Uma Calibragem do Algoritmo de Karn para Redes de Sensores Sem Fio

Eugênia Giancoli[1,2] and Filippe C. Jabour[1,2] e Aloysio C. P. Pedroza[1]

[1] PEE/COPPE/GTA, UFRJ, Rio de Janeiro, Brasil
[2] CEFET/MG, Leopoldina, Brasil
{eugenia, jabour, aloysio}@gta.ufrj.br

Abstract. This paper presents a calibration for well known Karn's algorithm used by TCP. In this work, the Karn's algorithm is used by a wireless sensor networks transport protocol named CTCP. CTCP aims at providing end-to-end reliability and adapts itself to different applications through a two level mechanism of reliability variation. CTCP achieves these properties using hop-by-hop acknowledgments and a dynamic storage control algorithm that operates at each node along a flow. It was observed that the correct adjustment of the timer control intervals has an important impact on the protocol reliability. This kind of calibration was not proposed yet and can be used as a basis for wireless sensor networks protocols that work with positive acknowledgement.

Keywords: sensor networks, transport protocol, Karn algorithm

1 Introdução

As redes de sensores sem fio (RSSF) fornecem uma solução de sensoriamento amplamente distribuída e econômica para ambientes onde as redes tradicionais não conseguem atuar. Suas principais aplicações estão em monitoramento militar, ambiental, médica, industrial ou infraestruturas domésticas. Estas redes são caracterizadas por atrasos longos e variáveis, freqüentes desconexões, altas taxas de erros e recursos limitados. Cada aplicação possui diferentes características e requisitos de tipos de dados, taxas de transmissão e confiabilidade. A maioria dos protocolos existentes para a camada de transporte das redes de sensores foram adaptados para funcionar com determinados tipos de aplicações, ou assumem que os nós trabalham com determinadas camadas de rede ou enlace. Como resultado, estes protocolos não podem ser aplicados em qualquer tipo de rede de sensores. Portanto, os autores projetaram um protocolo da camada de transporte, chamado CTCP (Collaborative Transporte Control Protocol) [1, 2], que suporta aplicações múltiplas na mesma rede, provendo controle de confiabilidade variável, controle de congestionamento, redução de perdas e suporte a desconexões freqüentes.

Este trabalho explora a calibragem dos intervalos de controle do temporizador do protocolo de transporte CTCP (Collaborative Transporte Control Protocol)

e está organizado da seguinte maneira: na Seção 2 O CTCP é resumidamente descrito, enquanto na Seção 3 nós relatamos as decisões de projeto utilizadas na calibragem dos intervalos de controle. As Conclusões podem ser encontrados em 4.

2 Descrição do CTCP

Os nós sensores, geralmente, são espalhados por uma região de difícil acesso formando uma rede de sensores sem fio. Cada um dos sensores é capaz de coletar dados e roteá-los para o nó sorvedouro que está fisicamente conectado à estação base. Os dados são roteados para o nó sorvedouro através de uma arquitetura de múltiplos saltos. Este trabalho considera que a estação base e os nós utilizam a pilha de protocolos sugerida por [3]. Esta pilha de protocolos consiste das camadas de aplicação, transporte, rede, enlace de dados e física.

O CTCP é um protocolo de transporte colaborativo baseado em mecanismos conhecidos de reconhecimentos (ACK) e temporizadores. Os dois níveis de confiabilidade garantem flexibilidade ao CTCP e possibilitam a sua adaptação a diferentes tipos de aplicação. Este mecanismo objetiva suportar interrupções de conexão sem perda de dados. Mesmo quando um nó recebe os dados a serem transmitidos e falha antes de reencaminhá-los, o protocolo está apto a recuperar esta perda. O CTCP usa reconhecimentos salto-a-salto, considerados eficientes por [4], com liberação imediata dos *buffers*, o que aumenta a capacidade de reencaminhar pacotes e previne o congestionamento. Além disso, o protocolo prevê um mecanismo de controle de congestionamento apto a evitar perdas relativas a *buffer* cheio. O CTCP foi projetado para trabalhar com quaisquer camadas subjacentes.

Cada aplicação possui requisitos diferentes de confiabilidade. Algumas, por exemplo, suportam perdas de dados, enquanto outras precisam garantir que cada um dos seus pacotes chega ao destino. Desta forma, para especificar a confiabilidade requerida, é preciso que se tenha conhecimento da aplicação e de seus objetivos. A alteração do nível de confiabilidade, que pode ser executada a qualquer momento, pode ser exemplificada como a esgotamento de energia dos nós. Neste caso, pode ser mais interessante diminuir a confiabilidade da rede para aumentar sua vida útil. Quando o nível de confiabilidade é alterado, um pacote RSP é enviado ao nó origem com o novo nível de confiabilidade solicitado. Para resolver este problema o CTCP se utiliza de um *Algoritmo Distribuído de Confiabilidade Dinâmica*, que está descrito a seguir:

Nível 1 de Confiabilidade: Este nível de confiabilidade visa economia de energia, através da redução de retransmissões, possui baixo custo de *buffers* e aplica-se principalmente a aplicações que possuem alguma redundância de dados ou que possam tolerar perdas.

Depois de receber um pacote de um nó A, o nó B guarda uma cópia do pacote no seu *buffer*, inicia o temporizador, reencaminha o pacote e envia ao nó A um reconhecimento (ACK), passando a ser temporariamente responsável pela entrega do pacote à estação base. Este processo acontece repetidamente, através

da rota estipulada pela camada de rede, até que a estação base receba o pacote de dados e envie um ACK ao nó imediatamente anterior. Qualquer um dos nós, ao receber um ACK, pode descartar o pacote enviado, poupando espaço em seu *buffer* conhecidamente reduzido. Esta situação está representada graficamente na Figura 1.

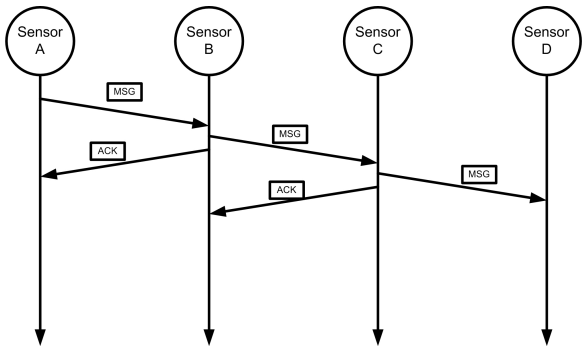

Fig. 1. O protocolo CTCP com nível 1 de confiabilidade.

Nível 2 de confiabilidade: O algoritmo do nível 2 de confiabilidade aumenta a probabilidade de entrega de uma mensagem, uma vez que a falha de um nó no caminho não interrompe a entrega dos dados.

O nó A envia os dados para o nó B e espera receber o *duplo ACK*. O duplo ACK é gerado da seguinte maneira: B recebe os dados de A e devolve para A o primeiro ACK. O nó B envia os dados para C que devolve para B o primeiro ACK. Quando B receber o primeiro ACK de C envia para A o segundo ACK (duplo ACK). Somente após receber o duplo ACK, A descarta os dados mantidos em *buffer*. Todos os nós repetem este processo, sucessivamente, até que os dados cheguem à estação base. A estação base deve enviar somente o duplo ACK ao nó imediatamente anterior. A Figura 2 representa graficamente esta troca de mensagens.

Se o nó B falhar antes de entregar os dados ao nó C, o nó A não recebe o duplo ACK e retransmite o pacote. Pressupõe-se que as falhas dos nós são monitoradas pelo algoritmo de roteamento e este é responsável por refazer a rota quando da falha de um determinado nó, ou conjunto deles.

Como no nível 1, a ausência de recebimento do primeiro ACK gera um esgotamento de temporização do nó origem e a retransmissão do pacote não reconhecido. Além disso, a ausência de recebimento do duplo ACK, também gera um esgotamento de temporização do nó origem e a conseqüente retransmissão do pacote. Repare que são necessários dois temporizadores distintos, uma vez que o tempo de transmissão entre A e B é diferente do tempo entre A e C.

3 Calibragem do Intervalo de Controle do Temporizador

O protocolo CTCP utiliza um mecanismo de controle de temporização (retransmissão) para recuperar segmentos perdidos. Embora conceitualmente simples, surgem algumas questões sutis quando implementamos um mecanismo de controle de temporização em um protocolo para redes de sensores sem fio.

O intervalo de controle do temporizador (intervalo de tempo medido pelo temporizador) é normalmente definido em função do tempo de transmissão do pacote e de seu reconhecimento (ACK). Na Figura 2 estão destacados os dois intervalos de controle que são discutidos nesta seção.

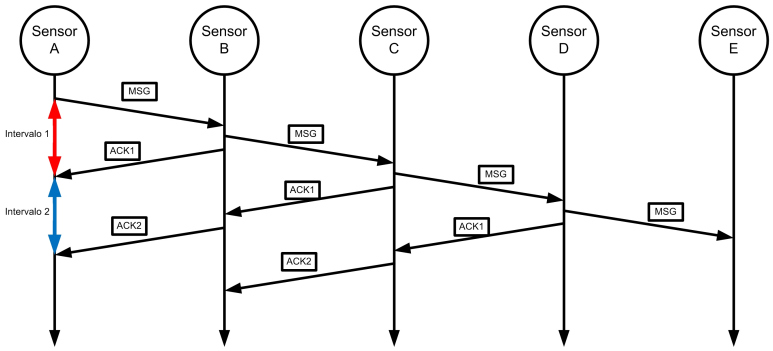

Fig. 2. O protocolo CTCP com nível 2 de confiabilidade. Estão destacados o primeiro e segundo intervalos de controle do temporizador.

O intervalo de controle do temporizador, geralmente, deve ser maior que o tempo de transmissão de ida e volta da conexão fim-a-fim (RTT - *Round Trip Time*). Segundo [5], *Round Trip Time* (RTT) é a medida de retardo da transmissão entre dois *hosts*. O tempo de retardo de transmissão (RTT) consiste no total de tempo que um único pacote ou datagrama leva para deixar um equipamento, alcançar o outro e retornar. Na maioria das redes cabeadas de comutação de pacotes, os retardos variam em função do congestionamento. Assim, a medida do tempo de retorno da transmissão é uma média que pode ter desvio padrão alto.

O protocolo CTCP trabalha com reconhecimentos positivos, o que possibilita a utilização do RTT. Deve-se lembrar que o RTT é utilizado pelo TCP [5] para medir o tempo transcorrido entre o momento em que o segmento é enviado (para a camada de rede da máquina origem) e o momento em que é recebido um reconhecimento para este segmento (enviado pela máquina destino). A máquina origem e a destino, no TCP, podem estar em redes distintas e distantes. Para utilizar o RTT como base para o intervalo de controle do temporizador do CTCP é necessário considerar cada um dos saltos, ou seja, nesta proposta, o RTT mede

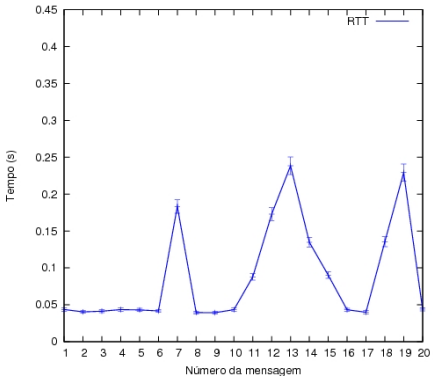

Fig. 3. Tempo de retorno (RTT) de 25 pacotes consecutivos em uma rede de sensores.

o tempo entre a ida de uma mensagem e a volta de um ACK em cada um dos saltos da conexão.

Para reunir dados necessários para um aperfeiçoamento do temporizador, o CTCP registra o instante em que cada segmento é enviado e o instante em que chega uma confirmação para este segmento. A partir dos instantes registrados, o CTCP calcula um período de tempo, chamado amostra do RTT. A Figura 3 ilustra o tempo de retorno de 25 pacotes consecutivos. Foi utilizado o nível 1 de confiabilidade do protocolo CTCP.

Na Figura 3 pode-se notar que, ao contrário do que acontece nas conexões fim-a-fim, o RTT não possui uma variação absoluta significativa. Os picos registrados no gráfico denotam momentos de retransmissão, onde o tempo medido foi aquele que transcorreu entre a primeira vez que o segmento foi enviado e a chegada do primeiro ACK do referido segmento. A partir destes resultados preliminares, optou-se por basear o mecanismo de temporização adaptável do CTCP no algoritmo de Karn [5], [6]. Este algoritmo, muito conhecido por sua implementação junto ao protocolo TCP, tem sido exaustivamente estudado e calibrado para redes cabeadas e, segundo Comer [5], a experiência prova que o algoritmo de Karn funciona bem, mesmo em redes com alto nível de perdas de pacotes. Este trabalho, pretende estipular os parâmetros α, β e γ (descritos a seguir), para o seu funcionamento nas redes de sensores sem fio.

O CTCP mantém uma média, denominada *RTTPrev*, dos valores de RTT calculados para cada segmento transmitido. Ao obter uma nova medida do RTT, o CTCP atualiza o *RTTPrev* de acordo com a seguinte fórmula:

$$RTTPrev = (1 - \alpha) \times RTTPrev + (\alpha \times Amostra) \quad (1)$$

Note que *RTTPrev* é uma média ponderada dos valores de RTT, onde o peso atribuído a uma amostra varia de acordo com o valor estipulado para α. α é uma constante de peso, $0 \leq \alpha \leq 1$, utilizada para avaliar a média antiga em relação à última amostra do RTT. Escolher um valor de α próximo a zero

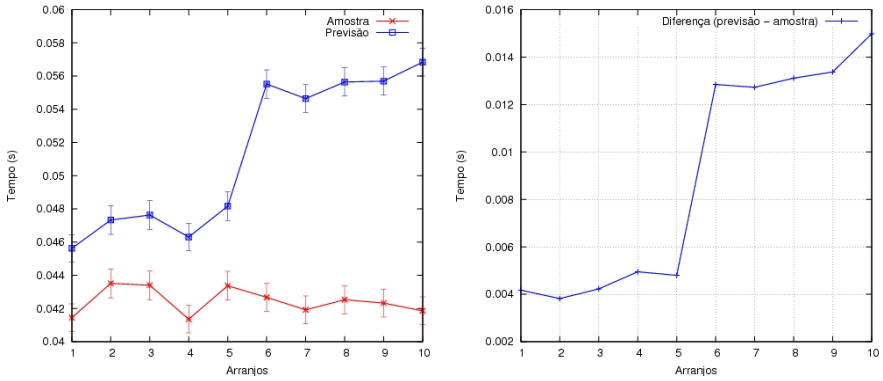

Fig. 4. Valores de P e A (esquerda) e suas diferenças (direita) em cada um dos arranjos.

torna a média ponderada imune às alterações de curta duração (por exemplo, um segmento único que encontra um intervalo longo). Escolher um valor próximo a um faz com que a média ponderada reaja muito rapidamente às alterações dos intervalos. Uma vez determinado o valor adequado de α, o CTCP, ao enviar um pacote, calcula um valor para o intervalo de controle ($Timeout$) como uma função do $RTTPrev$.

$$Timeout = \beta \times RTTPrev \tag{2}$$

onde β é um fator constante de peso, $\beta > 1$, que torna o $Timeout$ maior que a estimativa atual do tempo de ida e volta. Com o objetivo de detectar um pacote perdido rapidamente, o valor do $Timeout$ deve ser próximo ao RTTPrev e β deve possuir valor próximo de um. A rapidez na detecção de um pacote perdido aumenta a vazão da rede, pois o CTCP não vai esperar um tempo longo desnecessário para retransmitir o pacote. Por outro lado, se $\beta = 1$, um pequeno atraso do ACK provoca uma retransmissão desnecessária que consome energia da rede. A especificação deste parâmetro deve considerar um compromisso entre vazão e consumo de energia.

Os valores de α e β foram determinados para redes cabeadas que utilizam o protocolo TCP. Em redes de sensores, estes estudos não foram feitos, até então. Um dos objetivos desta seção é determinar valores adequados para estes parâmetros, considerando o protocolo proposto (CTCP) e a natureza restritiva das redes de sensores.

Na versão inicial da implementação do protocolo CTCP, descrita no trabalho [1], considerou-se um RTT fixo e árbitrário. Contudo, para determinar os valores de α e β, novas simulações foram realizadas com α variando entre $0,1$ e $0,9$. Para β, dois valores foram considerados $1,1$ e $1,3$.

Durante a execução das simulações foram criados arquivos de *log* que registravam o tempo real de ida e volta de uma determinada mensagem. A cada um destes tempos foi dado o nome de amostra. A partir desta amostra, o CTCP

calcula, baseado nas equações 1 e 2, o valor a ser atribuído para o próximo intervalo de tempo (previsão). Desta maneira, para cada arranjo de α e β, foram registradas as amostras e previsões do intervalo de tempo para cada uma das mensagens. Para cada arranjo de α e β, calculou-se a média aritmética das amostras e também das previsões. O resultado está na Figura 4.

O arranjo ideal é aquele onde a diferença entre a previsão (P) e a amostra (A) é a menor possível, guardada uma distância mínima ϵ. Logo, a acuidade do algoritmo proposto é inversamente proporcional à diferença entre previsão e amostra.

Seja $\Delta = P - A$

Deseja-se: $P > A$ e $\Delta \rightarrow (0 + \epsilon)$

Tem-se então as seguintes situações:

- $\Delta < 0$: podem ocorrer retransmissões desnecessárias. O temporizador expirou em P, a mensagem foi retransmitida enquanto a confirmação ainda estava a caminho;
- $\Delta \rightarrow 0$: uma previsão muito próxima das amostras fica susceptível a erros. Daí o uso dos fatores α e β descritos anteriormente;
- $\Delta >>> 0$: dificilmente ocorrem retransmissões desnecessárias. Entretanto, pode haver um aumento da latência, uma vez que há um maior atraso nas retransmissões necessárias.

Na Figura 4, lado esquerdo, pode-se notar que o gráfico é claramente dividido em dois blocos. O primeiro bloco refere-se aos arranjos de 1 a 5 onde $\beta = 1,1$ e o segundo refere-se aos arranjos onde $\beta = 1,3$. Desta maneira, pode-se concluir que o acerto é maior na primeira metade do gráfico, onde $\beta = 1,1$.

Para determinar o valor ideal de α plotamos o gráfico da Figura 4. Observa-se que o segundo arranjo, onde $\alpha = 0,7$, possui o menor erro entre previsão e amostra.

O CTCP, assim como o TCP, trabalha com um esquema de confirmação cumulativo, no qual um reconhecimento refere-se a um segmento com número de seqüência único, ou seja, se um segmento com número de seqüência x é enviado, o reconhecimento vem com número de seqüência x. Contudo, se o segmento x for retransmitido várias vezes, quando o reconhecimento chegar, não se pode determinar qual dos segmentos enviados gerou o reconhecimento. Por exemplo, o CTCP forma um segmento e o envia. O temporizador se esgota e o CTCP envia o segmento novamente, em um segundo pacote. Já que os dois pacotes transportam exatamente os mesmos dados e possuem o mesmo número de seqüência, o transmissor não tem como saber se uma confirmação corresponde ao pacote original ou ao retransmitido. Esse fenômeno tem sido denominado ambigüidade de confirmação, e as confirmações do TCP são conhecidas como ambíguas.

Assim, se uma transmissão original e a transmissão mais recente deixam de fornecer tempos de ida e volta precisos, o CTCP não deve atualizar a estimativa do tempo de ida e volta para segmentos retransmitidos. Essa idéia é um dos fundamentos do Algoritmo de Karn [5], que evita o problema de confirmações inteiramente ambíguas simplesmente estimando o tempo de ida e volta e ignorando as amostras que correspondam a segmentos retransmitidos. Contudo,

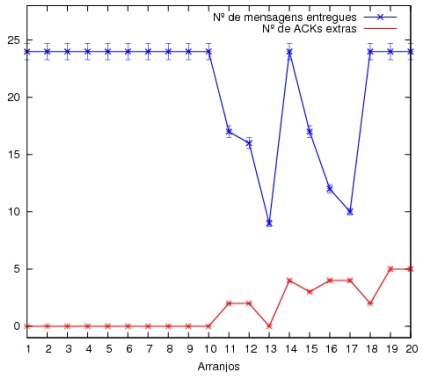

Fig. 5. Número de mensagens entregues e ACKs extras.

Karn ainda sugere a utilização de uma técnica de *backoff* do temporizador, onde o valor do intervalo de controle após um pacote ser retransmitido é aumentado. As implementações usam diferentes técnicas para calcular o *backoff*. A maioria escolhe um fator multiplicativo γ e fixa o novo valor em:

$$Timeout\_1 = \gamma \times Timeout \tag{3}$$

O algoritmo de Karn determina que as amostras do RTT não devem considerar os segmentos retransmitidos. Assim, uma nova etapa de simulações se inicia, com o objetivo de determinar os valores apropriados de γ. A Tabela 1 lista as combinações de parâmetros utilizados. Os resultados obtidos estão na Figura 5.

Arranjo	α	β	γ	Arranjo	α	β	γ
1	0,9	1,1	1,5	11	0,9	1,1	1
2	0,7	1,1	1,5	12	0,7	1,1	1
3	0,5	1,1	1,5	13	0,5	1,1	1
4	0,3	1,1	1,5	14	0,3	1,1	1
5	0,1	1,1	1,5	15	0,1	1,1	1
6	0,9	1,3	1,5	16	0,9	1,3	1
7	0,7	1,3	1,5	17	0,7	1,3	1
8	0,5	1,3	1,5	18	0,5	1,3	1
9	0,3	1,3	1,5	19	0,3	1,3	1
10	0,1	1,3	1,5	20	0,1	1,3	1

Table 1. Valores de α, β e γ utilizados nas simulações.

O valor de γ não influencia no cálculo do RTTPrev, mas sim no *backoff* do temporizador. Contudo, como foi dito anteriormente, só é necessário considerar um *backoff* para o temporizador durante a retransmissão de uma mensagem.

Assim, para avaliar a eficiência do fator multiplicativo γ, é preciso avaliar o número de ACKs extras para cada mensagem. Entende-se por ACK extra qualquer ACK duplicado, que chegue ao nó origem. Por exemplo, a mensagem com número de seqüência 33456 foi enviada do nó origem A para o nó destino B. O nó B deve responder ao nó A com somente um ACK para a mensagem 33456. Entretanto, se o *backoff* do temporizador do nó A estiver mal ajustado, o nó A retransmite para o nó B antes que este possa lhe responder. Em conseqüência, o nó B envia um ACK para cada uma das cópias recebidas e gera os ACKs extras no nó A. Concluindo, um ACK extra denota uma mensagem retransmitida sem necessidade em função de um intervalo de controle mal calculado.

O gráfico da Figura 5 está dividido em duas partes. Nos arranjos de 1 a 10, o valor de γ é $1,5$ e nos arranjos de 11 a 20 passou a ter valor 1, ou seja, deixou de influenciar o valor do *Timeout*, uma vez que γ é um fator multiplicativo (veja Equação 3).

É importante ressaltar que a ausência do γ, ou seja, $\gamma = 1$, gera grande instabilidade e aumento no número de mensagens retransmitidas desnecessariamente. Durante a análise do número de ACKs extras foi possível perceber que o valor de γ também afeta o número de mensagens entregues ao nó destino. O número de mensagens entregues só volta a crescer para os arranjos 18, 19 e 20 porque o maior valor de β ajudou a compensar a ausência de γ.

Concluindo, após as simulações e medições, assume-se que $\alpha = 0,7$, $\beta = 1,1$ e $\gamma = 1,5$ são os valores adequados para o cálculo do intervalo de controle do protocolo CTCP, em redes de sensores sem fio.

Para as simulações que se referem ao nível 2 de confiabilidade é necessário criar um segundo intervalo de controle do temporizador. Este segundo intervalo refere-se ao tempo necessário para esperar o recebimento do ACK duplo (ACK2).

O segundo intervalo de controle utiliza as mesmas equações (1), (2), (3) do primeiro intervalo. Portanto, é preciso, mais uma vez, analisar o valores ideais para α_2, β_2 e γ_2. Realizou-se mais uma bateria de simulações, variando os três parâmetros de acordo com a Tabela 1.

Observa-se pela Figura 6 que temos vários arranjos com valores reduzidos de Δ. Diferente do que ocorreu no cálculo do $Timeout\_1$, temos uma maior variação nos ACKS repetidos. Isto significa que, em muitas situações, ocorrem retransmissões desnecessárias. Assim, dentre o conjunto dos melhores arranjos segundo o valor de Δ (Figura 6), identifica-se o arranjo 16 como o mais adequado.

Os parâmetros ideais para o segundo intervalo de controle ($Timeout\_2$), referentes ao arranjo 16, são: $\alpha_2 = 0,9$, $\beta_2 = 1,3$ e $\gamma_2 = 1,0$.

4 Conclusões

Neste trabalho nós apresentamos uma calibragem do algoritmo de Karn para o protocolo CTCP utilizado em Redes de sensores sem fio (CTCP). Este protocolo se propõe a prover entrega confiável entre o sensor origem e a estação base nas redes de sensores.

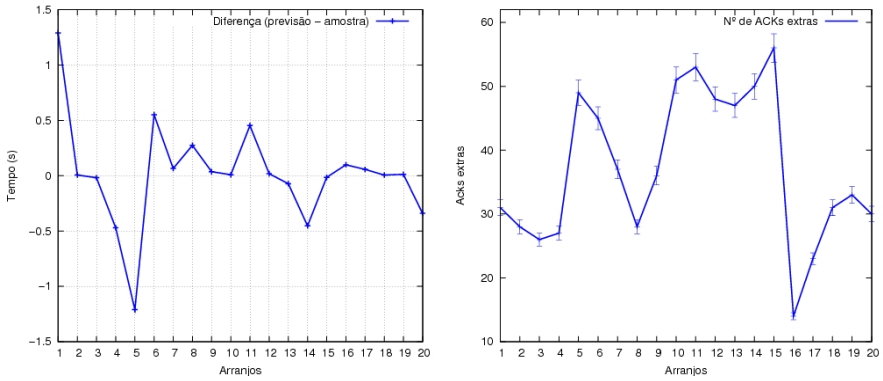

Fig. 6. Diferença entre P e A (esquerda) e Número de ACKs extras (direita) em cada um dos arranjos para $Timeout\_2$.

Para chegarmos aos valores corretos dos dois intervalos de tempo consideramos principalmente o número de ACKs extras e o número de mensagens entregues. Os resultados encontrados durante a calibragem do temporizador demonstram que o sucesso do CTCP está intimamente ligado à correta escolha de seu intervalo de controle.

Referências

1. Giancoli, E., Jabour, F., Pedroza, A.: CTCP: Reliable Transport Control Protocol for Sensor Networks. In: Fourth International Conference on Intelligent Sensors, Sensor Nertworks and Information Processing, Sydney, Australia (2008) 493–498
2. Giancoli, E., Jabour, F., Pedroza, A.: Collaborative Transport Control Protocol. In: Proceedings of IEEE International Conference on Computer and Electrical Engineering, Phuket, Tailândia (2008) 373–377
3. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: A survey. Computer Networks (2002) 393–422
4. Stann, F., Heidemann, J.: RMST: Reliable Data Transport in Sensor Networks. In: Proceedings of the First International Workshop on Sensor Net Protocols and Applications, Anchorage, Alaska, USA (2003) 102–112
5. Comer, D.E.: Interligação em Rede com TCP/IP. 3 edn. Volume 1. Editora Campus (1998)
6. Kurose, J.F., Ross, K.W.: Redes de Computadores e a Internet: Uma abordagem top-down. Trad. 3 ed. edn. Addison Wesley, São Paulo, Brasil (2006)