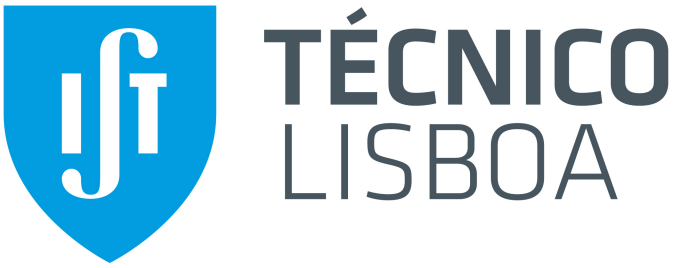

Instituto Superior Técnico

Mestrado Integrado Em Engenharia Eletrotécnica e de Computadores

Redes Móveis e Sem Fios

Lab 1

Trabalho realizado por:	*Número:*
Diogo Moura	86976
Diogo Alves	86980
Luís Crespo	87057

Turno L03 - Grupo 1 - 5a feira 14h:00m-15h30m

2019/2020

Conteúdo

1 Introdução

Neste trabalho estudamos, na 1ª secção, a influência da distância de comunicação, tamanho dos pacotes, ruído na comunicação e dum protocolo CSMA/CA (RTS/CTS), abstraindo-nos da comunicação com vários terminais de modo a estudar esta influência destes parâmetros de forma isolada na comunicação, nomeadamente no *througput*, *MAC retries* e *delay*.

Na 2ª secção do trabalho, estudamos a comunicação à luz desses mesmos parâmetros, adicionando-se uma nova variável que é a existência de vários terminais. Assim, estuda-se uma possível solução para resolver este problema, o protocolo de CSMA/CA , o RTS/CTS, que diminui a probabilidade de colisão entre pacotes.

Na 3ª secção estuda-se a utilização de um *access point*, uma infraestrutura de comunicação utilizada para estender o alcance da comunicação.

O estudo da influência destes parâmetros e soluções é importante para o dimensionamento de uma rede de comunicação sem-fios num cenário real com um ou vários terminais.

2 Work Description

2.1 Impact of Distance in Ad Hoc Point-to-Point Communications

No simulador definiu-se o cenário descrito no enunciado para comunicação de duas entidades, o cliente e o servidor.

2.1.1 Maximum communication range

O *Two-Ray Model* que diz que a potência do sinal recebido é dada por

$$P_r = \frac{P_t.G_t.G_r.\lambda^2}{(4.\pi.d)^2} \tag{1}$$

se a distância for inferior à distância de *crossover* ($d < d_c$), ou seja, caso o sinal se propague em *free space* e segundo

$$P_r = \frac{P_t.G_t.G_r.(h_t.h_r)^2}{d^2} \tag{2}$$

se a distância for superior à distância de *crossover* ($d > d_c$), ou seja, caso exista apenas uma única reflexão no solo. A distância de *crossover* é dada por

$$d_c = \frac{4.\pi.h_t.h_r}{\lambda} \tag{3}$$

e substituindo os valores correspondentes ao cenário em causa obtém-se $d_c = 100.53\,\text{m}$.

A distância máxima de comunicação corresponde a uma potência de recepção igual à sensibilidade do receptor (-85 dBm). Assim, através da equação (1), deduz-se

$$d_{max} = \sqrt{\frac{P_t.G_t.G_r.\lambda^2}{P_r.(4.\pi)^2}} \tag{4}$$

e obtém-se $d_{max} = 2122.66\,\text{m}$. Como a esta distância é superior à distância de crossover e a equação utilizada é para propagação em *free space*, esta expressão não é valida nesta situação. Utiliza-se portanto a equação (2),

$$d_{max} = \sqrt[4]{\frac{P_t.G_t.G_r.(h_t.h_r)^2}{P_r}} \tag{5}$$

e obtém-se $d_{max} = 274.674\,\text{m}$ que corresponde à máxima distância de comunicação calculada teoricamente.

Através do simulador, variando a distância entre o cliente e o servidor verificou-se que a distância a partir da qual não eram recebidos pacotes no servidor corresponde à calculada teoricamente.

2.1.2 Impact of distance and packet size.

Application throughput, message latency (s), percentage of packets received with errors

De forma a saturar a rede, utilizou-se um tamanho de pacote de 25 octetos e uma distância entre cliente e servidor de 1 m e diminuiu-se sucessivamente o intervalo entre pacotes até se verificar uma saturação no número de pacotes recebidos no servidor. Desta forma, obteve-se um intervalo entre pacotes de 0.05 ms. Com este intervalo, simulou-se durante 20 s o cenário para 25, 600 e 1100 octetos e para distâncias entre 1 m e 275.67 m. Dos resultados guardou-se o valor dos pacotes recebidos no servidor para calcular o *throughput*, o *end to end delay* para calcular a latência da mensagem e os pacotes recebidos com erros de forma a calcular a sua percentagem. Para além disto, calculou-se também o número médio de *retries* na camada MAC do cliente através da fórmula

$$averageMACretries = \frac{sentToLower : count(client) - sentToUpper : count(srv)}{sentToLower : count(client)} \tag{6}$$

A partir dos dados retirados da simulação construiu-se o gráfico da Figura 1.

Analisando o gráfico da Figura 1, verifica-se que quanto maior o número de octetos, maior é o *throughput*. Isto acontece porque cada pacote tem um tamanho de *overhead* fixo. Assim, a percentagem de *overhead* em pacotes pequenos é superior à percentagem de *overhead* em pacotes maiores, por este motivo, o *throughput* ao nível da aplicação é superior em pacotes maiores. Verifica-se também que não há muita variação do *throughput* com a distância, com excepção nos últimos valores (distância superior ao alcance

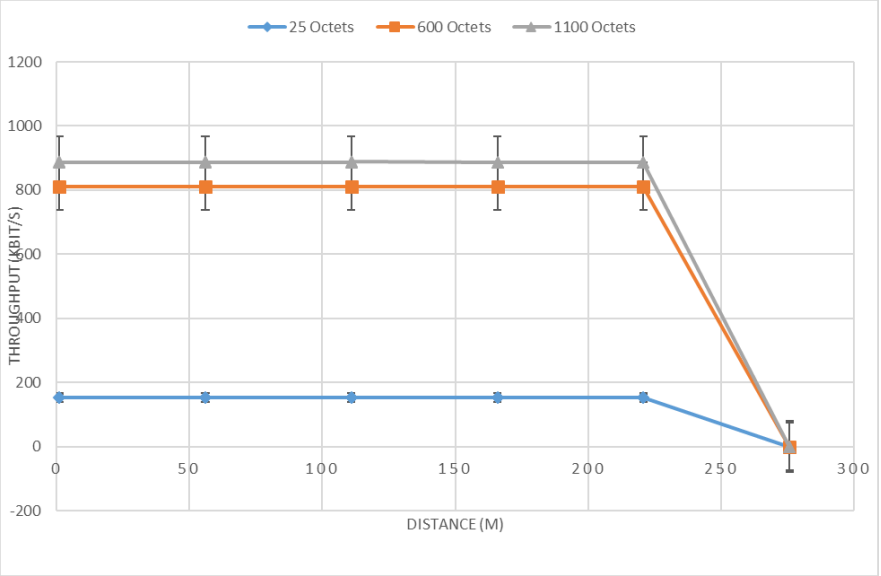

Figura 1: *Throughput como função da distância entre os nós.*

máximo), em que o *throughput* passa repentinamente para 0. Isto acontece porque se trata de um simulador, ou seja, não corresponde exatamente à realidade, de facto, no código do simulador, encontra-se um *if* o que explica este comportamento. Numa situação real, seria de esperar um decréscimo do *throughput* à medida que a distância aumenta.

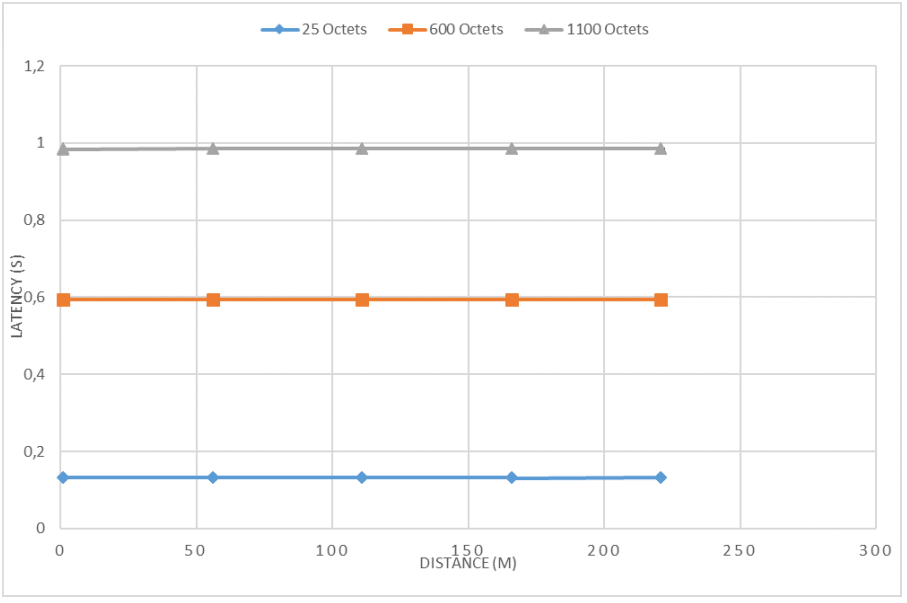

Figura 2: *Latency como função da distância entre os nós.*

Quanto à latência, verifica-se pelo gráfico da figura 2 que esta é tanto maior quanto maior é o tamanho do pacote e varia muito pouco com a distância, uma vez que neste caso o tempo de propagação (à velocidade da

luz) é desprezável face ao tempo de transmissão (à velocidade de 1Mbps), somado ao tempo de processamento do pacote nos vários níveis.

Em relação ao número médio de *retries* MAC (figura 3) , para este caso, como existe só um nó a transmitir e não existe outro tipo de interferência, este número é praticamente sempre 0.

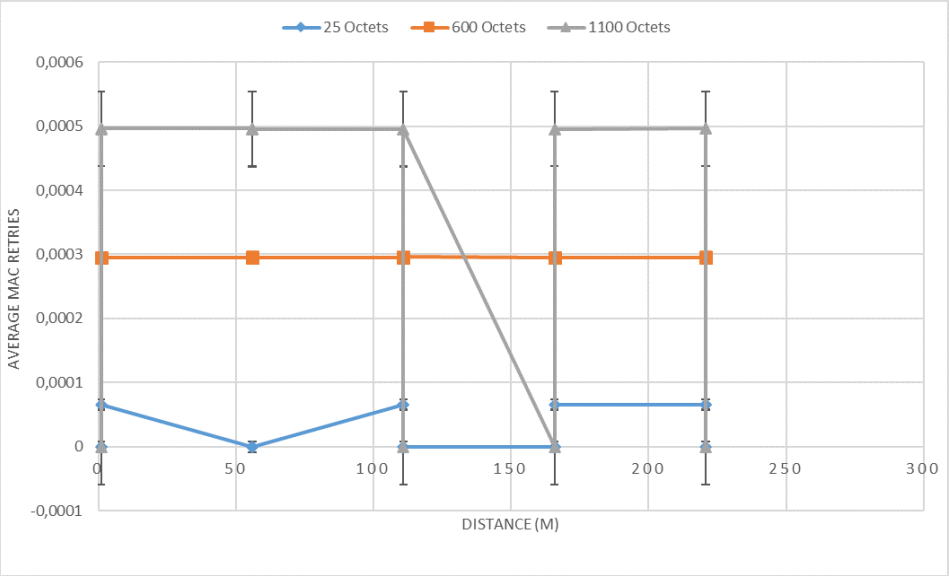

Figura 3: *Nº médio de retries MAC como função da distância entre os nós.*

2.1.3 Impact of distance and packet size with rtsThreshold = 500B. Application throughput, message latency (s), percentage of packets received with errors

Repetiu-se o cenário da alínea anterior mas com o campo *rtsThreshold* de 500 B em vez de 3000 B. O campo *rtsThreshold* define qual é o tamanho mínimo do pacote a partir do qual será estabelecido um *handshake* RTS/CTS antes do envio do pacote. Na situação da alínea anterior nunca foi estabelecido nenhum *handshake* RTS/CTS.

A partir dos pacotes recebidos no servidor construiu-se o gráfico da Figura 4.

Comparando os valores obtidos nas duas situações, verifica-se que para um tamanho de pacote igual a 25 B os valores de *throughput* são exatamente iguais, como seria de esperar, uma vez que não foi utilizado *handshake* RTS/CTS (25 B $<$ *rtsThreshold*). Relativamente aos pacotes de 600 B e 1100 B verifica-se uma ligeira diminuição do *throughput* face à situação anterior. Isto acontece porque nesta situação é utilizado o *handshake* RTS/CTS (600 B, 1100 B $>$ *rtsThreshold*) e portanto, antes de cada pacote, o cliente envia um *frame* RTS (Request to send) ao servidor e espera pela resposta (frame CTS (clear to send)). Naturalmemte, esta troca de *frames* faz com que o tempo de cada pacote aumente, o que reduz o *throughput*.

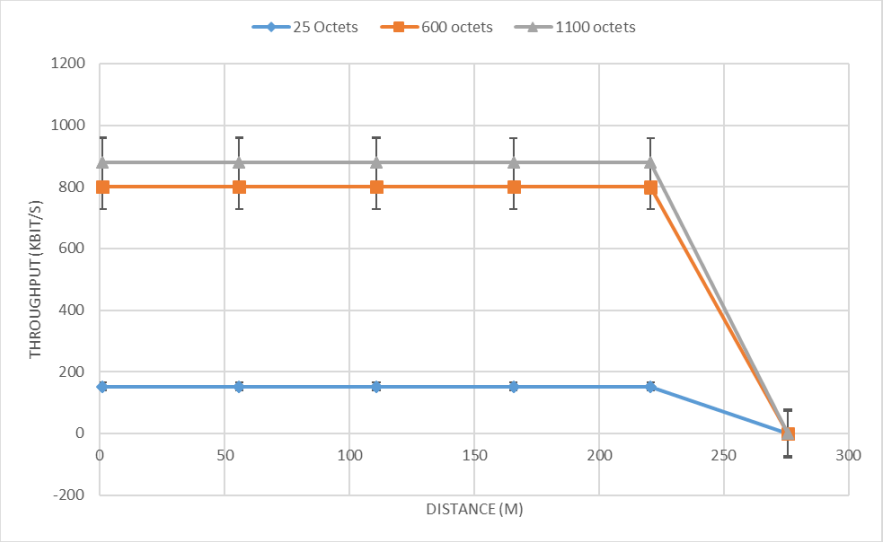

Figura 4: *Throughput como função da distância entre os nós.*

O *handshake* RTS/CTS é utilizado para reservar o canal e impedir que outros *hosts* transmitam durante um período de tempo, mas como neste caso não existem transmissões simultâneas, a utilização deste protocolo tem um efeito negativo no *throughput*.

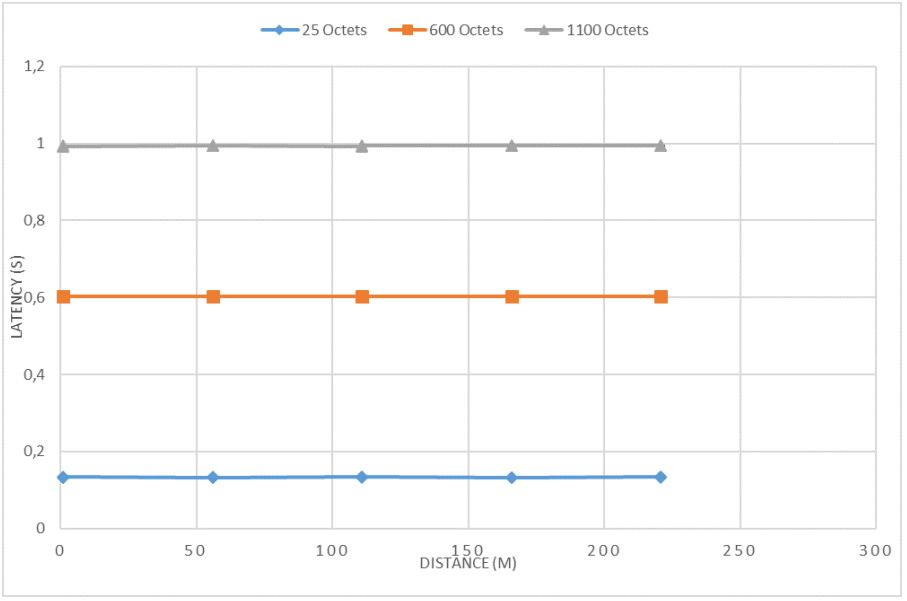

Figura 5: *Latency como função da distância entre os nós.*

Relativamente à *latency* (figura 5) , esta tem um comportamento semelhante à situação anterior, assumindo apenas valores ligeiramente superiores quando é utilizado o *handshake* RTS/CTS.

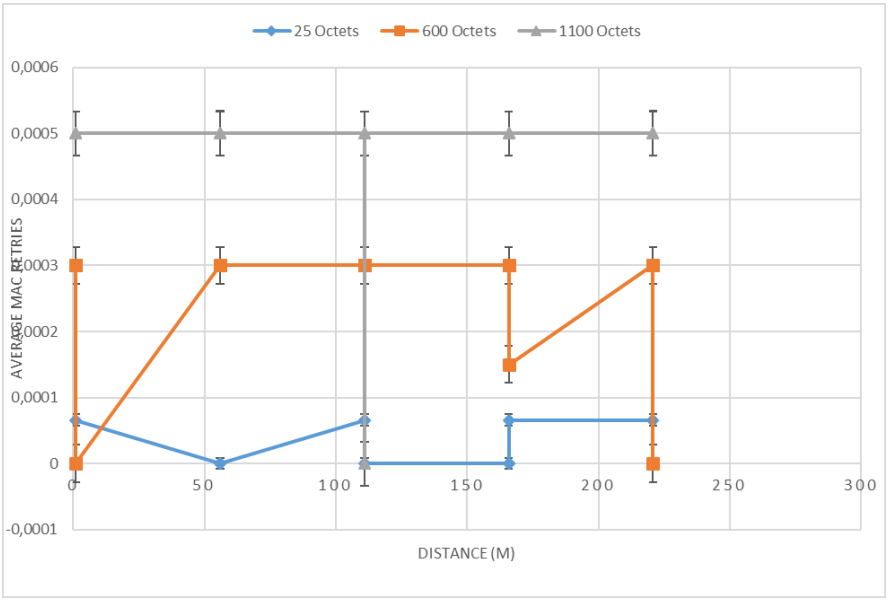

Figura 6: *Nº médio de retries MAC como função da distância entre os nós.*

Já quanto ao número médio de *retries* MAC (figura 6), para calcular este valor nas situações em que é usado RTS/CTS (600B, 1100B), temos que usar a fórmula

$$averageMACretries = \frac{(sentToLower : count(client)/2 - sentToUpper : count(srv))}{sentToLower : count(client)} \tag{7}$$

de modo a ter em conta os *frames* RTS enviados pelo cliente e recebidos na camada MAC do servidor e que não são enviados para a camada acima. Os resultados obtidos são, mais uma vez, muito semelhantes aos obtidos na questão anterior (praticamente 0).

2.1.4 Communication range with backgroundNoise.power=-84dBm

Alterou-se o valor do ruído de fundo para -84 dBm e simulou-se o cenário para pacotes de 25 B e várias distâncias. Verificou-se que o número de pacotes recebidos no servidor é 0 para todas as distâncias (incluindo d=0 m). Após uma análise mais detalhada, verificou-se que os pacotes que chegam à camada MAC do cliente não estão a ser enviados para a camada fisíca. De facto, como backgroundNoise.power=-84dBm > receiver.energyDetection = -85dBm, o cliente está constantemente a detetar outras transmissões e portanto a camada MAC nunca é capaz de enviar um pacote.

2.2 Performance of IEEE 802.11 Ad Hoc under Contention

Definiu-se o cenário no simulador que consiste em clientes distribuídos uniformemente segundo um círculo de raio a definir com centro no servidor com parâmetros de forma a saturar a rede.

2.2.1 Average total throughput

Nesta situação, utilizou-se um raio de 100 m sendo a distância máxima entre *hosts* de 200 m, que é inferior ao alcance máximo. Efectuou-se a simulação e os resultados estão no gráfico da Figura 7.

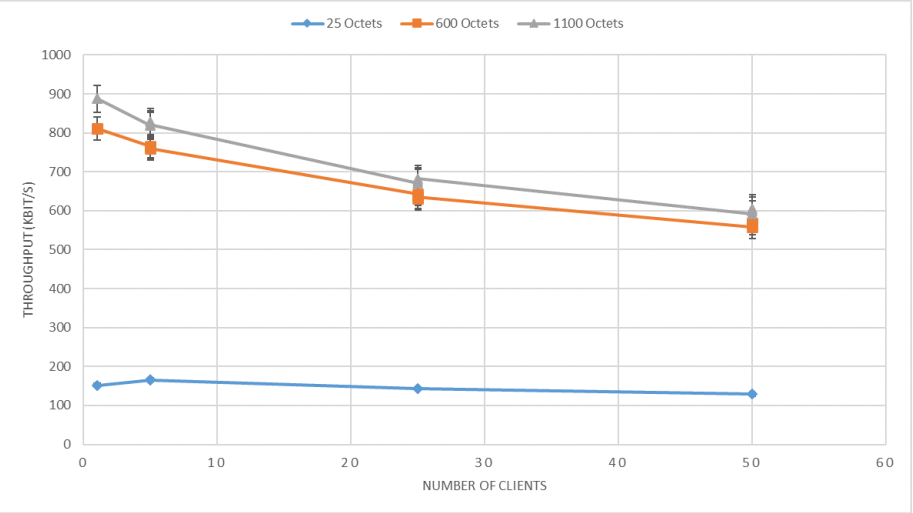

Figura 7: *Throughput como função do número de clientes.*

Como já foi referido, o *throughput* aumenta quando o tamanho dos pacotes aumenta devido ao *overhead*. Quanto à variação do *throughput* com o número de clientes, verificou-se que para pacotes de 25 octetos, o *throughput* mantém-se praticamente constante à medida que o número de clientes aumenta, mas para 600 e 1100 octetos, o *throughput* diminui à medida que o número de clientes aumenta. Isto pode ser explicado pelo facto de que cada *host* tem que "ouvir"o canal antes de começar a transmitir e se detectar alguma transmissão, tem que esperar um intervalo de tempo aleatório (*backoff*) aumentado exponencialmente cada vez que isto acontece. Deste modo, quantos mais *hosts*, maior a probabilidade de ser detetada uma colisão e maior se torna o intervalo de *backoff*, resultando num *throughput* mais baixo. Para um tamanho de pacote igual a 25 octetos isto não se verifica porque o tamanho de cada *frame* não é suficiente para que a probabilidade de colisão aumente significativamente.

2.2.2 Average total throughput outside of interference range

Para esta situação, o raio do círculo foi aumentado para 140m de forma a que dois hosts em lados opostos não estejam ao alcance um do outro. Observando o gráfico (Figura 8), verifica-se que o *throughput* decresce muito mais rapidamente com o aumento do número de clientes do que na situação anterior: para 600 e 1100 octetos, a partir dos 25 clientes o *throughput* já é praticamente zero. Isto acontece porque, como os hosts de lados opostos do círculo não estão ao alcance um do outro, eles não conseguem saber quando é

que o host do lado oposto está a transmitir e acabam por transmitir os dois simultaneamente, pelo que o servidor não consegue receber nenhum corretamente.

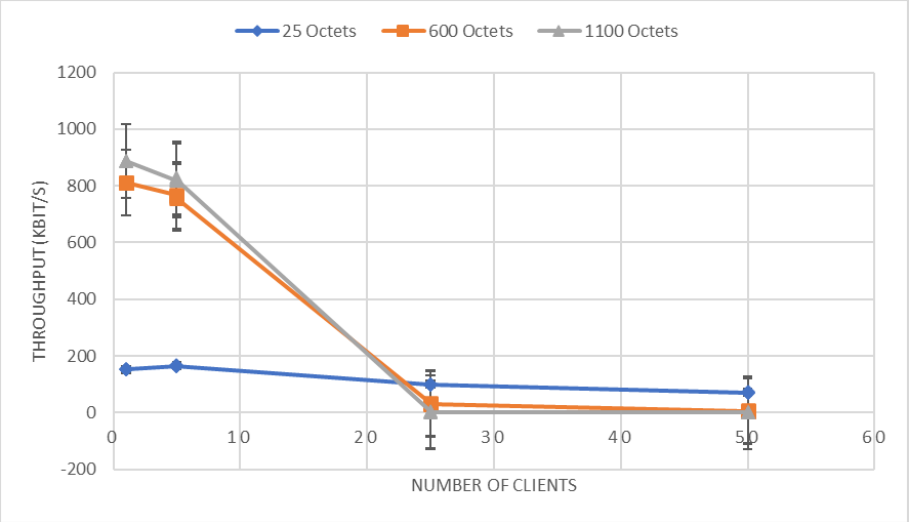

Figura 8: *Throughput como função do número de clientes.*

2.2.3 Average total throughput outside of interference range and with *rtsThreshold*=500

Nesta situação, alterou-se o parâmetro *rtsThreshold* para 500 B o que na prática significa que o protocolo RTS/CTS irá ser utilizado para o tamanho do pacote igual a 600B e 1100B. Através o gráfico (Figura 9), verifica-se que para 25 octetos, como seria de esperar, os valores de *throughput* são iguais à situação anterior. Já para 600 e 1100 octetos, verificamos que os valores de *throughput* são praticamente constantes à medida que se aumenta o número de clientes, Como neste caso é utilizado o *handshake* RTS/CTS, cada host precisa de enviar um frame RTS ao servidor e esperar pela resposta (frame CTS) antes de poder enviar o pacote de dados. Desta forma, é impossível dois hosts transmitirem pacotes de dados em simultâneo, mesmo que não estejam ao alcance um do outro, o que faz com que o *throughput* se mantenha praticamente constante.

Figura 9: *Throughput como função do número de clientes.*

2.2.4 Based on the simulations performed in Q.2.2.1, calculate the average total throughput measured at the MAC layer and compare with the application layer throughput. Explain the found differences

Voltando à simulação da questão 2.2.1 e retirando os pacotes recebidos ao nível da camada MAC, é possível calcular o *throughput* nesta camada.

Observando o gráfico obtido (figura 10) e comparando com o obtido na questão 2.2.1 (figura 7), verifica-se que os valores obtidos para o *throughput* são sempre superiores na nova situação, mas a diferença é mais acentuada quando o tamanho do pacote é de 25B. Isto acontece porque a camada MAC tem um header de tamanho fixo (31B neste caso) e portanto quanto menor for o *payload* da aplicação, maior será o *throughput* utilizado para transmitir este *header*.

Observámos também que, ao contrário do que acontece na situação anterior, nesta situação o *throughput* aumenta com o número de clientes. Como seria de esperar, como não está a ser utilizado o *handshake* RTS/CTS, todos os clientes enviam os pacotes em simultâneo e estes são recebidos na camada MAC do servidor, daí que o *throughput* aumente assintoticamente com o número de clientes. No entanto, inevitalmente muitos destes *frames* serão recebidos incorretamente devido a interferências entre as diferentes transmissões e serão perdidos na camada MAC, não contribuindo para o número de pacotes recebidos ao nível da aplicação, razão pela qual o *throughput* diminui ao nível da aplicação mas aumenta ao nível MAC.

Figura 10: *Throughput como função do número de clientes.*

2.3 Performance in Infrastructure Mode

Esta situação é bastante semelhante à da questão 2.2.1, com a excepção de que é usado um *access point* como intermediário entre o cliente e o servidor. Verifica-se então pelo gráfico (Figura 11) que os valores obtidos para o *throughput* são inferiores aos obtidos na alínea 2.2.1 e também decrescem mais rapidamente com o aumento do número de clientes. Analisando as estatísticas ao nível da camada MAC do *access point*, verificamos que isto se deve à existência de uma grande percentagem de frames que são perdidos devido ao *overflow* da fila de espera (*packetDropQueueOverflow*). De facto, o ritmo ao qual os pacotes chegam ao *access point* é superior ao ritmo ao qual este os consegue retransmitir, pelo que a fila eventualmente enche e há pacotes que são perdidos.

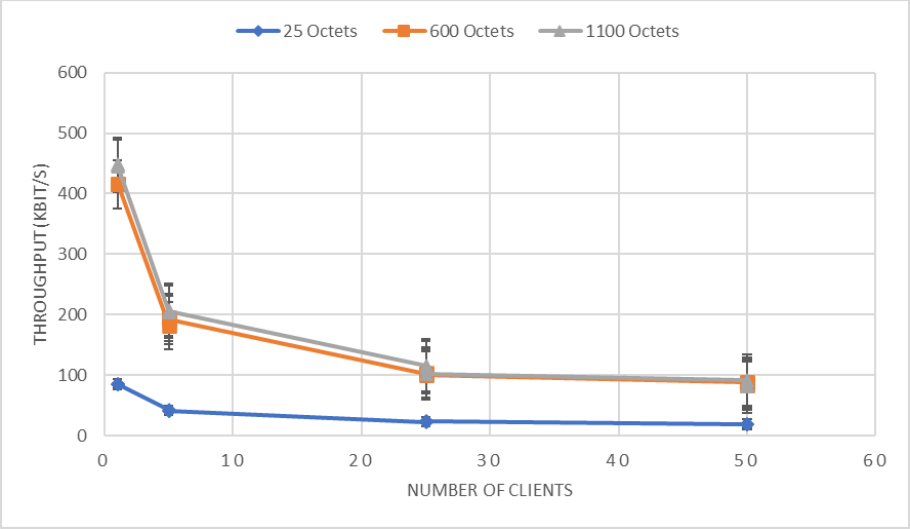

Figura 11: *Throughput como função do número de clientes.*

Na situação da alínea 2.2.1, como se tratava de uma ligação *ad-hoc*, os pacotes eram enviados pelos clientes e recebidos diretamente pelo servidor, pelo que não existia este problema uma vez que não passavam por filas de espera de retransmissão.