}}(m_l)$
24: **senão**
25: $L_{i_{new}}(m_l) = L_{i_{old}}(m_l) - 1$
26: $L_{j_{new}}(m_l) = 1$
27: **se** $(L_{j_{new}}(m_l) > 0)$
28: i repassa $L_{j_{new}}$ cópia de m_l a j
29: i guarda $L_{i_{new}}$ cópias de m_l para si
30: **senão se** $(L_{i_{old}}(m_l) == 1)$
31: **se** $(P_{(j,d)} > P_{(i,d)})$
32: i repassa m_l to j
33: i remove m_l de seu *buffer*

A atualização do *buffer* (linhas 2-6) ocorre antes de qualquer encaminhamento. Quando dois nós se encontram, eles compartilham uma lista de todas as mensagens cujas entregas finais já lhes foram informadas. Esse conhecimento é transmitido em cadeia por toda a rede. Em seguida, os nós atualizam suas capacidades e probabilidade de contato direto (linhas 7-8). Depois, para cada mensagem ainda presente em seus *buffers*, os nós analisam os destinos de cada mensagem e como proceder durante o encaminhamento (linhas 9-11).

A transmissão pode ser realizada diretamente ao nó de destino (linhas 12-15), o que acarretaria atualização da tabela *F(i)*. Caso contrário, analisa-se o número de cópias da mensagem. Se for superior a 1 (linhas 16-29), o nó transmissor está na fase *Spray* e pode: i) reter apenas uma única cópia da mensagem caso sua QoN for baixa (linhas 18-20); ii) recorrer a uma divisão balanceada pela QoN se sua qualidade for média (linhas 21-23); ou iii) repassar apenas uma cópia (linhas 24-26) no caso de uma QoN elevada. Em seguida, o encaminhamento de cópias é realizado (linhas 27-29).

No caso de uma única cópia da mensagem (linhas 30-33), o nó transmissor está na fase *Wait* e o encaminhamento é similar ao QoN-ASW.

IV. AVALIAÇÃO DE DESEMPENHO E RESULTADOS

O cenário de testes emprega o arquivo de trace coletado no IEEE Infocom 2006 [16]. Durante a conferência, 20 dispositivos estacionários e 78 móveis enviaram e receberam mensagens. Os demais dispositivos do arquivo Infocom foram usados para encaminhamentos. Utilizamos o simulador *ONE* [15], versão 1.4.1. Ele é executado em conjunto com o Eclipse 2018-12 em uma máquina com *Windows* 10 HSL x64, 8GB RAM e *Intel Core* i5-5200U (2,2GHz).

A Tabela 1 contém configurações do QoN++. O parâmetro α pondera a importância da capacidade estimada e da capacidade do *slot* anterior no cálculo atualizado da nova capacidade de um nó (Eq. 2). β é responsável pela contribuição de nós intermediários para o cálculo da probabilidade de contato entre um par de nós, com função semelhante à adotada pelo PRoPHET [12]. ξ reduz exponencialmente a probabilidade de contato entre um par de nós com o passar do tempo [12]. O parâmetro η pondera a importância da capacidade e da probabilidade no cálculo de QoN (Eq.7). P_{init} é o valor inicial fixo da probabilidade (Eq.5). O parâmetro λ corresponde à influência da força de conectividade sobre a probabilidade de contato entre um par de nós (Eq.5). *TS* é o tempo fixo alocado para um *slot* de tempo (Eq.1).

Tabela 1 – Configurações para o QoN++

Parâmetro	Valor
α	0,15
β	0,10
ξ	0,80
η	0,70
P_{Init}	0,70
λ	0,90
TS (min)	60

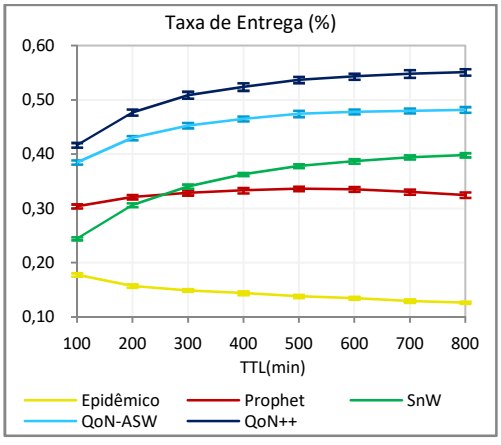
A Tabela 2 contém um valor padrão para cada parâmetro básico das simulações.

Tabela 2 – Configurações para as simulações

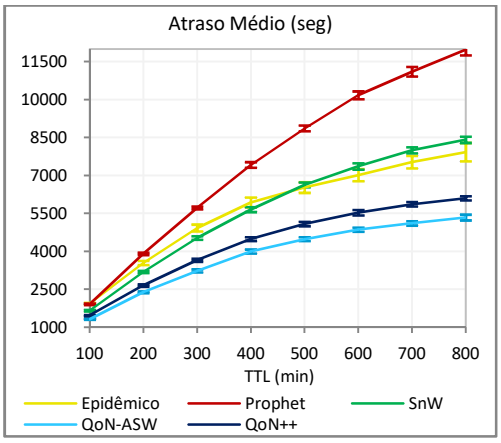
Parâmetro	Valor
Tempo total de simulação (h)	40
Intervalo de geração de mensagens (s)	60
Tamanho do *Buffer* (MB)	05
Velocidade de Transmissão (kBps)	250
Tamanho das mensagens (kB)	[300,500]
Time-to-Live ou TTL (min)	300
Número de Cópias - SW/QoN-ASW/QoN++	06
Número de *Slots* - QoN-ASW/QoN++	40

Três experimentos foram realizados. O primeiro apresentou variações no valor de TTL, com demais parâmetros mantidos. Do mesmo modo, o segundo experimento apresentou variações do tamanho de *buffer* e o terceiro teve tempo de simulação variável. Cada ponto dos gráficos é o resultado da média de 20 execuções com *seeds* randômicos de criação de mensagens e os respectivos intervalos de confiança de 95%.

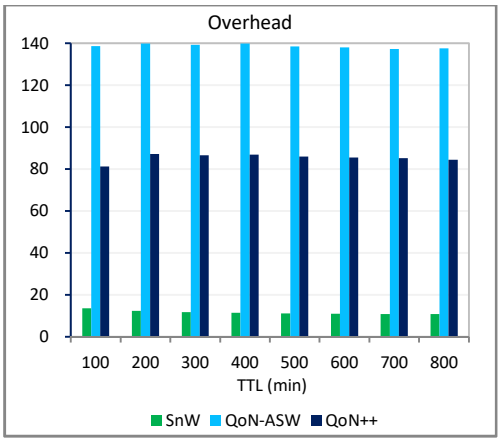
A. Experimento 1 – TTL variável



a)



b)

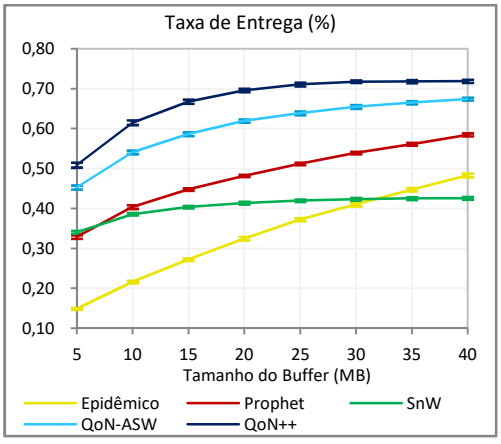


c)

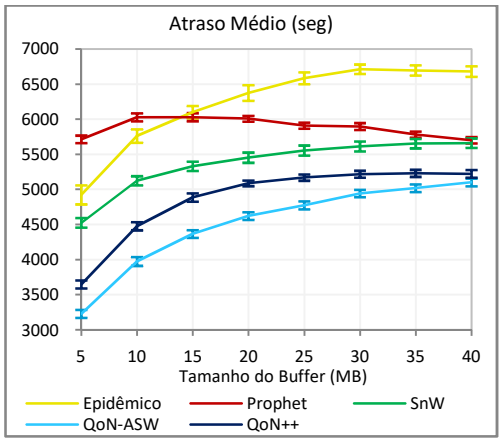
Figura1. Desempenho do roteamento para TTL variável

À medida que o valor de TTL aumenta, as mensagens podem permanecer no *buffer* por mais tempo. Estratégias do tipo *flooding* são essenciais ao Epidêmico, mas, quando implementadas sem controle do número de cópias, podem causar saturação de rede e perda de pacotes, reduzindo a eficiência na entrega. Algoritmos que possuem controle de cópias e as encaminham de acordo com cálculos de probabilidade minimizam congestionamentos, aumentam taxas de entrega (Fig.1a) e reduzem atrasos (Fig.1b). Os valores de sobrecarga do Epidêmico e do PRoPHET foram muito elevados, não sendo plotados. O QoN++ reduz a sobrecarga do QoN-ASW em até 40% (Fig.1c).

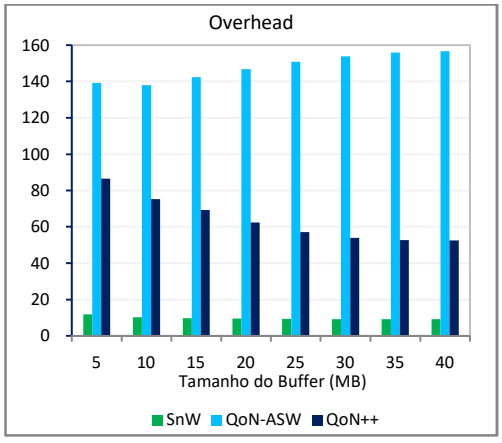
B. Experimento 2 – *Buffer* variável



a)



b)

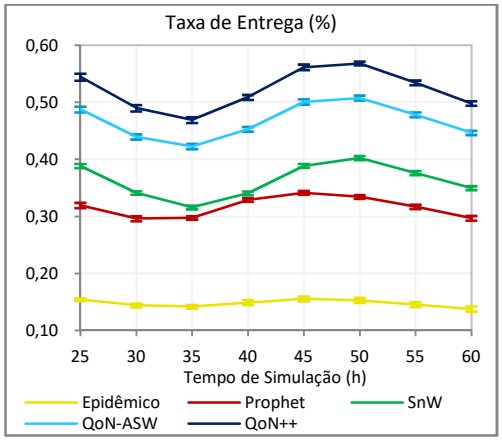


c)

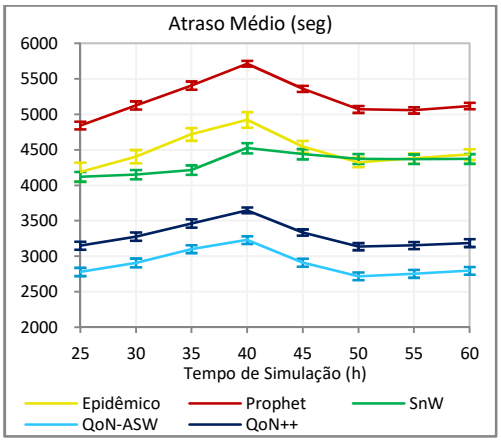
Figura2. Desempenho do roteamento para diferentes tamanhos de *buffer*

O aumento do *buffer* favorece estratégias do tipo *flooding*, melhorando o desempenho do Epidêmico e do PRoPHET na entrega de pacotes. Como o QoN-ASW e o QoN++ conciliam controle de cópias e cálculos de probabilidade, os desempenhos destes continuam sendo os melhores (Fig.2a e 2b). O controle de encaminhamentos faz com que a sobrecarga de QoN++ seja até 65% menor que o de QoN-ASW (Fig.2c).

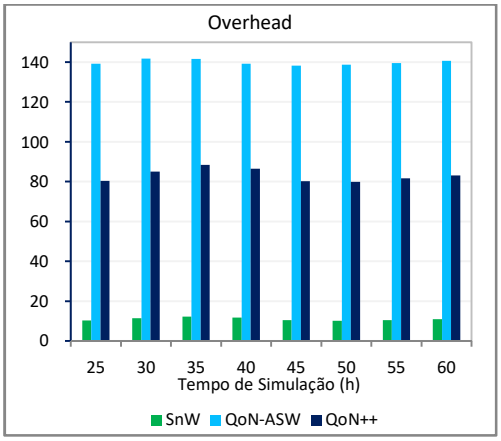
C. Experimento 3 – Tempo de Simulação variável



a)



b)



c)

Figura3. Desempenho do roteamento para diferentes tempos de simulação

O controle de cópias e a realização de cálculos de probabilidade reduzem o congestionamento de rede. Quando os nós com maiores qualidades detêm prioridade no repasse, eles possuem maiores chances de entregar as mensagens (Fig.3a) em menor tempo (Fig.3b), reduzindo a necessidade de novos encaminhamentos (Fig.3c). A sobrecarga de QoN++ pode ser até 42% menor que o de QoN-ASW.

V. CONCLUSÕES

A atualização dos *buffers* realizada pelo QoN++ contribui para um melhor gerenciamento de recursos de memória de dispositivos de uma rede. O ajuste da estratégia de repasses baseado no conceito de QoN faz com que dispositivos com qualidades inferiores não sejam sobrecarregados. Como consequência, há um ganho nas taxas de entrega e uma grande redução da sobrecarga na rede. Em compensação, como o número de encaminhamentos é menor, existe a tendência a atrasos maiores do QoN++ em relação a QoN-ASW. Dessa forma, o QoN++ mostra-se especialmente útil em cenários em que os nós sejam tolerantes a atrasos, porém com mais restrições de recursos de rede, como memória ou energia.

AGRADECIMENTOS

Esta pesquisa foi financiada: i) conforme previsto no Art. 48 do decreto nº 6.008/2006, pela Samsung Eletrônica da Amazônia Ltda, nos termos da Lei Federal nº 8.387/1991, através de convênio nº 004, firmado com o Centro de P&D em Eletrônica e Tecnologia da Informação da Universidade Federal do Amazonas – CETELI / UFAM; ii) pela coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes); iii) pela Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM/ProgramaPPP); iv) fundo setorial de infraestrutura (CT-INFRA); v) MCT/CNPQ.

Emails: eng.wcr@gmail.com, celsocarvalho75@gmail.com

REFERÊNCIAS

[1] C. Carvalho; E. Mota; E. Ferraz; P. Seixas; P. Souza; V. Tavares; W. L. Filho; D. Ferreira; P. Manzoni; C. Calafate. *Entropy based routing for mobile, low power and lossy wireless sensors networks*. International Journal of Distributed Sensor Networks, Vol.15 (7), 2019.

[2] A. Shadid; B. Khalid; S. Shaukat; H. Ali; M. Y. Qadri. *Internet of Things Shaping Smart Cities: A Survey*. Internet of Things and Big Data Analytics Toward Next-Generation Intelligence, Springer International Publishing AG, 2018.

[3] G. J. Joyia; R. M. Liaqat; A. Farooq; S. Rehman. *Internet of Medical Things (IOMT): Applications, Benefits and Future Challenges in Healthcare Domain*, Journal of Communications, Vol. 12, No. 4, 2017.

[4] D. H. Roman e K. D. Conlee. *The Digital Revolution comes to US Healthcare*. Internet of Things, Vol. 5, Ed. Goldman Sachs, 2015.

[5] H. Boyes; B. Hallaq; J. Cunningham; T. Watson. *The Industrial Internet of Things (IIoT): An Analysis Framework*. Computers in Industry, Vol. 101, Ed. Elsevier, 2018.

[6] P. Daugherty.; W. Negm; P. Banerjee; A. Alter. Driving Unconventional Growth through the Industrial Internet of Things. Accenture Technology, 2016.

[7] Q. Zhang. Precision Agriculture Technology for Crop Farming. CRC Press, 2015.

[8] L. Xiao e Z. Wang. Internet of Things: a New Application for Intelligent Traffic Monitoring System. Journal of Networks, Vol. 6, No. 6, 2011.

[9] C. E. Perkins e E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. The Second IEEE Workshop on Mobile Computing Systems and Applications, 1999.

[10] D. B. Johnson e D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. Mobile Computing, Kluwer Academy, 1996.

[11] A. Vahdat e D. Becker. Epidemic routing for partially connected ad hoc networks. Department of Computer Science, Duke University, 2000.

[12] A. Lindgren; A. Doria; O. Schelén. Probabilistic routing in intermittently connected networks. Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc, 2003.

[13] T. Spyropoulos; K. Psounis; C. S. Raghavendra. Spray and Wait: An efficient routing scheme for intermittently connected mobile networks. Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, 2005.

[14] J. Cui; S. Cao; Y. Chang; L. Wu; D. Liu; Y. Yang. An Adaptive Spray and Wait Routing Algorithm Based on Quality of Node in Delay Tolerant Network. IEEE Access, April 2019.

[15] A. Keränen; J. Ott; T. Kärkkäinen. The ONE Simulator for DTN protocol evaluation. In Proc. Int. Conf. Simulation TOOLS Techn., Rome, Italy, 2009.

[16] A. Chaintreau; P. Hui; J. Crowcroft; C. Diot; R. Gass; J. Scott. Impact of human mobility on opportunistic forwarding algorithms. IEEE Trans. Mobile Comput., Vol. 6, No. 6, 2007.