UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

BEATRIZ TRABBOLD PINTO ARRUDA

Análise do efeito do escorregamento de relógio no tempo de descoberta de vizinho para mecanismos de ciclo de trabalho assíncrono baseados em escalonamento

NITERÓI
2023

UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

BEATRIZ TRABBOLD PINTO ARRUDA

Análise do efeito do escorregamento de relógio no tempo de descoberta de vizinho para mecanismos de ciclo de trabalho assíncrono baseados em escalonamento

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica e de Telecomunicações. Área de concentração: Redes de Computadores.

Orientador:
Ricardo Campanha Carrano
Coorientador:
Diego Gimenez Passos
Coorientador:
Cledson de Oliveira

NITERÓI
2023

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

A773a Arruda, Beatriz Trabbold Pinto
Análise do efeito do escorregamento de relógio no tempo de descoberta de vizinho para mecanismos de ciclo de trabalho assíncrono baseados em escalonamento / Beatriz Trabbold Pinto Arruda. - 2023.
55 f.: il.

Orientador: Ricardo Campanha Carrano.
Coorientador: Diego Gimenez Passos; Cledson De Oliveira.
Dissertação (mestrado)-Universidade Federal Fluminense, Escola de Engenharia, Niterói, 2023.

1. Redes de Computadores. 2. Redes de Sensores. 3. Ciclo de Trabalho. 4. Produção intelectual. I. Carrano, Ricardo Campanha, orientador. II. Passos, Diego Gimenez, coorientador. III. De Oliveira, Cledson, coorientador. IV. Universidade Federal Fluminense. Escola de Engenharia.V. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

BEATRIZ TRABBOLD PINTO ARRUDA

Análise do efeito do escorregamento de relógio no tempo de descoberta de vizinho para mecanismos de ciclo de trabalho assíncrono baseados em escalonamento

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica e de Telecomunicações. Área de concentração: Redes de Computadores.

BANCA EXAMINADORA

Documento assinado digitalmente
gov.br
RICARDO CAMPANHA CARRANO
Data: 14/09/2023 10:35:35-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Ricardo Campanha Carrano – Orientador, UFF

Assinado por: Diego Gimenez Passos
Num. de Identificação: TR:PT-04N4K8485
Data: 2023.09.11 16:35:59 +0100

Prof. Dr. Diego Gimenez Passos – Coorientador, UFF

Documento assinado digitalmente
gov.br
CLEDSON OLIVEIRA DE SOUSA
Data: 11/09/2023 13:15:20-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Cledson de Oliveira – Coorientador, UFF

Documento assinado digitalmente
gov.br
NATALIA CASTRO FERNANDES
Data: 11/09/2023 15:55:45-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. Natalia Castro Fernandes, UFF

Documento assinado digitalmente
gov.br
RAPHAEL MELO GUEDES
Data: 13/09/2023 07:42:58-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Raphael Guedes, UERJ

Niterói

Julho de 2023

À minha família e a todos que me apoiaram ao longo dessa jornada

Agradecimentos

Primeiramente a Deus, pela saúde, por todas as oportunidades e pela proteção de em todos os momentos.

Aos amigos Raiane Lima, Orlando Marques e Lucas Barboza pela ajuda durante todas as etapas no decorrer dessa experiência acadêmica.

A minha mãe Marta Trabbold, por toda a educação e suporte incondicional fornecido durante todos esses anos.

Aos integrantes do laboratório Midiacom da Universidade Federal Fluminense, pelo auxílio no trabalho que ora se apresenta.

Aos professores Ricardo Campanha Carrano, Diego Passos e Cledson Oliveira pela orientação do trabalho de forma irretocável, paciência, comprometimento, auxílio intenso e sábios conselhos.

Resumo

O ciclo de trabalho é um mecanismo fundamental para redes sem fio operadas por bateria, como as redes de sensores sem fio. Devido à sua importância, este mecanismo é parte de diversos protocolos de acesso ao meio e outras tecnologias sem fio. Nos ciclos de trabalho assíncronos baseados em escalonamento, os nós ativam e desativam suas interfaces de rádio de acordo com um escalonamento de *slots* pré-projetado que garante a sobreposição de tempo de atividade entre dois nós, independente da diferença entre seus relógios internos, o que torna a comunicação possível. Esta dissertação apresenta dois novos modelos para predição do Tempo de Descoberta de Vizinho (ou NDT, do inglês *Neighbor Discovery Time*) em redes que operam com tais ciclos de trabalho. Estes modelos são chamados de unidirecional e bidirecional e, diferente dos modelos anteriores, eles consideram a possibilidade de os *slots* nos escalonamentos dos nós vizinhos não terem suas bordas perfeitamente alinhadas — o caso mais comum na prática. A importância de considerar este aspecto é demonstrada através de simulações. Assim como nas abordagens anteriores, o modelo unidirecional proposto possibilita até uma oportunidade de encontro por ciclo, porém, no modelo bidirecional, que representa o funcionamento real dos nós em uma rede RSSF, são possíveis até duas oportunidades de encontro por ciclo, o que resulta em um NDT menor.

Palavras-chaves: ciclo de trabalho, Redes de Sensores sem Fio, Escalonamento, Tempo de Descoberta do Vizinho.

Abstract

Duty cycling is a fundamental mechanism for battery-operated wireless networks, such as wireless sensor networks. Due to its importance, it is an integral part of several Medium Access Protocols and related wireless technologies. In Schedule-based Asynchronous Duty Cycle, nodes activate and deactivate their radio interfaces according to a pre-designed schedule of slots, which guarantees overlapping uptime between two neighbors, regardless of the offset between their internal clocks, making communication between them possible. This thesis presents two new models for the prediction of Neighbor Discovery Time (NDT) of Schedule-based Asynchronous duty cycle methods in networks that operate with such duty cycles, the unidirectional and the bidirectional. Differently from previous models, they account for the possibility of the slots in the schedules of the two neighbors not being perfectly border-aligned — the most common case in practice. The importance of considering this aspect is demonstrated through simulations. As in the previous approaches, the proposed unidirectional model allows up to one encounter opportunity per cycle, however, in the bidirectional model, which represents the real functioning of the nodes in a WSN network, up to two encounter opportunities per cycle are possible, which results in a smaller NDT.

Keywords: Duty Cycle, Wireless Sensor Networks, Schedule-based, Neighbor Discovery Time.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

PPGEET	Programa de Pós-graduação em Engenharia Elétrica e de Telecomunicações
IEEE	*Institute of Electrical and Electronic Engineers*
RSSFs	Redes de sensores sem fio
NDT	*Neighbor Discovery Time*
LPL	*Low Power Listening*
RFID	*Radio Frequency Identification*
CSMA	*Carrier Sense Multiple Access*
CCA	*Clear channel assessment*
MAC	*Media Acess Control*
ACK	*Acknowledge*
SFD	*Start frame delimiter*
BW	*Backoff window*
B-MAC	*Berkeley MAC*
RI-MAC	*Receiver-Initiated MAC*
PW-MAC	*Predictive-Wakeup MAC*

Sumário

Capítulo 1 - Introdução

A eficiência energética é fundamental para a maioria das redes de sensores sem fio (RSSFs), que são redes que possuem diversos nós sensores distribuídos com o objetivo de adquirir dados do ambiente e/ou detectar a ocorrência de eventos. Estes nós são de baixo custo e baixo gasto energético. RSSFs são um importante pilar do que se convencionou chamar Internet das Coisas e são utilizadas em diversas aplicações, como previsão do tempo, monitoramento militar, medicina, entre outras [1]. Os nós sensores utilizados nas RSSF são compactos e possuem uma bateria com duração limitada [2]. Por se localizar, em muitos casos, em áreas remotas [2], a troca ou a recarga da bateria de um sensor podem ser inviáveis e a vida útil da rede acaba sendo relacionada à duração de sua fonte de alimentação, o que torna a otimização do gasto energético um dos maiores desafios deste tipo de rede.

A arquitetura de uma RSSF é afetada por vários objetivos, como escalabilidade, tolerância a falhas, eficiência energética, entre outros. Os sensores presentes neste tipo de rede esgotam sua fonte de energia principalmente através do sensoriamento propriamente dito e da transmissão de dados entre nós sensores e a estação base. No entanto, em geral, a transmissão de dados entre nós consome mais energia do que a detecção e o processamento de informações [1]. Medições experimentais mostram que o custo para transmitir um único bit de informação é aproximadamente o mesmo necessário para processar mil operações internas em um nó sensor [30].

A limitação energética dos nós sensores é considerada um dos maiores desafios das redes RSSF, tendo em vista que, caso a energia acabe, o nó será inutilizado, podendo causar uma falha ou partição na rede. Com isso, a técnica do ciclo de trabalho se torna fundamental para diminuir o gasto energético dos nós e aumentar, assim, a vida útil da rede [2]. Esta técnica tem como principal objetivo reduzir o consumo de energia através da redução da escuta ociosa, que ocorre quando um rádio espera em vão por um quadro. Vale ressaltar que a escuta ociosa não é a única fonte de desperdício de energia. O *overhearing* (quando um nó desperdiça energia ao ouvir quadros que não são direcionados a ele) [2], por exemplo, também desperdiça energia.

A conservação de energia no ciclo de trabalho é alcançada ao se colocar o rádio (tanto transmissor quanto receptor) no modo de hibernação (baixa potência) sempre que a comunicação não for necessária. Idealmente, o rádio deve ser desligado assim que não houver mais dados a serem recebidos ou transmitidos e deve retomar seu funcionamento quando um novo pacote de dados ficar pronto para ser transmitido. Desta forma, os nós alternam entre períodos ativos e de hibernação, de acordo com a atividade da rede. Neste contexto, o termo ciclo de trabalho é também usado para denotar a fração de tempo em que os nós estão ativos durante sua vida.

Entretanto, os nós sensores executam uma tarefa cooperativa e, por isso, é necessária uma coordenação nos seus horários de atividade e hibernação. Logo, é necessária uma solução específica para que esta técnica possa ser realizada. Esta solução define quando os nós devem fazer a transição do modo de atividade para o de hibernação, de forma que os nós vizinhos estejam ativos ao mesmo tempo para que seja possível a troca de pacotes. As soluções utilizadas podem ser síncronas ou assíncronas. Esta dissertação irá focar em soluções assíncronas.

Existem diversas subcategorias de soluções assíncronas [3], que serão apresentadas com mais detalhes no Capítulo 2. Porém, este trabalho tem como foco o aperfeiçoamento nos modelos que calculam o NDT (*Neighbor Discovery Time*) em soluções assíncronas baseadas em escalonamentos. Estas soluções dependem de uma programação inteligente para despertar dos nós. Caso todos os nós empreguem esta mesma programação, haverá sobreposição de estados ativos, o que torna a comunicação possível.

1.1 Objetivos

O presente trabalho apresenta dois novos modelos para predição do NDT, que é o número de *slots* de tempo decorridos desde o momento em que um determinado nó decide enviar um quadro para um nó vizinho até o momento em que este vizinho recebe o quadro enviado com sucesso. Ao contrário dos modelos propostos anteriormente [9] [13] [3], neste trabalho é considerado um dos aspectos do escorregamento do relógio, o desalinhamento dos *slots*, o que torna seus resultados destas predições mais precisos.

1.2 Estrutura do Trabalho

O trabalho está organizado da seguinte forma. O Capítulo 2 apresenta uma revisão bibliográfica com a introdução de conceitos importantes para o modelo aqui definido, como tempo de descoberta do vizinho, *Block Designs* e escorregamento de relógio. O Capítulo 3 apresenta o histórico de pesquisas relacionadas que serviram como base para esse trabalho. O Capítulo 4 explica os modelos para descoberta do NDT que são propostos neste trabalho e detalha suas operações. O Capítulo 5 traz os resultados de simulações dos modelos propostos e, por fim, o Capítulo 6 apresenta considerações finais, conclusões e trabalhos futuros.

Capítulo 2 - Conceituação Teórica

Este capítulo apresenta uma explicação dos conceitos que serão importantes neste trabalho, como soluções assíncronas de ciclo de trabalho e protocolos de acesso ao meio que as incluem, NDT, *Block Designs* e escorregamento de relógio.

Como foi dito anteriormente, o objetivo principal do ciclo de trabalho é reduzir o consumo de energia do modelo através da redução da escuta ociosa, que é o tempo em que o nó está ativo, mas nenhum dado é enviado ou recebido [2]. Existem três classes de soluções de ciclo de trabalho: síncronas, assíncronas e híbridas [2]. Nas soluções síncronas, os nós sensores dormem e acordam de acordo com um *relógio*, o que garante que os nós fiquem ativos somente no momento oportuno para a transmissão de dados, economizando energia. A solução síncrona mais conhecida é o *Rendezvous* [17], onde todos os nós ligam e desligam seus rádios ao mesmo tempo. Contudo, estas soluções requerem sincronização entre os nós, o que causa uma sobrecarga de mensagens de controle e torna os nós sensores mais complexos e caros.

Por outro lado, em uma solução assíncrona, cada nó sensor dorme e acorda independentemente, sem seguir um *relógio*. Dado que os nós podem acordar em momentos diferentes, é necessário garantir que um transmissor e seu receptor estejam ativos no momento da transmissão dos dados. Temos como exemplos de soluções assíncronas o B-MAC [18] e o WiseMAC [20].

Por fim, nas soluções híbridas, a rede é dividida em *clusters* que são sincronizados internamente, enquanto a comunicação entre os *clusters* é realizada de forma assíncrona [14].

A sincronização de nós em uma rede sem fio pode ser considerada difícil e cara, exigindo às vezes *hardware* extra ou até mesmo a adição de tráfego de controle frequente, o que consome energia [3]. Com isso, o trabalho aqui apresentado segue a abordagem de [4] com foco no cálculo do NDT em soluções assíncronas, mais especificamente as baseadas em escalonamento.

2.1 Modelos assíncronos de ciclo de trabalho

As soluções de ciclo de trabalho assíncrono diferem entre si na forma pela qual buscam reduzir a escuta ociosa. Elas podem ser subcategorizadas e nesta seção serão apresentadas as subcategorias mais relevantes. São elas: amostragem de preâmbulo, transmissão iniciada pelo receptor, despertar sob demanda e ciclo de trabalho aleatório [3].

Em [5], é introduzida a técnica de Amostragem de Preâmbulo. Esta técnica também é conhecida como escuta de baixa potência (LPL – *Low Power Listening*) e foi inicialmente utilizada pelas soluções B-MAC [18] e WiseMAC [20]. O objetivo dos autores é reduzir a escuta ociosa através da transferência do gasto energético do receptor para o transmissor. Nesta técnica, cada nó dorme e acorda de forma assíncrona e, quando está acordado verifica a atividade do canal. Cada quadro é precedido por um preâmbulo mais longo que a duração dos tempos ativos e de sono somados. Isso garante que os nós terão tempo de acordar, detectar a transmissão do preâmbulo e ficar acordados para receber o respectivo quadro. Esta técnica apresenta alguns problemas, como a grande apropriação do canal por parte do preâmbulo, o que causa um desperdício de banda e impede que outros nós transmitam informações. Além disso, a latência fim-a-fim pode ser muito grande comparada a outros modelos. Por fim, há *overhearing*, uma vez que os nós desinteressados também permanecerão ativos enquanto escutam o preâmbulo [17].

A técnica de amostragem de preâmbulo foi aprimorada no X-MAC [16] com a introdução de técnicas de preâmbulo curto, que substituem o preâmbulo longo por uma série intermitente de pacotes curtos, cada um contendo o ID do nó de destino [16]. Esta série de pacotes é interrompível e dá ao receptor uma oportunidade antecipada de identificar a intenção do transmissor e diminuir o tempo de sinalização. Quando um nó acorda e recebe um pequeno pacote de preâmbulo, ele verifica o ID do nó de destino que está incluído no pacote. Se o nó não é o destinatário pretendido, ele volta a dormir imediatamente e continua seu ciclo de trabalho. Se o nó for o destinatário pretendido, ele permanecerá acordado para o receber o pacote de dados subsequente. Além disso, ao invés de um preâmbulo vazio, os pacotes curtos podem conter o endereço do receptor, o que diminui o *overhearing* [17].

Em [6], os autores apresentam uma solução baseada em transmissões iniciadas pelo receptor. Esta técnica é utilizada pelo RI-MAC. Nela, o transmissor espera por um *beacon* periódico do receptor e só inicia a transmissão depois que a sinalização é escutada. Ao contrário da amostragem de preâmbulo, o responsável por sinalizar que um quadro pode ser

transmitido é o receptor e, com isso, o preâmbulo é substituído por um *beacon* de sinalização muito mais curto, o que otimiza a utilização do meio de transmissão [17] [15]. Neste protocolo o gasto extra de energia recai sobre o receptor que, neste caso, deve permanecer ativo até receber a sinalização do transmissor pretendido. Contudo, os autores do RI-MAC relataram uma melhora significativa em relação ao X-MAC, principalmente quando vários fluxos de dados estão presentes. No entanto, a técnica de transmissão iniciada pelo receptor ainda possui um atraso fim-a-fim significativamente elevado [17].

Em [7], é apresentada a técnica de despertar sob demanda, que é baseada na ideia de que o rádio primário de um determinado nó pode ser despertado quando necessário. Para isso, é necessário que seja utilizada outra interface de comunicação, como um rádio despertador, que é um rádio de baixa potência que escuta um sinal de despertar e notifica a CPU ativar o rádio primário. Vale ressaltar que o rádio de despertar possui um consumo energético consideravelmente menor do que o rádio de dados (apenas alguns microwatts [17]). Alguns autores ainda propõem soluções onde o rádio de despertar completo é substituído por circuitos disparados por rádio que despertam o rádio primário do nó após a detecção de um sinal, assim como as etiquetas RFID [17]. Por adicionar uma nova interface de comunicação, o rádio de despertar, esta solução acaba se tornando mais complexa do que as vistas em [5] e [17].

Em [8], é apresentada a técnica de ciclo de trabalho aleatório. Esta técnica é utilizada pelo protocolo RAW e parte do princípio de que, em topologias densas, os nós podem adormecer e despertar aleatoriamente e, mesmo assim, há uma alta probabilidade de que haja vizinhos ativos aptos a receber a informação no momento em que eles tentam transmitir dados [17]. Esta técnica tem limitações consideráveis. Por exemplo, ela é restrita a cenários muito densos e o ciclo de trabalho deve ser ajustado cuidadosamente à quantidade de nós disponíveis. Caso contrário, as taxas de entrega são muito baixas. Isso significa que a técnica aleatória necessita de ajustes de topologia, o que aumenta a sua complexidade. As vantagens do ciclo de trabalho aleatório incluem uma distribuição justa da carga de tráfego (devido à aleatoriedade) e baixo atraso de ponta a ponta (devido a eliminação do tempo em que o nó está dormindo) [17].

2.2 Protocolos com utilização de ciclo de trabalho assíncrono

Nesta seção são apresentados alguns protocolos de acesso ao meio que incluem técnicas de ciclo de trabalho assíncronas, seu modo de operação e principais características.

2.2.1 B-MAC

O B-MAC [18] é um protocolo CSMA (*Carrier Sense Multiple Access*) [31] com serviços de rede, como organização, sincronização e roteamento. Para ter um consumo baixo de energia, o B-MAC emprega um modelo de amostragem de preâmbulo adaptável, o que busca minimizar o ciclo de trabalho e a escuta ociosa [18].

Este modelo utiliza o CCA (*Clear channel assessment*) e *backoffs* para arbitragem de canal, confirmações de camada de enlace para confiabilidade e LPL para comunicação de baixa potência.

Para receber dados de forma confiável, o comprimento do preâmbulo é ajustado ao intervalo em que o canal é verificado quanto à sua atividade, logo, o preâmbulo precisa ter, no mínimo, o mesmo tamanho do intervalo de verificação. Por exemplo, se o canal for verificado a cada 100 ms, o preâmbulo deve ter pelo menos 100 ms de comprimento para que o nó possa acordar, detectar atividade no canal, receber o preâmbulo e, em seguida, receber a mensagem. O tamanho do intervalo de verificação pode variar, sendo que, quanto menor o intervalo, menos energia é gasta e mais eficiente é o ciclo de trabalho. A escuta ociosa ocorre quando o nó acorda para fazer a amostragem do canal e não há atividade [18].

2.2.2 X-MAC

O X-MAC é uma solução semelhante ao B-MAC, porém emprega uma abordagem com preâmbulo abreviado, ao contrário do preâmbulo longo do B-MAC que introduz um excesso de latência em cada salto, mantendo as vantagens da escuta em baixa potência, ou seja, comunicação com baixo consumo de energia, simplicidade e desacoplamento entre os escalonamentos do transmissor e do receptor. A principal ideia do preâmbulo abreviado é incorporar informações de endereço do destinatário no preâmbulo para que os receptores

não-alvo possam voltar rapidamente ao estado de sono, o que mitiga o problema da escuta ociosa. Uma segunda melhoria em relação ao B-MAC é a utilização de um preâmbulo reduzido para permitir que o receptor alvo interrompa o preâmbulo assim que ele acordar e se der conta de que é o receptor alvo. A abordagem do preâmbulo abreviado reduz o tempo e a energia desperdiçados esperando que todo preâmbulo seja processado [16].

Um problema que ocorre em técnicas baseadas em amostragem de preâmbulo tradicionais é quando há uma série de transmissores esperando para enviar dados a um determinado receptor [19]. Depois que o primeiro transmissor começa a enviar pacotes de preâmbulo, os transmissores subsequentes ficarão acordados esperando até que o canal esteja livre. Consequentemente, cada transmissor transmite todo o seu preâmbulo sendo que o receptor foi acordado pelo primeiro transmissor da série. Para resolver este problema, no X-MAC, quando um transmissor pretende enviar, mas detecta um preâmbulo, ele escuta o canal e, se ouve um quadro de confirmação do nó para o qual deseja enviar os dados, o transmissor fará o *backoff* e, em seguida, irá enviar seus dados sem preâmbulo. O *backoff* aleatório é necessário pois pode haver mais de um transmissor esperando para enviar, e este *backoff* serve para mitigar colisões entre pacotes de vários transmissores [16]. Além disso, o *backoff* deve ser longo o suficiente para permitir que o transmissor inicial conclua sua transmissão de dados. Para habilitar esta técnica, após receber um pacote de dados, o receptor deve permanecer acordado por um breve período no caso de haver transmissões adicionais esperando o seu momento de envio. O período pelo qual um receptor permanece acordado após receber um pacote de dados é igual à duração máxima do período de *backoff* dos transmissores, para garantir que o receptor permaneça acordado por tempo suficiente para receber qualquer pacote de dados do transmissor adicional. Para garantir que o recebimento dos preâmbulos seja bem-sucedido e que não haja desconexão, a duração da sequência do preâmbulo deve ser maior do que o período máximo de sono do receptor [16].

2.2.3 WiseMAC

O WiseMAC, assim como o B-MAC e o X-MAC, é um protocolo de controle de acesso ao meio que utiliza a técnica de amostragem de preâmbulo. Nele, também é realizada a amostragem do meio, onde os nós sensores escutam o canal periodicamente por um curto período, com o objetivo de verificar a disponibilidade do canal para enviar dados. Se o meio amostrado estiver ocupado, um nó sensor o escuta novamente até que um quadro de dados

seja recebido ou que o meio fique ocioso. No lado do transmissor, um preâmbulo despertador de tamanho igual ao período de amostragem é adicionado na frente de cada quadro de dados para se certificar de que o receptor será acordado quando os dados do pacote chegarem [20].

O WiseMAC explora o conhecimento da programação de amostragem de seus vizinhos diretos para utilizar um preâmbulo de despertar com tamanho reduzido. Se algum nó não souber o padrão de ativação dos seus vizinhos, ele pode enviar um preâmbulo de duração T, a fim de checar o intervalo de amostragem do vizinho [21]. Neste modelo, os nós sensores aprendem o deslocamento entre a programação de amostragem dos seus vizinhos diretos e o seu próprio e, ao conhecer a programação de amostragem do destino, os nós sensores enviam as mensagens apenas no tempo certo com um preâmbulo de comprimento minimizado.

Cada nó mantém uma tabela atualizada com a programação de amostragem de seus vizinhos diretos. As informações de deslocamento da programação são obtidas por meio da inclusão da duração restante até a próxima amostragem de preâmbulo programada em cada pacote de confirmação. Com base nesta tabela, um nó pode determinar os intervalos de ativação de todos os seus vizinhos e minimizar o comprimento do preâmbulo para as próximas transmissões. Se um nó precisar enviar um quadro para um receptor já conhecido, ele aguarda o despertar do receptor, mantendo seu próprio transceptor no estado de sono, e transmite o quadro apenas no momento apropriado [22]. Este quadro transmitido apenas será precedido por um pequeno preâmbulo que compensa o desvio máximo que os relógios dos nós envolvidos podem ter desenvolvido desde o último agendamento. Assim como em [16] e [18], uma desvantagem deste modelo é que, depois que o primeiro transmissor começa a enviar pacotes de preâmbulo, os transmissores subsequentes ficarão acordados esperando até que o canal esteja livre. Logo, cada transmissor transmite todo o seu preâmbulo sendo que o receptor foi acordado pelo primeiro transmissor da série.

2.2.4 RI-MAC

O RI-MAC (*Receiver-Initiated MAC* – MAC iniciado pelo receptor) utiliza transmissão de dados iniciada pelo receptor para operar com eficiência em uma ampla gama de cargas de tráfego. Ele busca minimizar o tempo em que um transmissor e seu receptor ocupam o meio sem fio em busca de um momento de encontro para a troca de dados. Este

protocolo difere dos protocolos assíncronos estudados anteriormente (B-MAC, X-MAC e WiseMAC) no que diz respeito a como o transmissor e o receptor chegam a um encontro de tempo para que as informações possam ser transmitidas. Aqui, o transmissor permanece ativo e espera silenciosamente até que o receptor indique explicitamente quando iniciar a transmissão dos dados através do envio de um quadro de *beacon* [15].

Neste modelo, cada nó acorda periodicamente com base na sua programação para verificar se existe algum quadro de dados destinado a este nó. Após ligar o rádio, caso o meio esteja ocioso, um nó imediatamente transmite um *beacon* anunciando que está ativo e pronto para receber um quadro de dados. Um nó com dados pendentes para envio permanece ativo silenciosamente enquanto espera pelo sinal do receptor. Ao receber o *beacon*, o transmissor inicia o envio de dados imediatamente. Este envio será confirmado pelo receptor através do envio de outro *beacon* – nesse caso, um *beacon* ACK [15]. Vale ressaltar que a função do *beacon* ACK é dupla neste protocolo: primeiro, ele confirma o recebimento correto do quadro de dados enviado e, em segundo lugar, ele convida o transmissor a enviar um novo quadro de dados para o mesmo receptor. Se não houver dados após a transmissão de um *beacon*, o nó transmissor entra em modo de sono.

O RI-MAC reduz significativamente o tempo pelo qual um par de nós ocupa o meio antes de chegar a um momento de encontro onde possa ser realizada a troca de informações em comparação com a transmissão do preâmbulo no B-MAC, X-MAC e WiseMAC. O curto tempo de ocupação do meio permite que mais nós troquem quadros de dados com seus receptores pretendidos, o que ajuda a aumentar a capacidade da rede e, assim, a eficiência potencial [23]. Além disso, é permitido que um *beacon* sirva como um reconhecimento para o recebimento de dados e, ao mesmo tempo, como um pedido para o início da próxima transmissão de dados.

O controle de acesso ao meio entre transmissores que desejam transmitir quadros de dados para um mesmo receptor é feito principalmente pelo receptor. Esta escolha de projeto torna o RI-MAC mais eficiente em detectar colisões e recuperar quadros de dados perdidos do que o B-MAC e o X-MAC quando os transmissores estão ocultos um para o outro, o que pode ser comum em redes de sensores *ad hoc*. Aqui, após transmitir um *beacon*, um receptor detecta colisões dentro da janela de *backoff* (BW) especificada no *beacon*, que é muito mais curta do que o atraso de um intervalo de sono necessário no B-MAC e no X-MAC [23]. O RI-MAC também reduz a escuta ociosa, visto que um receptor espera pelos dados apenas dentro de uma pequena janela após a transmissão do *beacon*. Juntamente com o menor custo

de detecção de colisões e recuperação de quadros de dados perdidos, o RI-MAC atinge uma melhor eficiência energética, especialmente quando a carga da rede aumenta [24].

Como o transmissor pode iniciar a transmissão de dados apenas após receber um *beacon*, o receptor consegue saber o atraso máximo antes da chegada de um quadro. Este atraso pode ser calculado a partir do valor BW do *beacon* anterior. O receptor precisa apenas detectar o início do *Frame Delimiter* (SFD) para aprender sobre o primeiro quadro: se nenhum SFD é detectado a tempo, enquanto alguma atividade do canal é detectada pela verificação CCA, o receptor irá definir que houve uma colisão e irá gerar outro *beacon* com um valor de BW maior. No RI-MAC, este novo *beacon* é transmitido para que todos os rádios dos transmissores já estejam no modo de recepção. Antes de transmitir um *beacon*, um nó faz um *backoff* aleatório para evitar possíveis colisões com *beacons* de outros nós [19].

2.2.5 PW-MAC

O PW-MAC (*Predictive-Wakeup MAC*) é um protocolo onde cada nó acorda para receber os quadros em horários aleatórios e assíncronos. Ele reduz o consumo de energia do nó sensor, permitindo que os transmissores prevejam os tempos de despertar do receptor. Assim como o RI-MAC, ele é um protocolo iniciado pelo receptor, porém ele introduz o uso de uma sequência pseudoaleatória gerada de forma independente para controlar os tempos de despertar de cada nó, permitindo que os transmissores possam prever com precisão a hora que cada receptor irá acordar. Assim, enquanto o RI-MAC reduz o ciclo de trabalho apenas nos receptores, o PW-MAC reduz o ciclo de trabalho tanto nos receptores quanto nos transmissores [25]. Além disso, para prevenir que os transmissores percam o despertar dos receptores devido a fatores como latência e escorregamento, este protocolo apresenta um modelo de correção de erros de previsão sob demanda. Por fim, o PW-MAC fornece um modelo de retransmissão baseado em predição para alcançar alta eficiência energética, mesmo quando ocorrem colisões e os pacotes precisam ser retransmitidos [26].

Para permitir que um transmissor preveja com precisão os horários de despertar de um receptor, cada nó do PW-MAC deve calcular seus horários de ativação utilizando seu gerador de programação de ativação pseudoaleatória, ao invés de acordar em uma programação verdadeiramente aleatória. Nós diferentes têm sementes diferentes para seus geradores de números para evitar que eles gerem as mesmas sequências. Ao usar uma

programação de ativação pseudoaleatória em vez de uma fixa, a possibilidade de que os nós acordem ao mesmo tempo é reduzida e, com isso, as chances de colisões também diminuem. Assim como no RI-MAC, cada nó, ao despertar, transmite um *beacon* para anunciar que está ativo e pronto para receber pacotes de dados. Ao contrário do RI-MAC, em que o transmissor permanece acordado por, em média, meio intervalo de despertar, esperando o receptor antes de iniciar a transmissão de dados, o PW-MAC reduz este tempo de escuta ociosa para quase 0 [23].

No PW-MAC, os transmissores detectam as colisões, mudam para o estado de sono e escolhem de forma inteligente quando acordar e retransmitir os pacotes. Se um transmissor recebe o *beacon* de despertar do receptor, mas não recebe um *beacon* ACK para o pacote de dados enviado, ele reconhece que a transmissão do pacote de dados ou do ACK falhou. O transmissor, então, muda para o estado de sono e acorda na próxima janela de despertar prevista do receptor para retransmitir o pacote de dados, o que reduz a energia gasta na espera do receptor. Se dois transmissores enviarem pacotes de dados ao mesmo tempo para um receptor, tal colisão de transmissão é resolvida usando o mesmo modelo do RI-MAC, no qual o receptor detecta a colisão através do CCA e notifica os transmissores para retransmitirem seus pacotes após aumentarem as janelas de *backoff* aleatórias [23].

2.3 Métodos baseados em escalonamento

Nas soluções baseadas em escalonamento, os nós dividem o tempo em ciclos, subdivididos em *slots* ativos ou inativos, de acordo com uma programação selecionada, onde cada ciclo é uma repetição do anterior, como ilustrado na Figura 1.

Figura 1 - Exemplo de escalonamento com ciclos de 7 *slots*, onde três destes *slots* estão ativos. Adaptado de [3].

A literatura define alguns métodos baseados em escalonamento, como o GRID [32], Torus [33], Disco [34] e *Block Designs* [9]. Estes métodos serão discutidos nas próximas seções.

2.3.1 GRID

Este mecanismo apresenta a propriedade de fechamento para rotação, que consiste em um sistema de conjuntos onde quaisquer dois números apresentam interseções não nulas. No GRID é considerada uma matriz *n x n*, onde cada participante seleciona uma linha e coluna. Quando utilizado no cronograma de despertar de ciclos de trabalho assíncronos, cada nó divide o tempo em ciclos de n^2 *slots* e decide se eles estão ativos ou não, dependendo de sua posição na matriz. Nesta aplicação, para quaisquer dois nós deve haver pelo menos dois *slots* ativos em comum [32], conforme ilustrado na Figura 2.

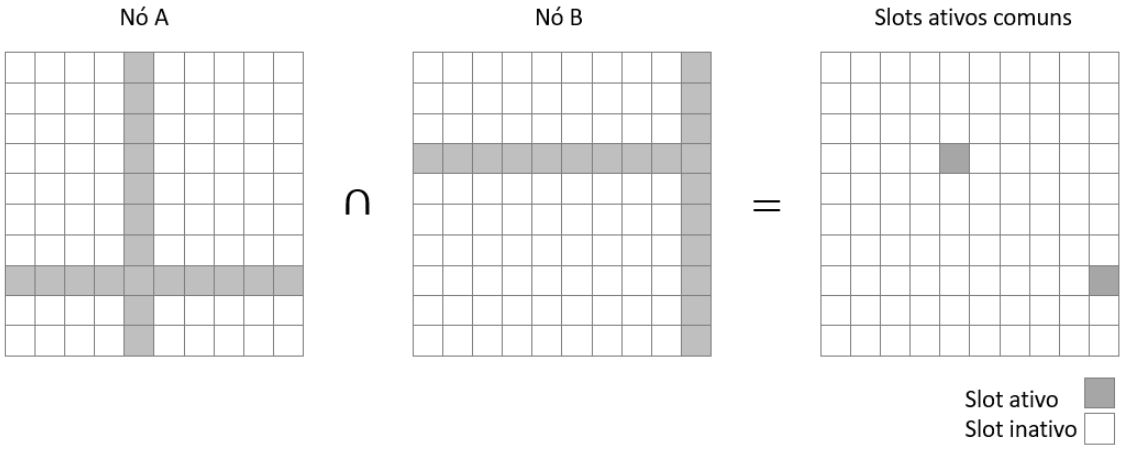

Figura 2 – Ilustra dois nós (A e B) operando no grid e suas interseções.

2.3.2 Torus

O Torus apresenta uma melhoria do GRID em relação ao ciclo de trabalho. Assim como o GRID ele apresenta a propriedade de fechamento para rotação. Este método se baseia no fato de que, para garantir a sobreposição de *slots* ativos, não é necessário selecionar uma linha completa da matriz. Em vez disso, se *n* é o número de colunas e *c* a coluna selecionada, basta que os nós selecionem metade mais um dos *slots* em cada coluna [33], como mostrado na figura 3.

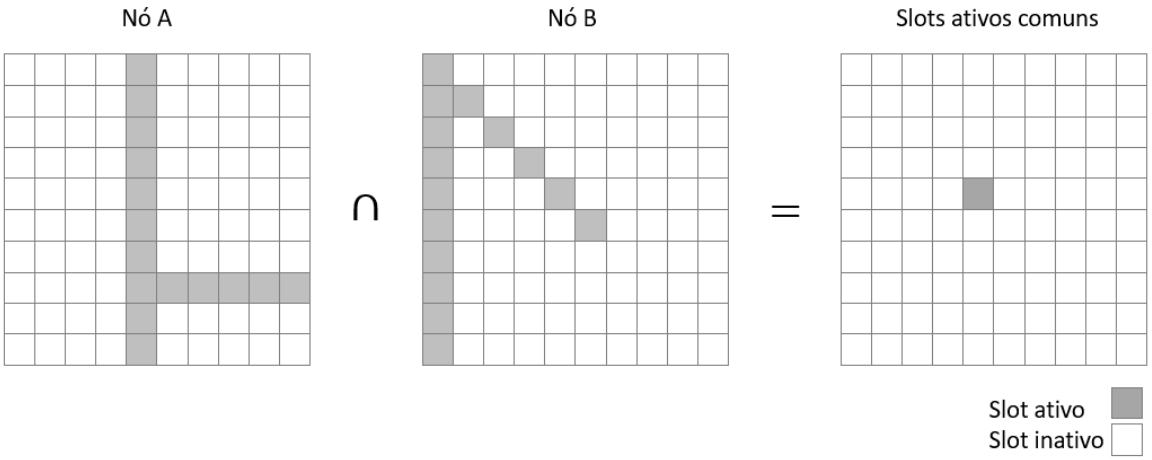

Figura 3 - Ilustração do modelo Torus em uma matriz 10x10.

2.3.3 Disco

O Disco é um método onde o cronograma de despertar possui dependência das propriedades dos números primos. Se dois nós selecionarem números primos diferentes (*m* e *n*) e ativarem cada *m*-ésimo e *n*-ésimo *slots* em um ciclo repetidamente, haverá um momento em que ambos estarão ativos, independente de seus *offsets*. Ele propõe que cada nó selecione dois números primos e ative os *slots* múltiplos a qualquer um dos primos, o que permite mais flexibilidade no ciclo de trabalho [34]. A figura 4 ilustra o modelo Disco para o nó A com os números primos 5 e 7 e o nó B com os primos 3 e 13.

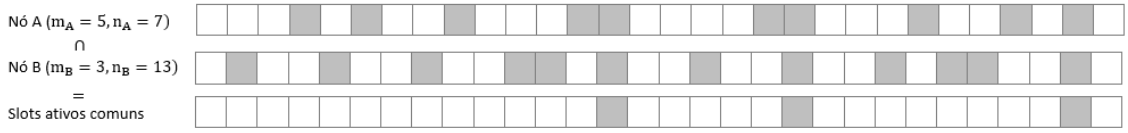

Figura 4 - Ilustração da operação do método Disco

2.3.4 *Block Designs*

Block Designs simétricos são esquemas combinatórios, *i.e.*, formas de arranjar, combinar ou selecionar elementos de um certo conjunto. Dado um conjunto finito V de v

elementos e inteiros k, λ ≥ 1, um *Block Design* representado por {v, k, λ} terá v blocos de k elementos e as seguintes propriedades [9]:

1. Todo e qualquer elemento de V ocorre exatamente em v blocos;
2. Quaisquer dois blocos terão exatamente λ elementos em comum.

Em aplicações de ciclo de trabalho assíncrono, os blocos do *Block Design* correspondem aos escalonamentos cíclicos dos nós, sendo cada elemento do bloco um *slot* de tempo ativo. Os nós têm um número fixo de *slots* ativos comuns (λ) em cada ciclo. Esta propriedade garante que quaisquer dois nós apresentem coincidências entre *slots* ativos, independentemente de seus deslocamentos de tempo, pois sempre haverá uma combinação com sobreposição de *slots* ativos. Além disso, pode-se constatar que dois nós operando sob um *Block Design* {v, k, λ} terão sobreposição de *slots* de tempo ativos, mesmo que suas bordas de *slot* não estejam alinhadas, ou seja, que os *slots* dos dois nós não iniciem no mesmo instante de tempo [3].

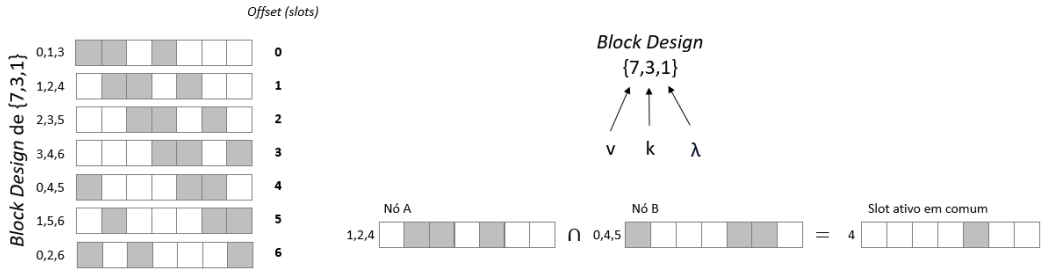

Figura 5 - Exemplo de *Block Design* onde ocorre interseção de *slots*. Adaptado de [3].

No exemplo da Figura 5, é possível verificar dois nós (A e B) operando sob dois blocos diferentes ([1,2,4] e [0,4,5]) com uma interseção no *slot* 4, o que permite a comunicação. O ciclo de trabalho em um *Block Design* é representado pela grandeza expressa por k/v, o que corresponde a 43% neste exemplo.

Além disso, é possível verificar que os escalonamentos gerados por cada bloco são simplesmente rotações (mudança de posição) dos escalonamentos de outros blocos do mesmo *Block Design*. Além disso, os *Block Designs* garantem a existência de exatamente λ *slots* ativos coincidentes por ciclo para qualquer *offset* (*i.e.*, em diferentes blocos) > 0 relativo entre os relógios dos nós [10]. Até o momento, a literatura sobre redes considerou principalmente, se não exclusivamente, uma categoria particular de *Block Design* chamada de Planos Projetivos. Um Plano projetivo é um *Block Design* que assume a forma $\{s^2+s+1,$

s+1, 1}, onde o parâmetro s é chamado de ordem. Conjectura-se que só existem planos projetivos para valores de s que são potências de primos [3].

Como exemplo, suponha-se um *Block Design* {7, 3, 1}, que consiste em sete blocos com três elementos cada. Um bloco corresponde a um escalonamento e os demais blocos de um mesmo *Block Design* são rotações do mesmo escalonamento. Logo, se dois nós operam sob um mesmo escalonamento derivado de um mesmo bloco de um *Block Design*, porém com um *offset* relativo diferente de 0, isso pode ser visto como cada nó operando sob escalonamentos derivados de blocos distintos do *Block Design* [3].

Tabela 1 - Exemplos de *Block Designs* com seus respectivos ciclos de trabalho e *slots* ativos

Design	ciclo de trabalho (%)	*slots* ativos
{7, 3, 1}	42,86	0, 1, 3
{183, 14, 1}	7,65	0, 12, 19, 20, 22, 43, 60, 71, 76, 85, 89, 115, 121, 168
{9507, 98, 1}	1,03	0, 1, 3, 37, 52, 191, 308, 332, 433, 914, 919, 984, 1093, 1155, 1231, 1238, 1600, 1678, 1723, 1732, 1755, 1773, 1826, 1930, 1938, 2099, 2116, 2141, 2457, 2712, 2859, 3058, 3187, 3466, 3524, 3655, 3675, 3748, 4139, 4145, 4183, 4297, 4301, 4518, 4528, 4600, 4720, 4777, 4964, 5043, 5054, 5176, 5268, 5329, 5356, 5496, 5526, 5601, 5617, 5851, 6151, 6173, 6491, 6539, 6759, 6778, 6792, 6878, 7021, 7163, 7226, 7290, 7490, 7650, 7747, 7860, 7941, 8028, 8056, 8154, 8304, 8339, 8370, 8438, 8450, 8505, 8534, 8574, 8797, 9005, 9048, 9094, 9107, 9133, 9154, 9270, 9326, 9400

2.4 Escorregamento de Relógio

No mundo real, não existe um escalonamento globalmente acordado e cada nó utiliza seu próprio relógio físico para ditar seu escalonamento. O escorregamento de relógio é um fenômeno que ocorre quando um relógio, devido a algum fator, tipicamente a qualidade de seu cristal, perde a sincronização com algum relógio de referência. Ou seja, o relógio

dessincroniza gradualmente do referencial. Todos os relógios estão sujeitos ao escorregamento, causando eventuais divergências, a menos que sejam ressincronizados [11].

A maioria dos nós em redes de sensores sem fio utiliza cristais ou osciladores cuja taxa máxima de escorregamento do relógio é de 30 ppm [12], o que causa um erro de pelo menos 30 μsegundos a cada 1 segundo. Ao contrário das soluções de ciclos de trabalho síncronas, que utilizam mensagens de sincronização para ajustar os relógios dessincronizados, o escorregamento do relógio em soluções assíncronas causa estimativa imprecisa do momento de despertar do nó, o que aumenta a sobrecarga da comunicação, pois mais mensagens são perdidas no momento do envio [12]. Como em redes que operam com ciclo de trabalho é necessário que pelo menos dois nós estejam ativos simultaneamente para que a comunicação possa ser realizada com sucesso, o escorregamento do relógio pode afetar o escalonamento e, em algum momento, os *slots* ativos podem não ser mais coincidentes, o que impossibilita a troca de informações entre os nós e prejudica o funcionamento da rede. Logo, considerar o fenômeno do escorregamento de relógio na análise do NDT é necessário para que se possa mensurar esta grandeza com mais precisão.

Capítulo 3 - Trabalhos Relacionados

Vários trabalhos da literatura abordam o problema da modelagem do NDT. No entanto, nenhum destes modelos considera o desalinhamento dos *slots*, causado pelo escorregamento do relógio, em sua modelagem.

Em [9], foi apresentada uma solução assíncrona de ciclo de trabalho baseada em *Block Design*, onde o tempo era dividido em *slots* de comprimento fixo que poderiam ser ativos ou inativos de acordo com um escalonamento pré-definido. Ao contrário de trabalhos mais modernos, o modelo adotado pelos autores para a avaliação do NDT não considerava a probabilidade de sucesso do enlace (p) e, por isso, seus resultados se mostraram imprecisos se comparados com abordagens mais recentes.

Em [13], foi apresentado um novo modelo para estimar o NDT que incorporava as perdas de mensagens e deslocamentos de tempo relativos entre os vizinhos, porém ainda utilizava a discretização do tempo em *slots*, assumindo que os *slots* dos dois nós possuem suas bordas alinhadas. O modelo proposto considerava dois nós (A e B) operando de forma assíncrona sob um *Block Design* {v, k, λ}. Supondo que A e B operam em diferentes *offsets* (*i.e.*, em diferentes blocos), era definido e_i, $1 < i < \lambda$, como o *i-ésimo slot* ativo em comum entre dois nós em um dado ciclo (*i.e.*, *i-ésima* oportunidade de encontro em um ciclo). Os trabalhos estudados anteriormente consideravam, em sua maioria, planos projetivos, que são uma categoria de *Block Designs* onde o λ é 1 e onde só é possível uma oportunidade de descoberta por ciclo. Como a expressão apresentada em [13] leva o parâmetro λ em consideração, foi possível analisar se *Block Designs* com valores maiores de λ são vantajosos e sob que circunstâncias. A conclusão foi que, embora os planos projetivos forneçam um ciclo de trabalho mínimo, o aumento na frequência de oportunidades de encontro causada pelo aumento de λ pode reduzir o NDT à medida que a qualidade do link se deteriora (valores baixos de p).

Em [3], é apresentada uma expressão para determinar o NDT quando as oportunidades de encontro são distribuídas em intervalos de tempo arbitrários, levando em consideração a probabilidade de entrega da mensagem, sendo esta incluída no parâmetro de probabilidade de entrega. Nesta expressão, são considerados dois nós (A e B) operando sob o mesmo escalonamento cíclico, formado por um certo número de intervalos de tempo, onde em apenas alguns *slots* os nós se encontram ativos e levando em consideração que as

programações não estão sincronizadas. Esta expressão apresenta o número esperado de intervalos de tempo que decorrerão até que A escute a primeira mensagem de B, ou seja o NDT, em *slots*. Este modelo também não considerava os efeitos do alinhamento de bordas no NDT e, por isso, seus resultados não são tão precisos.

O trabalho apresentado em [3] propôs a Equação (1) para estimar o NDT para *Block Design*. Ela foi obtida por análise combinatória de todas as possíveis interseções entre os escalonamentos de dois nós não sincronizados operando com deslocamentos aleatórios. Esta equação considera os parâmetros *v* (número de *slots* por ciclo) e λ (número de oportunidades de encontro por ciclo) do *Block Design* e a probabilidade de sucesso *p*.

$$NDT = \frac{v+1}{p(\lambda+1)} - \frac{(v+1)(1-p)^{\lambda}-(\lambda+1)}{(\lambda+1)(1-p)^{\lambda}-1} \quad (1)$$

Esta expressão mostra que, para um determinado ciclo, designs com maior λ irão reduzir o NDT às custas de um ciclo de trabalho mais elevado.

A Tabela 1 apresenta uma comparação entre os modelos apresentados em [9], [13] e [3].

Tabela 1 - Comparação entre modelos apresentados nos trabalhos relacionados.

Modelo	Discretização do tempo em *slots*	perda de mensagens	Diferença de *offset*	Desalinhamento de *slots*
[9]	Sim	Não	Não	Não
[13]	Sim	Sim	Sim	Não
[3]	Sim	Sim	Sim	Não

Capítulo 4 - Modelos Propostos

Nesta dissertação, são propostos dois novos modelos para predição do NDT, o unidirecional e o bidirecional, que buscam apresentar o tempo médio para descoberta de um nó vizinho. Ao contrário dos modelos propostos anteriormente, estes consideram um dos aspectos do escorregamento do relógio, o desalinhamento dos *slots*, o que torna seus resultados mais precisos.

Assim como em [3], nos modelos aqui apresentados, são considerados dois nós (A e B) operando sob o mesmo escalonamento cíclico, formado por um certo número de intervalos de tempo (*slots*). Também como em [3], assume-se que os *slots* sejam longos o suficiente para a transmissão e recepção de ao menos um *beacon* e assume-se que os nós possam apresentar um *offset* aleatório entre seus escalonamentos.

As propostas aqui apresentadas se diferenciam das anteriores, pois modelam o desalinhamento dos *slots*. Isso é obtido através da divisão dos *slots* em *subslots* — ou *fatias* —, onde os nós se encontram ativos em apenas alguns *subslots* de acordo com o escalonamento avaliado. O número de *subslots* de cada *slot* é um parâmetro do modelo proposto e pode variar. A ideia de subdividir os *slots* em *subslots* tem como objetivo aproximar a modelagem do funcionamento dos escalonamentos no mundo real — onde o tempo é contínuo. Quanto maior o número de *subslots* nos quais cada *slot* é fatiado, maior a resolução dos modelos em representar este tempo contínuo. Desta forma, acredita-se que, quanto maior o número de fatias utilizado, mais realístico e preciso serão os modelos. No limite, quando o número de fatias tende a infinito, os modelos seriam uma representação fiel ao cenário real em termos deste aspecto.

Uma vez que A decide enviar um quadro para seu vizinho, para que a transmissão seja realizada com sucesso, ou seja, para que ocorra uma oportunidade de encontro, tanto para o modelo unidirecional quanto para o bidirecional é necessário que ambos os nós estejam em *subslots* ativos simultaneamente. Além disso, no caso unidirecional, a transmissão só é iniciada caso o primeiro *subslot* de um *slot* do transmissor coincide com qualquer *subslot* ativo do receptor. Já no caso bidirecional, só é necessário que ou o receptor ou o transmissor esteja no primeiro *subslot* de seu respectivo *slot*. Vale ressaltar que a divisão de *slots* em *subslots* torna possível a análise do caso bidirecional, pois torna viável o início

de uma comunicação no meio de um *slot* (que está fatiado), o que não é possível no modelo tradicional onde não acontece o fatiamento.

Nos modelos aqui propostos, assume-se que o enlace sem fio é caracterizado por uma recepção de quadro com probabilidade *p*. Essa probabilidade engloba fatores como a potência de transmissão de A, ruído, interferência, largura de banda, modulação, codificação e eventuais colisões. Assim, mesmo em uma oportunidade de encontro, uma tentativa de transmissão pode falhar com probabilidade 1 – *p* [3].

No Capítulo 5 deste trabalho, são apresentados resultados de experimentos realizados com divisões de 2, 4, 32 e 64 *subslots*. Ambos os modelos assumem que, no início de cada *slot* ativo (ou seja, no primeiro *subslot*), um nó transmite um *beacon* e passa o resto do seu tempo ativo aguardando por *beacons* dos nós vizinhos. Também se assume que, ao ouvir um *beacon* válido, um nó permanece com seu rádio ligado para receber o quadro correspondente até o final, independentemente de seu escalonamento.

4.1 Modelo Unidirecional

No modelo unidirecional um dos nós é definido arbitrariamente como transmissor e apenas ele é capaz de iniciar a transmissão. Neste modelo considera-se que há uma *oportunidade de encontro* entre dois nós se (i) ambos estiverem em *subslots* ativos (rádios ligados simultaneamente) e (ii) se o primeiro *subslot* de um *slot* do transmissor coincide com qualquer *subslot* ativo do receptor. Vale ressaltar que este modelo é apenas teórico e não representa o funcionamento de uma RSSF.

Na Figura 6, é apresentado um exemplo de funcionamento do modelo unidirecional proposto para o escalonamento dado pelo *Block Design* {7, 3, 1}, com cada *slot* dividido em dois *subslots*. No exemplo, há um *offset* entre os relógios internos de A e B de 1,5 *slot* (o que equivale a 3 *subslots*). A terceira linha mostra em quais *subslots* ocorrem as oportunidades de encontro. Como pode ser visto, a oportunidade de encontro ocorre quando o primeiro *subslot* de um *slot* ativo de A, que no exemplo é o transmissor, encontra um *subslot* ativo de B, que é o receptor. Neste exemplo, a primeira oportunidade de encontro ocorre em um *subslot* em que o nó A (transmissor) inicia a transmissão do seu *beacon*, pois trata-se do primeiro *subslot* de um *slot* ativo. Por sua vez, B (receptor) está no segundo *subslot* de um *slot* ativo.

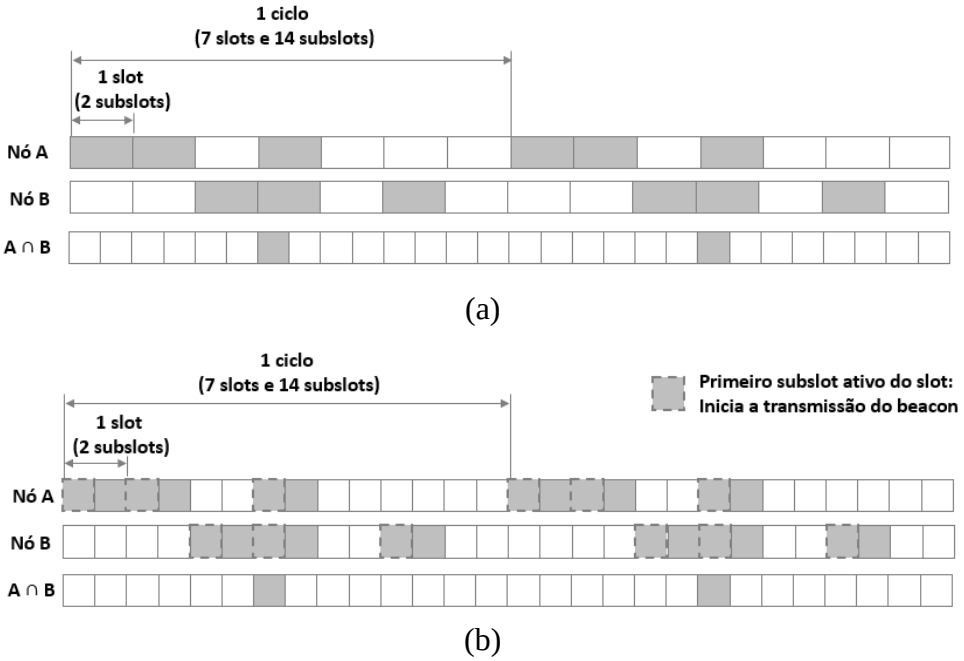

Figura 6 - Exemplo de funcionamento do modelo proposto para o escalonamento com o *Block Design* {7, 3, 1}, divisão em 2 *subslot*s e *offset* de 2 *slots*: (a) visualização sem quebra de *subslots*.

A Figura 7 apresenta um segundo exemplo do modelo unidirecional, mas agora considerando um *offset* de 2 *slots* (4 *subslots*). É possível verificar que, assim como no caso do *offset* de 1,5 *slot*, é prevista uma única oportunidade de encontro por ciclo. Este valor é consistente com o que se espera de um escalonamento baseado em *Block Designs* (particularmente, planos projetivos), dado que estes garantem exatamente um *slot* coincidente por ciclo para qualquer *offset* que não seja 0 ou múltiplo de *v*.

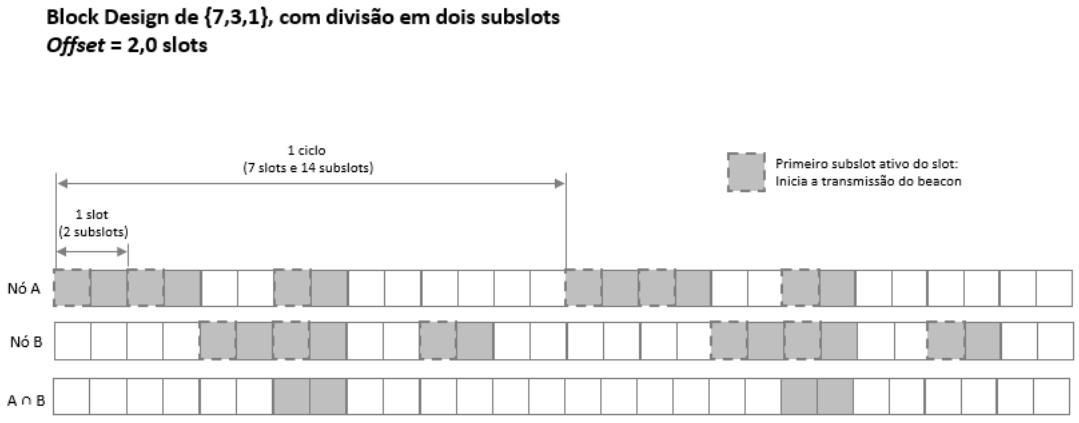

Figura 7 - Exemplo de funcionamento do modelo unidirecional para o escalonamento com *Block Design* de {7, 3, 1}, divisão em 2 *subslots* e *offset* de 2 *slots*.

4.2 Modelo Bidirecional

No modelo bidirecional, considera-se que só há uma *oportunidade de encontro* entre dois nós caso (i) ambos estiverem em *subslots* ativos (para terem seus rádios ligados simultaneamente) e (ii) se um deles estiver no primeiro *subslot* de seu respectivo *slot* (para que transmita um *beacon*). Caso *p* seja zero, o NDT tende ao infinito, pois a troca de informações não é possível pelo enlace. Ao contrário do modelo unidirecional, onde o primeiro *subslot* de um *slot* do transmissor deveria coincidir com qualquer *subslot* ativo do receptor, aqui é necessário que pelo menos um dos nós (ou transmissor ou receptor) esteja no primeiro *subslot* de seu respectivo slot, o que caracteriza uma comunicação bidirecional, que é a utilizada em redes RSSF no momento da descoberta de vizinhos.

Na Figura 8, é apresentado um exemplo de funcionamento do modelo bidirecional para o escalonamento dado pelo *Block Design* {7, 3, 1}, com cada *slot* dividido em dois *subslots*. No exemplo, há um *offset* entre os relógios internos de A e B de 1,5 *slot* (o que equivale a 3 *subslots*). A terceira linha mostra em quais *subslots* ocorrem as oportunidades de encontro. Como pode ser visto, as oportunidades de encontro ocorrem sempre no primeiro *subslot* de cada *slot* ativo de A ou de B, desde que o outro nó esteja também em um *subslot* ativo (qualquer). Neste exemplo, a primeira oportunidade de encontro ocorre em um *subslot* em que o nó B inicia a transmissão do seu *beacon*, pois trata-se do primeiro *subslot* de um *slot* ativo. Por sua vez, A está no segundo *subslot* de um *slot* ativo. Já na segunda oportunidade, A inicia a transmissão de um *beacon*, pois está no primeiro *subslot* de um de seus *slots* ativos, enquanto o *subslot* ativo de B não é o primeiro do *slot* correspondente.

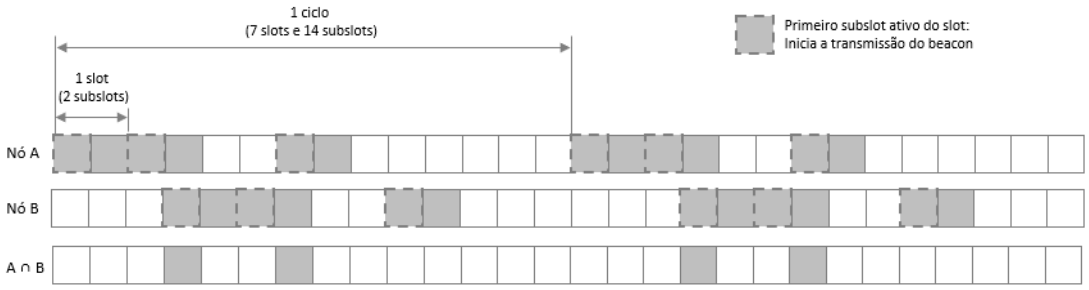

Figura 8 - Exemplo de funcionamento do modelo bidirecional para o escalonamento com *Block Design* de {7, 3, 1}, divisão em 2 *subslots* e *offset* de 1,5 *slot*.

A Figura 9 apresenta um segundo exemplo do modelo bidirecional, mas agora considerando um *offset* de 2 *slots* (4 *subslots*). É possível verificar que, neste caso, assim como no modelo unidirecional, é prevista uma única oportunidade de encontro por ciclo. No caso do modelo bidirecional, é esperado um NDT menor em relação ao modelo unidirecional.

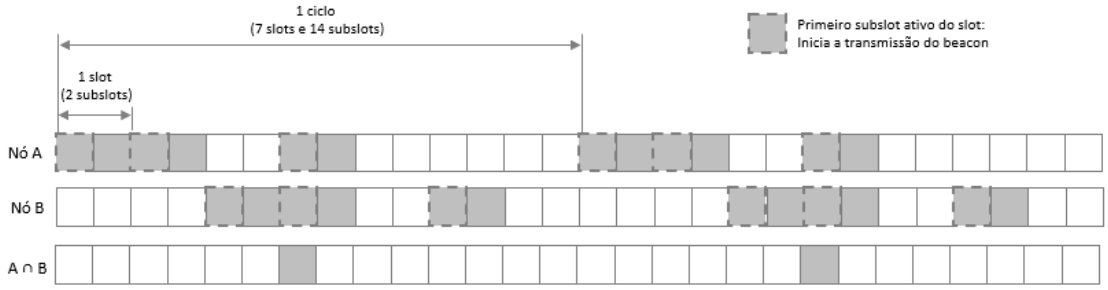

Figura 9 - Exemplo de funcionamento do modelo bidirecional proposto para o escalonamento com *Block Design* de {7, 3, 1}, divisão em 2 *subslots* e *offset* de 2 *slots*.

4.3 Comparação com modelos anteriores

A título de comparação, a Figura 10 ilustra o modelo tradicional proposto em [3] considerando um *offset* de 2 *slots*. O modelo de [3] não considera a subdivisão dos *slots* em *subslots* e, portanto, assume que uma comunicação só será bem-sucedida caso os *slots* de A e B estejam alinhados o que, na prática, pode causar colisões. Conforme ilustrado na Figura 10, o número de oportunidades de encontro previstas pelo modelo tradicional é o mesmo do modelo unidirecional, ilustrado na Figura 6: uma oportunidade por ciclo.

Por outro lado, no exemplo do caso bidirecional ilustrado na Figura 8, foi considerado um *offset* de 1,5 *slot*. Este valor não-inteiro de *offset* significa que os *slots* de A e B não se encontram perfeitamente alinhados: ao contrário, os *slots* de A começam exatamente no meio dos *slots* de B. Conforme mostrado na figura, para este caso, o modelo bidirecional proposto prevê até duas oportunidades de encontro a cada ciclo do escalonamento, ao invés de uma única, como no caso de um *offset* inteiro.

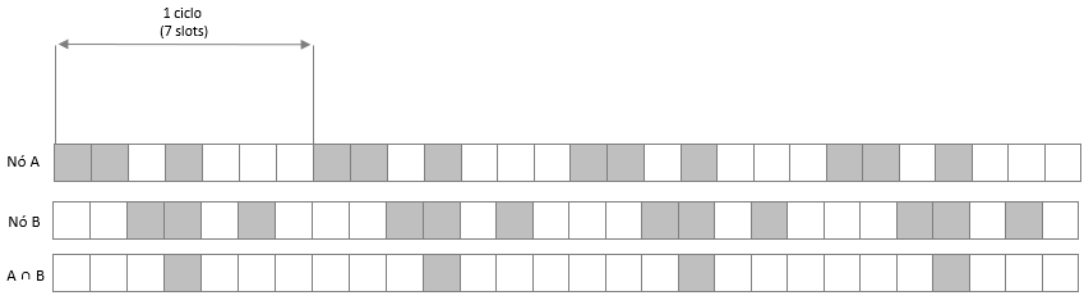

Figura 10 - Exemplo de funcionamento do modelo proposto em [3] para o escalonamento com *Block Design* de {7, 3, 1} e *offset* de 2 *slots*.

De fato, um *offset* não-inteiro *x* pode ser entendido como uma superposição entre os *offsets* ⌊*x*⌋ e ⌈*x*⌉: se considerarmos a interseção entre os *slots* ativos de A e B, há sobreposição parcial dos *slots* que seriam comuns para o *offset* ⌊*x*⌋ e também dos *slots* que seriam comuns para o *offset* ⌈*x*⌉. Esta característica dá origem a uma quantidade maior de oportunidades de encontro (para *Block Designs*, o dobro) justamente quando os *slots* de A e B não estão perfeitamente alinhados.

Desta forma, conclui-se que, ao não levar em conta a possibilidade de *slots* desalinhados, os modelos tradicionais só consideram oportunidades de encontro que ocorrem no início dos *slot*s o que, na maioria das vezes, não é o que ocorre e, por isso, ao serem comparados com o modelo unidirecional, apresentam um valor de NDT menor. Já ao se comparar o modelo tradicional com o modelo bidirecional, é possível concluir que os tradicionais subestimam a frequência de ocorrência de oportunidades de encontro e, portanto, superestimam o NDT.

Além da possibilidade de desalinhamento de *slots*, outro aspecto que não é levado em conta em modelos de NDT na literatura é o efeito do atraso de propagação. Se, em uma dada rede, o atraso de propagação dos enlaces for da ordem de um ou mais *slots*, então a hipótese de que as oportunidades de encontro ocorrem quando A e B têm seus rádios ligados simultaneamente se torna falsa – já que o sinal correspondente ao *beacon* do transmissor só chegaria ao receptor depois de um tempo correspondente a um ou mais *slots*.

Embora o atraso de propagação não seja um parâmetro explícito dos modelos propostos nesta dissertação, seu efeito é implicitamente capturado. Isso ocorre porque o efeito do atraso de propagação pode ser contabilizado no próprio *offset* entre os nós. Por

exemplo, se o *offset* entre os relógios internos de A e B é de 5 *slots* e o atraso de propagação é de 2 *slots*, então, do ponto de vista das oportunidades de encontro, o cenário se torna equivalente ao caso de um *offset* de 7 *slots* com um atraso de propagação nulo. Embora um raciocínio similar possa ser feito para os modelos existentes na literatura, a capacidade dos modelos aqui propostos de lidar com *offsets* não-inteiros os torna também mais adequados para capturar os efeitos do atraso de propagação (que, em geral, não será um múltiplo da duração dos *slots*).

Capítulo 5 - Validação dos Modelos

Para validar os modelos propostos, foram desenvolvidos dois códigos em *Python*, um para o modelo unidirecional e um para o bidirecional, que simulam os escalonamentos e retornam medidas de NDT máximo, NDT médio e NDT mínimo. Como entrada, o simulador recebe o escalonamento a ser analisado, a probabilidade de sucesso do enlace simulado, o número de repetições desejadas e o número de *fatias* do *slot*. São, então, realizadas tantas repetições quanto solicitadas de simulações do processo de encontro entre dois nós A e B. Em cada repetição, o simulador sorteia aleatoriamente *slots* iniciais para A e B, o que tem como consequência *offsets* aleatórios entre os dois nós, e simula os escalonamentos e transmissões de *beacons* — considerando a probabilidade de sucesso *p* especificada — até que haja um sucesso. Quando isto ocorre, o tempo decorrido para aquela repetição é armazenado para que, posteriormente, estatísticas sejam calculadas. Vale ressaltar que o simulador não gera pacotes nem simula dinamicamente características do enlace sem fio. Para simular uma tentativa de transmissão, ele sorteia um número aleatório entre 0 e 1 e compara o valor sorteado com o da probabilidade *p*. Caso o valor sorteado seja menor ou igual a probabilidade, considera-se uma transmissão bem-sucedida e, caso este valor seja maior, considera-se que a transmissão não foi realizada com sucesso.

5.1 Modelo Unidirecional

Na primeira análise do modelo unidirecional, foram realizadas simulações com o *Block Design* {7, 3, 1}. As simulações realizadas variaram o número de *subslots* e a probabilidade de sucesso do enlace (*p*). Foram considerados cenários com 1, 2, 4, 16, 32 e 64 *subslots* por *slot* e $p \in \{0{,}1;\ 0{,}2,\ 0{,}3;\ 0{,}4;\ 0{,}5;\ 0{,}6;\ 0{,}7;\ 0{,}8;\ 0{,}9;\ 1{,}0\}$ para as simulações. Para cada combinação de parâmetros, foram realizadas 10.000 repetições. Além disso, para que fosse possível visualizar a diferença percentual das curvas, os valores de NDT do gráfico da Figura 11 foram normalizados pelo NDT para uma fatia, ou seja, o NDT resultante foi dividido pelo número de fatias.

Esta primeira análise buscou verificar como o NDT médio se comportava ao variar a probabilidade de sucesso na transmissão *p* para diferentes quantidades de *subslots*. Como

pode ser visto na Figura 11, a diferença nos valores do NDT médio para *p* próximo de 1,0 é significativa, especialmente à medida que o número de fatias aumenta. Esta diferença se torna menor à medida que os valores de *p* caem. Além disso, esta figura mostra que a simplificação encontrada na literatura – o alinhamento inicial dos *slots* – traz erros significativos que aumentam à medida que a análise se aproxima do tempo contínuo. Por exemplo, para 64 fatias, como pode ser visto através da simulação representada na Figura 11, o erro de estimativa do NDT já é superior a 17,5% em enlaces com boas condições.

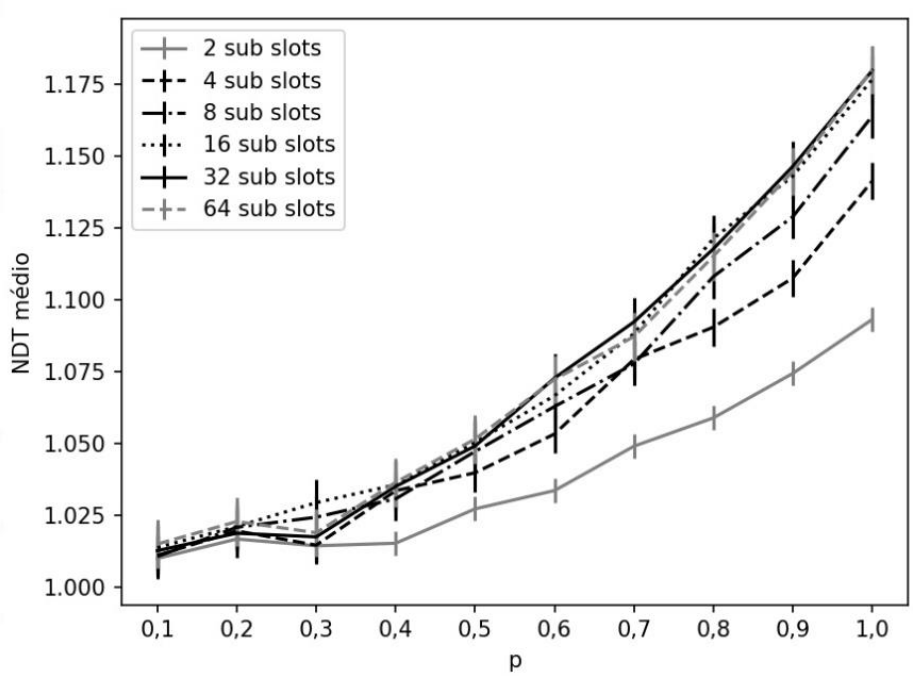

Figura 11 - NDT médio (normalizado pelo caso de 1 *subslot* por *slot*) pela probabilidade de sucesso (*p*) para o *Block Design* de {7, 3, 1}.

Ainda para o modelo unidirecional, para entender melhor o efeito do número de *fatias* no modelo, foi realizada uma segunda análise que buscou relacionar este parâmetro com o NDT médio para os escalonamentos {7, 3, 1} e {183, 14, 1}, como mostrado na Figura 12. Para esta análise, foi considerada uma probabilidade de sucesso de 1. É possível observar que o comportamento qualitativo das curvas é idêntico para os dois escalonamentos estudados, apesar da diferença na escala. Além disso, essa análise confirma o que foi observado na análise anterior de que o valor do NDT médio aumenta com o aumento do número de *fatias* — aproximadamente 0,5 *slots* em relação ao valor previsto para 1 fatia quando o número de *fatias* tende ao infinito. Isto acontece pois os modelos tradicionais assumem que as oportunidades de encontro começam no início do *slot*, o que, na maioria das vezes, não ocorre. Através destas observações, conjecturou-se que, para $p = 1$, o NDT

calculado em uma simulação com um determinado número de *fatias* pode ser aproximado pela função (2):

$$NDT_f = NDT_1 + \frac{(f-1)}{(2\times f)} \quad (2)$$

onde *f* é o número de *fatias* e NDT_1 é o valor do NDT médio para uma fatia. Esta função também está plotada nos gráficos da Figura 12 (pontilhada) para comparação e mostra que, à medida que a quantidade de fatias aumenta, o erro na estimativa do NDT se aproxima a 0,5 *slot*.

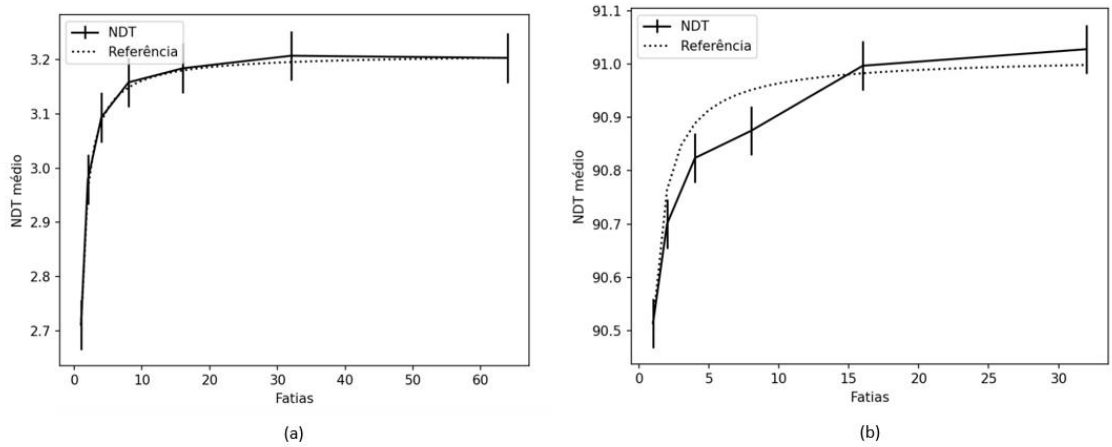

Figura 12 - NDT médio pela quantidade de fatias para os *Block Designs* (a) {7, 3, 1} e (b) {183, 14, 1}.

5.2 Modelo Bidirecional

Para o modelo bidirecional, foram realizadas simulações com os *Block Designs* {7, 3, 1}, {183, 14, 1} e {9507, 98, 1}. As simulações realizadas variaram o número de *subslots* e a probabilidade de sucesso do enlace (*p*). Foram considerados cenários com 1, 2, 4, 16, 32 e 64 *subslots* por *slot* e p ∈{0,1, 0,2, 0,3, 0,4, 0,5, 0,6, 0,7. 0,8, 0,9, 1,0} para as simulações. Para cada combinação de parâmetros, foram realizadas 5.000 repetições.

A primeira análise realizada buscou verificar como o NDT médio se comportava ao variar a probabilidade de sucesso na transmissão *p* para diferentes quantidades de *subslots*. Como pode ser visto nas Figuras 13 e 14, independente do escalonamento escolhido, o comportamento das curvas é semelhante. Para p << 1, à medida que o número de *subslots* por *slot* aumenta, o NDT tende a cair pela metade em relação ao caso de uma fatia. Isso é consistente com a observação feita na Seção 4 de que o número de oportunidades de encontro

aumenta — em particular, dobra — para *offsets* não-inteiros: com mais *fatias* a chance de *offsets* não-inteiros cresce, o NDT cai. Em particular, para valores mais baixos de *p*, são tipicamente necessárias várias tentativas de transmissão de *beacon* até o sucesso, causando uma influência direta entre o número de oportunidades de encontro por ciclo e o NDT. Por outro lado, para *p* próximo de 1, o NDT cai com o aumento do número de *fatias*, porém não pela metade. Isso ocorre porque o tempo até a primeira oportunidade de encontro se torna mais relevante e este pode não ser diretamente relacionado ao número de oportunidades por ciclo.

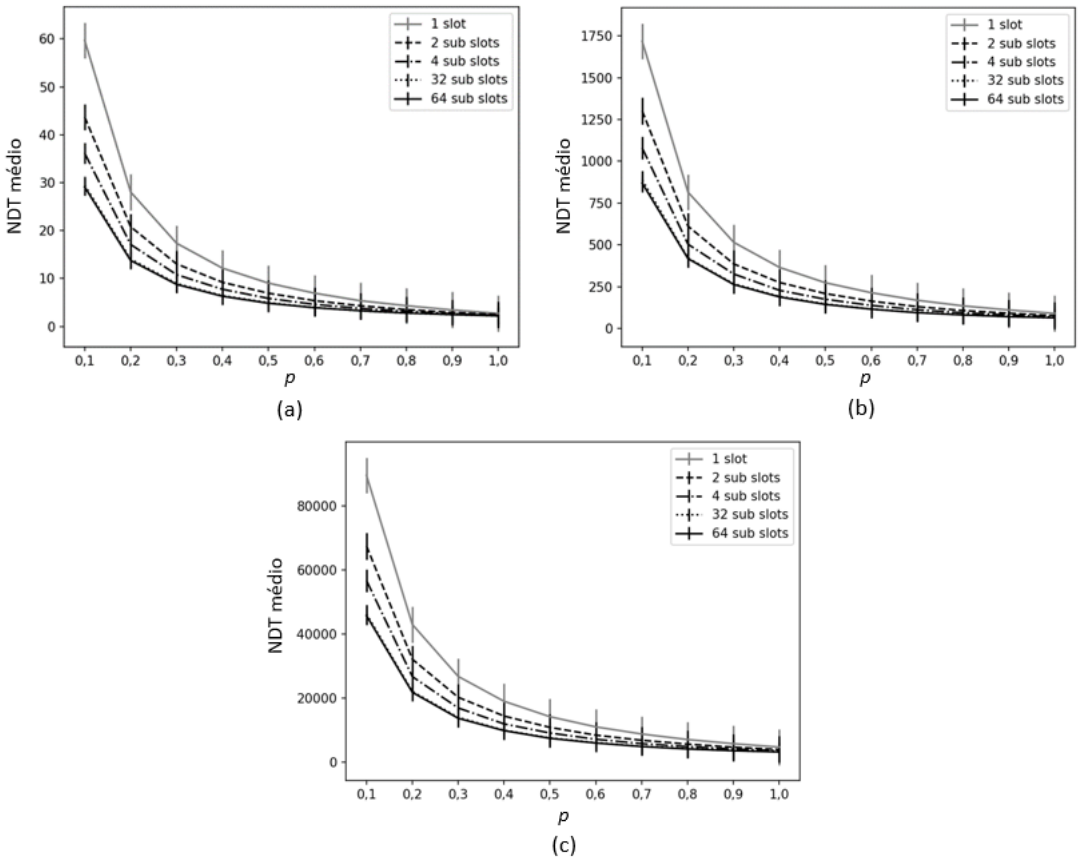

Figura 13 - NDT médio pela probabilidade de sucesso (p) para os *Block Designs* de (a) {7, 3, 1}, (b) {183, 14, 1} e (c) {9507, 98, 1}.

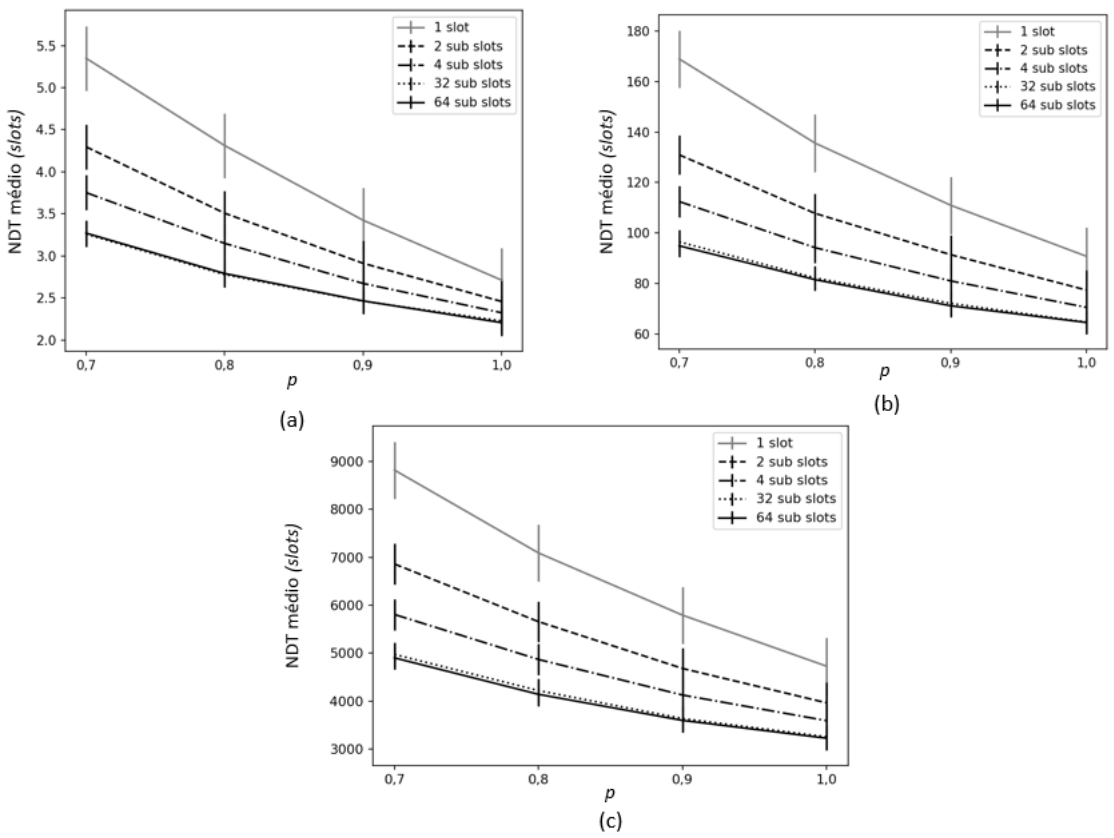

Figura 14 - NDT médio pela probabilidade de sucesso (p) para os Block Designs de (a) {7, 3, 1}, (b) {183, 14, 1} e (c) {9507, 98, 1} com foco em probabilidades (*p*) próximas a 1,0.

Para entender melhor o efeito do número de fatias no modelo, a segunda análise realizada buscou relacionar este parâmetro com o NDT médio para os escalonamentos {7, 3, 1}, {183, 14, 1} e {9507, 98, 1}, como mostrado na Figura 15. Para esta análise foi considerada uma probabilidade de sucesso de 0,1. É possível observar que o comportamento qualitativo das curvas é idêntico para os três escalonamentos estudados, apesar da diferença na escala. Além disso, essa análise confirma o que foi observado na análise anterior de que o valor do NDT médio cai com o aumento do número de *fatias* — aproximadamente metade do valor previsto para 1 fatia quando o número de *fatias* tende ao infinito. Através destas observações, conjecturou-se que, para valores baixos de *p*, o NDT para um dado número de *fatias* pode ser aproximado pela função (3):

$$NDT_f = (f+1) \times \frac{NDT_1}{(2 \times f)} \quad (3)$$

onde *f* é o número de *fatias* e NDT_1 é o valor do NDT médio para uma fatia. Esta função também está plotada nos gráficos da Figura 15 (pontilhada) para comparação.

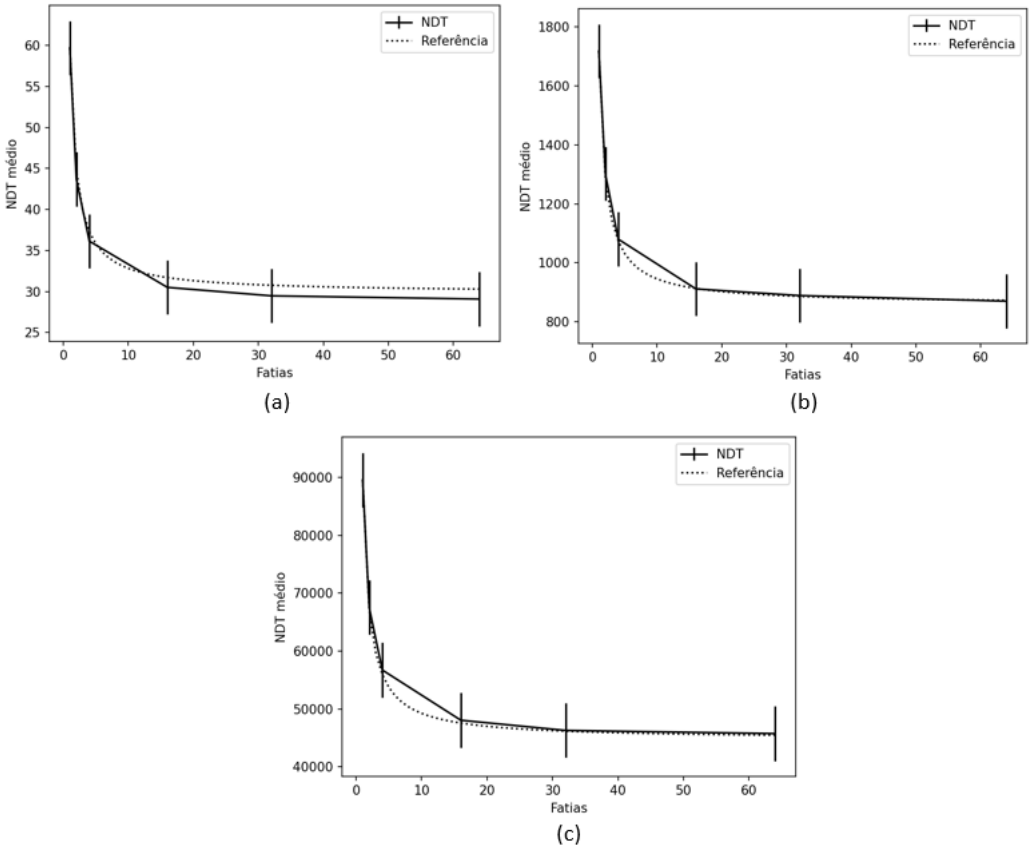

Figura 15 - NDT médio pela quantidade de fatias para os *Block Designs* de (a) {7, 3, 1}, (b) {183, 14, 1} e (c) {9507, 98, 1}.

Nota-se que a aproximação funciona bem, particularmente para escalonamentos mais longos. Este resultado é relevante, pois o modelo aqui proposto é equivalente ao de [3] quando se considera uma única fatia, e são conhecidas na literatura várias expressões fechadas para o NDT médio para diversos tipos de escalonamento baseadas no modelo de [3]. Logo, é possível adaptar estas expressões para o modelo proposto nesta dissertação para os casos em que $p << 1$ através da função (3).

Capítulo 6 - Conclusão

O objetivo principal deste trabalho foi a aprofundamento do estudo do NDT em escalonamentos assíncronos, considerando que não há nenhum tipo de sincronização entre os nós e, por isso, seus *slots* não são iniciados simultaneamente. Para estudar o efeito deste desalinhamento, o trabalho dividiu os *slots* em *subslots*, sendo estes últimos capazes de capturar o desalinhamento e, à medida que seu número aumenta, o cenário se aproxima do caso real (tempo contínuo).

No estudo, foram analisados dois cenários para medir o NDT que consideram a possibilidade de os *slots* nos escalonamentos dos nós vizinhos não terem suas bordas perfeitamente alinhadas: o unidirecional e o bidirecional. No modelo unidirecional, considera-se como sendo uma oportunidade de encontro entre dois nós se ambos estiverem em *subslots* ativos e se o primeiro *subslot* de um *slot* do transmissor coincide com qualquer *subslot* ativo do receptor. Já no modelo bidirecional, considera-se que só há uma *oportunidade de encontro* entre dois nós se, assim como no modelo unidirecional, ambos estiverem em *subslots* ativos (para terem seus rádios ligados simultaneamente) e se um deles estiver no primeiro subslot de seu respectivo slot.

É possível concluir que tanto o modelo unidirecional quanto o bidirecional, ao considerar o desalinhamento de slots para o cálculo do NDT, fornecem uma previsão mais precisa que a dos modelos propostos anteriormente na literatura. Vale ressaltar que os modelos de NDT na literatura não consideram o efeito do atraso de propagação e, embora este parâmetro não esteja explícito nos modelos propostos nesta dissertação, seu efeito é implicitamente capturado, sendo contabilizado no próprio *offset* entre os nós.

Através das simulações realizadas para validação dos modelos, foi possível concluir que, para o modelo unidirecional, o NDT médio aumenta com o aumento do número de *fatias* — aproximadamente 0,5 *slots* em relação ao valor previsto para 1 fatia quando o número de *fatias* tende ao infinito. Já para o modelo bidirecional, independentemente do escalonamento analisado, para um valor de probabilidade de sucesso $p << 1$ e um número de *subslots* por *slot* tendendo ao infinito, o NDT tende a cair pela metade em relação ao caso de uma fatia. Para *p* próximo de um, o NDT também cai, mas não de forma tão acentuada.

6.1 Trabalhos Futuros

No futuro, pretende-se investigar melhor a relação de queda do NDT com o aumento do número de fatias para o caso de p próximo a 1. O objetivo final é obter expressões fechadas para o NDT médio de diversos tipos de escalonamento, talvez baseados nas expressões já disponíveis para o caso de 1 fatia.

Bibliografia

[1] Zeadally, Sherali et al. Design architectures for energy harvesting in the Internet of Things. Renewable and Sustainable Energy Reviews, v. 128, p. 109901, 2020.

[2] G. Anastasi, M. Conti, M. D. Francesco e A. Passarella, “Energy conservation in wireless sensor networks: a survey,” *Ad Hoc Networks,* p. 537–568, 2009.

[3] R. C. CARRANO, “A comprehensive analysis on the use of schedule-based asynchronous duty cycling in wireless sensor networks,” *Ad Hoc Networks,* pp. 142-164, 2014.

[4] J. Jiang, Y. Tseng, C. Hsu e T. Lai, “Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc networks,” Mobile Networks and Applications, p. 169–181, 2005.

[5] Amin, Farhan, et al. "An Overview of Medium Access Control and Radio Duty Cycling Protocols for Internet of Things." Electronics 11.23 (2022): 3873.

[6] Y. Sun, O. Gurewitz e D. Johnson, “RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks,” em *Proceedings of the 6th ACM International Conference on Embedded Networked Sensor Systems (SenSys ’08)*, Nova Iorque, 2008.

[7] Gupta, Sunita, Sakar Gupta, and Dinesh Goyal. "Wireless sensor network in IoT and performance optimization." Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science) 15.1 (2022): 14-22.

[8] Tripathi, Yogesh, Arun Prakash, and Rajeev Tripathi. "A novel slot scheduling technique for duty-cycle based data transmission for wireless sensor network." Digital Communications and Networks 8.3 (2022): 351-358.

[9] R. ZHENG, J. C. HOU e L. SHA, “Asynchronous wakeup for ad hoc networks,” em *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, 2003.

[10] A. SARAIVA, “Reduzindo a latência de comunicação em múltiplos saltos dos modelos de duty cycle assíncrono baseados em schedule através de sincronização de baixa resolução,” em *Simpósio Brasileiro de Redes de Computadores e modelos Distribuídos*, 2017.

[11] Phan, Linh-An, et al. "Performance analysis of time synchronization protocols in wireless sensor networks." Sensors 19.13 (2019): 3020.

[12] W. PAK, “W-MAC: supporting ultra low duty cycle in wireless sensor networks,” em *IEEE GLOBECOM*, 2008.

[13] R. C. CARRANO, “Neighbor discovery time in schedule-based asynchronous duty cycling,” *IFIP Wireless Days. IEEE*, pp. 1-3, 2012.

[14] Subramanyam, R., Bala, G. J., Perattur, N., & Kanaga, E. G. M. (2021). Energy Efficient MAC with Variable Duty Cycle for Wireless Sensor Networks. International Journal of Electronics, 1-24.

[15] Sun, Y., Gurewitz, O., & Johnson, D. B. (2008, November). RI-MAC: a receiver-initiated asynchronous , Cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In Proceedings of the 6th ACM conference on Embedded network sensor systems (pp. 1-14).

[16] Buettner, M., Yee, G. V., Anderson, E., & Han, R. (2006, October). X-MAC: a short preamble MAC protocol for Duty-Cycled wireless sensor networks. In Proceedings of the 4th international conference on Embedded networked sensor systems (pp. 307-320).

[17] Carrano, R. C., Passos, D., Magalhaes, L. C., & Albuquerque, C. V. (2013). Survey and taxonomy of Duty cycling mechanisms in wireless sensor networks. IEEE Communications Surveys & Tutorials, 16(1), 181-194.

[18] Jang, B., Lim, J. B., & Sichitiu, M. L. (2013). An asynchronous scheduled MAC protocol for wireless sensor networks. Computer Networks, 57(1), 85-98.

[19] Gudnavar, A., & Manjanaik, N. (2021). A Survey on Energy-Efficient MAC Protocols for Wireless Sensor Networks. In Smart Sensors Measurements and Instrumentation (pp. 177-188). Springer, Singapore.

[20] Pande, H., Kharat, M. U., Saharan, K., & Mukhopadhyay, D. (2013, October). Various Ways to Implement Energy Efficient WiseMAC Protocol for Wireless Sensor Network. In 2013 IEEE International Conference on Systems, Man, and Cybernetics (pp. 22-25). IEEE.

[21] Hurni, P., Braun, T., & Anwander, M. (2010). Evaluation of WiseMAC and extensions on wireless sensor nodes. Telecommunication Systems, 43(1), 49-58.

[22] El-Hoiydi, A., & Decotignie, J. D. (2004, June). WiseMAC: an ultra-low power MAC protocol for the downlink of infrastructure wireless sensor networks. In Proceedings. ISCC 2004. Ninth International Symposium on Computers and Communications (IEEE Cat. No. 04TH8769) (Vol. 1, pp. 244-251). IEEE.

[23] Zhang, D. G., Zhou, S., & Tang, Y. M. (2018). A low Duty Cycle efficient MAC protocol based on self-adaption and predictive strategy. Mobile Networks and Applications, 23(4), 828-839.

[24] Singh, P., & Chen, Y. C. (2020). Energy efficient broadcast protocols for asynchronous Duty-Cycled wireless sensor networks. Wireless Networks, 26(2), 1373-1388.

[25] Tang, L., Sun, Y., Gurewitz, O., & Johnson, D. B. (2011, April). PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks. In 2011 Proceedings IEEE INFOCOM (pp. 1305-1313). IEEE.

[26] Khandelwal, A., & Jain, Y. K. (2019). Computational analysis of clustering techniques for the efficient cluster head selection. International Journal of Advanced Technology and Engineering Exploration, 6(60), 248-249.

[27] Jang, B., Lim, J. B., & Sichitiu, M. L. (2008, September). AS-MAC: An asynchronous scheduled MAC protocol for wireless sensor networks. In 2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (pp. 434-441). IEEE.

[28] Liao, W. H., & Wang, H. H. (2008). An asynchronous MAC protocol for wireless sensor networks. Journal of Network and Computer Applications, 31(4), 807-820.

[29] Lee, H., Hong, J., Yang, S., Jang, I., & Yoon, H. (2010). A pseudo-random asynchronous Duty Cycle MAC protocol in wireless sensor networks. IEEE Communications Letters, 14(2), 136-138.

[30] G. Pottie, W. Kaiser, Wireless integrated network sensors, Communication of ACM 43 (5) (2000) 51–58.

[31] Yun, S. Y., Yi, Y., & Shin, J. (2012, November). Optimal CSMA A survey. In *2012 IEEE international conference on communication systems (ICCS)* (pp. 199-204). IEEE.

[32] J.-R. Jiang, Y.-C. Tseng, C.-S. Hsu, T.-H. Lai, Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc networks, Mobile Networks Appl. 10 (2005) 169–181.

[33] R. Zheng, J.C. Hou, L. Sha, Asynchronous wakeup for ad hoc networks, in: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc ’03), ACM, New York, NY, USA, 2003, pp. 35–45.

[34] P. Dutta, D. Culler, Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications, in: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys ’08), ACM, New York, NY, USA, 2008, pp. 71–84.

Anexo A – Código Python modelo unidirecional

```
import sys
from random import random
import math
import argparse

def progressbar(it, prefix="", size=60, out=sys.stdout): # Python3.3+
    count = len(it)
    def show(j):
        x = int(size*j/count)
        print("{}[{}{}] {}/{}".format(prefix, "#"*x, "."*(size-x), j, count),
              end='\r', file=out, flush=True)
    show(0)
    for i, item in enumerate(it):
        yield item
        show(i+1)
    print("\n", flush=True, file=out)

def parseArgs():

    parser = argparse.ArgumentParser(description='Calcula estatisticas de NDT para escalonamentos potencialmente assimetricos com base em simulacoes numericas.')
    parser.add_argument('schAString', metavar='schA', nargs=1,
                        help='Escalonamento do primeiro sensor no formato va,sa1[,sa2[,sa2...]. va eh o tamanho do ciclo, sa1,sa2,... sao os indices dos slots ativos')
    parser.add_argument('schBString', metavar='schB', nargs=1,
                        help='Escalonamento do segundo sensor no formato vb,sb1[,sb2[,sb2...]. vb eh o tamanho do ciclo, sb1,sb2,... sao os indices dos slots ativos')
    parser.add_argument('p', metavar='p', type=float, nargs=1,
                        help='probabilidade de sucesso do enlace')
    parser.add_argument('reps', metavar='reps', type=int, nargs=1,
                        help='numero de repeticoes da simulacao')
```

```
        parser.add_argument('fatias', metavar='fatias', type=int, nargs='?', default=1,
                            help='numero de fatias em que cada slot eh subdivido')
        parser.add_argument('--amostras', '-a', metavar='arquivo', default='',
                            help='nome de arquivo a ser gerado com as amostras de NDT obtidas.
Se \'-\' for especificado, amostras sao impressas na saida padrao. Se nao for especificado,
amostras nao sao impressas.')

        args = parser.parse_args()

        schA = [int(i) for i in args.schAString[0].split(",")]
        schB = [int(i) for i in args.schBString[0].split(",")]
        p = args.p[0]
        reps = args.reps[0]
        fatias = args.fatias
        samplesFile = args.amostras

        va = schA[0] * fatias
        vb = schB[0] * fatias
        schA = [i*fatias + j for i in schA[1:] for j in range(fatias)]
        schB = [i*fatias + j for i in schB[1:] for j in range(fatias)]

        params = {

            "p": p,
            "va": va,
            "vb": vb,
            "schA": schA,
            "schB": schB,
            "reps": reps,
            "fatias" : fatias,
            "samplesFile" : samplesFile
        }
```

```
    return params

# Dado o slot atual, retorna o indice
# do proximo slot ativo em sch
def nextActiveSlot(slot, sch):

    for i in range(len(sch)):

        if slot <= sch[i]:
            return i

    return 0

# Retorna a diferenca entre dois slots, considerando
# o tamanho do ciclo do sensor
def timeUntilSlot(currentSlot, targetSlot, v):

    diff = targetSlot - currentSlot
    if diff < 0:
        diff = v + diff
    return diff

def simNDT(va, schA, vb, schB, p):

    # Contar o tempo de simulacao em t
    t = 0

    # Manter variavel com o tempo desde a ultima oportunidade de encontro.
    # Sera usado para detectar falta de fechamento para rotacao
    timeSinceLastOpportunity = 0

    # Para o mesmo proposito, calcular o MMC entre os comprimentos dos
    # escalonamentos. Este valor representa o tempo maximo ate que uma
```

```
            # oportunidade de encontro ocorra (assumindo fechamento para rotacao).
            maxIntervalUntilOpportunity = math.lcm(va, vb)

            # Sortear slots iniciais aleatorios para cada sensor
            slotA = math.floor(random() * va)
            slotB = math.floor(random() * vb)

            # Descobrir qual o proximo slot ativo para cada sensor
            nextActiveA = nextActiveSlot(slotA, schA)
            nextActiveB = nextActiveSlot(slotB, schB)

            # Repetir ate o encontro
            while True:

               # Descobrir quanto tempo falta ate o proximo slot ativo de
               # um dos dois nos
                  timeUntilNextActive = min(timeUntilSlot(slotA, schA[nextActiveA], va),
timeUntilSlot(slotB, schB[nextActiveB], vb))

               # Avancar o tempo de simulacao
               t = t + timeUntilNextActive
               slotA = (slotA + timeUntilNextActive) % va
               slotB = (slotB + timeUntilNextActive) % vb

               # Para cada sensor, verificar se o proximo slot é ligado
               if slotA == schA[nextActiveA]:

                  # Sim.
                  activeSlotA = True

                  # Avancar indice do proximo slot ativo.
                  nextActiveA = (nextActiveA + 1) % len(schA)
               else:
```

```
            activeSlotA = False

        if slotB == schB[nextActiveB]:

            # Sim.
            activeSlotB = True

            # Avancar indice do proximo slot ativo.
            nextActiveB = (nextActiveB + 1) % len(schB)
        else:
            activeSlotB = False

        # Verificar se houve uma oportunidade de encontro (i.e.,
        # ambos os nos com radio ligado)
        if activeSlotA == True and activeSlotB == True:

            # Sim. Sortear valor aleatorio para determinar se
            # comunicacao foi bem sucedida.
            if random() < p:

                # Sim. Houve comunicacao. Fim da simulacao.
                return t

            # Atualizar o tempo desde a ultima oportunidade
            timeSinceLastOpportunity = 0

        else:

            # Atualizar o tempo desde a ultima oportunidade
                    timeSinceLastOpportunity = timeSinceLastOpportunity +
timeUntilNextActive

            # Verificar se o tempo eh maior que o maximo teorico
```

```
                if timeSinceLastOpportunity >= maxIntervalUntilOpportunity:
                    print("Escalonamentos nao tem fechamento para rotacao: falha para offset
{}".format(slotA - slotB))
                    sys.exit(1)


    params = parseArgs()
    samples = [0] * params["reps"]

    print("")
    for i in progressbar(range(params["reps"]), "Processando: ", 40):
            samples[i] = simNDT(params["va"], params["schA"], params["vb"],
params["schB"], params["p"])

    print("Estatisticas:")
    print("\t- NDT minimo: {}".format(min(samples) / float(params["fatias"])))
    print("\t- NDT maximo: {}".format(max(samples) / float(params["fatias"])))
    print("\t- NDT medio: {}".format((sum(samples) / float(params["reps"]) /
float(params["fatias"]))))

    print("")

    if params["samplesFile"] != "":
        if params["samplesFile"] == "-":
            print("##############")
            print("Amostras:")
        else:
            f = open(params["samplesFile"], "w")
            sys.stdout = f

        for i in samples:
            print(i)
```

Anexo B – Código Python modelo bidirecional

```
import sys
from random import random
import math
import argparse

def progressbar(it, prefix="", size=60, out=sys.stdout): # Python3.3+
    count = len(it)
    def show(j):
        x = int(size*j/count)
        print("{}[{}{}] {}/{}".format(prefix, "#"*x, "."*(size-x), j, count),
              end='\r', file=out, flush=True)
    show(0)
    for i, item in enumerate(it):
        yield item
        show(i+1)
    print("\n", flush=True, file=out)

def parseArgs():

    parser = argparse.ArgumentParser(description='Calcula estatisticas de NDT para escalonamentos potencialmente assimetricos com base em simulacoes numericas.')
    parser.add_argument('schAString', metavar='schA', nargs=1,
                        help='Escalonamento do primeiro sensor no formato va,sa1[,sa2[,sa2...]. va eh o tamanho do ciclo, sa1,sa2,... sao os indices dos slots ativos')
    parser.add_argument('schBString', metavar='schB', nargs=1,
                        help='Escalonamento do segundo sensor no formato vb,sb1[,sb2[,sb2...]. vb eh o tamanho do ciclo, sb1,sb2,... sao os indices dos slots ativos')
    parser.add_argument('p', metavar='p', type=float, nargs=1,
                        help='probabilidade de sucesso do enlace')
    parser.add_argument('reps', metavar='reps', type=int, nargs=1,
                        help='numero de repeticoes da simulacao')
    parser.add_argument('fatias', metavar='fatias', type=int, nargs='?', default=1,
                        help='numero de fatias em que cada slot eh subdivido')
```

```
        parser.add_argument('--amostras', '-a', metavar='arquivo', default='',
                            help='nome de arquivo a ser gerado com as amostras de NDT obtidas. Se \'-\' for especificado, amostras sao impressas na saida padrao. Se nao for especificado, amostras nao sao impressas.')

        args = parser.parse_args()

        schA = [int(i) for i in args.schAString[0].split(",")]
        schB = [int(i) for i in args.schBString[0].split(",")]
        p = args.p[0]
        reps = args.reps[0]
        fatias = args.fatias
        samplesFile = args.amostras
        schA_ativos = [i*fatias for i in schA[1:]]
        schB_ativos = [i*fatias for i in schB[1:]]

        va = schA[0] * fatias
        vb = schB[0] * fatias
        schA = [i*fatias + j for i in schA[1:] for j in range(fatias)]
        schB = [i*fatias + j for i in schB[1:] for j in range(fatias)]

        params = {

            "p": p,
            "va": va,
            "vb": vb,
            "schA": schA,
            "schB": schB,
            "reps": reps,
            "fatias" : fatias,
            "schA_ativos": schA_ativos,
            "schB_ativos": schB_ativos,
            "samplesFile" : samplesFile
```

```
    }

    return params

# Dado o slot atual, retorna o indice
# do proximo slot ativo em sch
def nextActiveSlot(slot, sch):

    for i in range(len(sch)):

        if slot <= sch[i]:
            return i

    return 0

# Retorna a diferenca entre dois slots, considerando
# o tamanho do ciclo do sensor
def timeUntilSlot(currentSlot, targetSlot, v):

    diff = targetSlot - currentSlot
    if diff < 0:
        diff = v + diff
    return diff

def simNDT(va, schA, vb, schB, p, schA_ativos, schB_ativos):

    # Contar o tempo de simulacao em t
    t = 0

    # Manter variavel com o tempo desde a ultima oportunidade de encontro.
    # Sera usado para detectar falta de fechamento para rotacao
    timeSinceLastOpportunity = 0
```

```
        # Para o mesmo proposito, calcular o MMC entre os comprimentos dos
        # escalonamentos. Este valor representa o tempo maximo ate que uma
        # oportunidade de encontro ocorra (assumindo fechamento para rotacao).
        maxIntervalUntilOpportunity = math.lcm(va, vb)

        # Sortear slots iniciais aleatorios para cada sensor
        slotA = math.floor(random() * va)
        slotB = math.floor(random() * vb)

        # Descobrir qual o proximo slot ativo para cada sensor
        nextActiveA = nextActiveSlot(slotA, schA)
        nextActiveB = nextActiveSlot(slotB, schB)
        #print (schA)
        #print(schA_ativos)

        # Repetir ate o encontro
        while True:

            # Descobrir quanto tempo falta ate o proximo slot ativo de
            # um dos dois nos
                timeUntilNextActive = min(timeUntilSlot(slotA, schA[nextActiveA], va),
timeUntilSlot(slotB, schB[nextActiveB], vb))

            # Avancar o tempo de simulacao
            t = t + timeUntilNextActive
            slotA = (slotA + timeUntilNextActive) % va
            slotB = (slotB + timeUntilNextActive) % vb

            # Para cada sensor, verificar se o proximo slot é ligado
            if slotA == schA[nextActiveA]:

                # Sim.
                activeSlotA = True
```

```
    # Avancar indice do proximo slot ativo.
    nextActiveA = (nextActiveA + 1) % len(schA)
else:
    activeSlotA = False

if slotB == schB[nextActiveB]:

    # Sim.
    activeSlotB = True

    # Avancar indice do proximo slot ativo.
    nextActiveB = (nextActiveB + 1) % len(schB)
else:
    activeSlotB = False



    # Verificar se houve uma oportunidade de encontro (i.e.,
    # ambos os nos com radio ligado)
if activeSlotA == True and activeSlotB == True:
    if schA[nextActiveA] in schA_ativos or schB[nextActiveB] in schB_ativos:
        # Sim. Sortear valor aleatorio para determinar se
        # comunicacao foi bem sucedida.
        if random() < p:

            # Sim. Houve comunicacao. Fim da simulacao.
            return t

        # Atualizar o tempo desde a ultima oportunidade
        timeSinceLastOpportunity = 0

else:
```

```
                # Atualizar o tempo desde a ultima oportunidade
                        timeSinceLastOpportunity = timeSinceLastOpportunity + timeUntilNextActive

                # Verificar se o tempo eh maior que o maximo teorico
                if timeSinceLastOpportunity >= maxIntervalUntilOpportunity:
                    print("Escalonamentos nao tem fechamento para rotacao: falha para offset {}".format(slotA - slotB))
                    sys.exit(1)

        params = parseArgs()
        samples = [0] * params["reps"]

        print("")
        for i in progressbar(range(params["reps"]), "Processando: ", 40):
            samples[i] = simNDT(params["va"], params["schA"], params["vb"], params["schB"], params["p"], params["schA_ativos"], params["schB_ativos"])

        print("Estatisticas:")
        print("\t- NDT minimo: {}".format(min(samples) / float(params["fatias"])))
        print("\t- NDT maximo: {}".format(max(samples) / float(params["fatias"])))
        print("\t- NDT medio: {}".format((sum(samples) / float(params["reps"]) / float(params["fatias"]))))

        print("")

        if params["samplesFile"] != "":
            if params["samplesFile"] == "-":
                print("##############")
                print("Amostras:")
            else:
                f = open(params["samplesFile"], "w")
                sys.stdout = f
```

```
for i in samples:
    print(i)
```