Dissertação de Mestrado

Inatel
Instituto Nacional de Telecomunicações

Estudo da Codificação de Rede e Análise do seu Desempenho com Fonte de Tráfego HTTP

DANIEL CARLI COLARES E SILVA

SETEMBRO/ 2011

Silva, Daniel Carli Colares e
S586e
Estudo da Codificação de Rede e Análise do seu Desempenho com Fonte de Tráfego HTTP / Daniel Carli Colares e Silva. – Santa Rita do Sapucaí, 2011.
96 p.

Orientador: Dr. Carlos Roberto dos Santos
Dissertação de Mestrado – Engenharia de Telecomunicações – Instituto Nacional de Telecomunicações – INATEL.
Inclui bibliografia.

1. Desempenho 2. Codificação de Rede 3. Store-and-Forward 4. HTTP. I. Santos, Carlos Roberto dos. II. Instituto Nacional de Telecomunicações – INATEL. III. Título.

CDU 621.39

INSTITUTO NACIONAL DE TELECOMUNICAÇÕES – INATEL

MESTRADO EM TELECOMUNICAÇÕES

ESTUDO DA CODIFICAÇÃO DE REDE E ANÁLISE DO SEU DESEMPENHO COM FONTE DE TRÁFEGO HTTP

DANIEL CARLI COLARES E SILVA

Dissertação apresentada ao Instituto Nacional de Telecomunicações INATEL, como requisito parcial para obtenção do título de Mestre em Telecomunicações.

ORIENTADOR: PROF. Dr. Carlos Roberto dos Santos

SANTA RITA DO SAPUCAÍ – MG

2011

DANIEL CARLI COLARES E SILVA

ESTUDO DA CODIFICAÇÃO DE REDE E ANÁLISE DO SEU DESEMPENHO COM FONTE DE TRÁFEGO HTTP

Santa Rita do Sapucaí, 05 de setembro de 2011

Membros da banca avaliadora

_____

Prof. Dr. Carlos Roberto dos Santos
(Orientador) - INATEL

_____

Prof. Dr. Antônio Marcos Alberti - INATEL

_____

Prof. Dr. Nilton Alves Maia - UNIMONTES

_____

Prof. Dr. Luciano Leonel Mendes
Coordenador do Curso de Mestrado – INATEL

AGRADECIMENTOS

Primeiramente a Deus, por me conceder momentos de descoberta, por perceber que existem pessoas dispostas a dividir, por me dar condições de ensinar e de ser ensinado, por perceber o quanto se pode ser útil usando a inteligência e a mente para a busca de conhecimento e sabedoria, o quanto se podem realizar ações de respeito à vida e o quanto a vida é uma dádiva, divina e abençoada.

Ao meu orientador, Carlos Roberto dos Santos, pela sua contribuição em meu trabalho.

Aos meus pais, pelos conselhos, pelas palavras, pelo esforço, enfim, por viabilizarem esse sonho.

Ao Inatel, essa excelente instituição de ensino.

DEDICATÓRIAS

Dedico este trabalho,

Aos meus pais, os pilares fundamentais para minha formação, sem eles, não seria possível a realização desse sonho.

A minha filha Alanna Carli Colares Xavier, que chegou a minha vida durante este trabalho, e mudou-a para melhor.

RESUMO

O volume de dados trafegados na internet cresce de forma exponencial. Do tráfego gerado, o tráfego *web* é estimado como sendo mais de 70%. Atualmente, são pesquisadas várias tecnologias para um melhor aproveitamento da largura de banda para as transmissões das aplicações, e a técnica de codificação de rede é uma delas. Neste trabalho são apresentados estudos sobre algumas técnicas de codificação de rede e seus benefícios, bem como uma análise de desempenho com fonte de tráfego HTTP utilizando os métodos *store-code-forward* e *store-and-forward*. Esta análise de desempenho é feita em um nó de gargalo em uma rede de topologia borboleta considerando os dois modos de *download* HTTP/1.0-*busrt mode transfer* e o HTTP/1.1-*persistent mode transfer*. Nos cenários investigados, tanto teoricamente quanto via simulação, a codificação de rede faz um melhor aproveitamento da taxa de transmissão de um enlace comparado ao método tradicional *store-and-forward*.

Palavras-Chave: desempenho, codificação de rede, *store-and-forward*, HTTP.

Abstract

The volume of data traffic on the Internet grows exponentially. The web traffic is estimated to be over 70% of traffic generated on the Internet. Currently, several technologies are being investigated for a better utilization of bandwidth for the transmission of applications, and network coding technique is one. This work presents some studies on network coding techniques and their benefits, as well as a performance analysis with traffic source HTTP using the methods store-and-forward. This performance analysis is done at a topology bottleneck node in a butterfly net considering two ways to download, HTTP/1.0-busrt transfer mode, and HTTP/1.1-persistent transfer mode. In the scenarios investigated both theoretically and via simulation, the network coding makes better use of the transmission rate of a linkage to compared store-and-forward method traditional.

Keywords: performance, network coding, store-and-forward, HTTP.

LISTA DE FIGURAS

LISTA DE TABELAS

Lista de Abreviaturas e Siglas

COPE -	*Opportunistic Code*
FTP -	*File Transfer Protocol*
IP -	*Internet Protocol*
ISP -	*Internet Service Provider*
HTTP -	*Hypertext Transfer Protocol*
Max-Flow Min-Cut -	*Maximum Flow Minimum Cut*
MTU -	*Maximum Transmission Unit*
OSI -	*Open Systems Interconnection*
PDF -	*Probability Density Function*
PPP -	*Point-to-Point Protocol*
P2P -	*Peer-to-peer*
SMTP -	*Simple Mail Transfer Protocol*
TCP -	*Transmission Control Protocol*
TDMA -	*Time Division Multiple Access*

Sumário

1 Introdução

1.1 Motivação e Escopo do Trabalho

Com o advento das tecnologias de informação e comunicação, o mundo passou a se comunicar utilizando meios de transmissão de dados oriundos de diferentes formatos e mídias. Observa-se que empresas, instituições de ensino, a sociedade em geral, usam os serviços de redes e de comunicação de dados para resolver os mais diversificados problemas, o que tem provocado o aumento de forma exponencial no volume de dados trafegados na rede. De acordo com Fragouli e Soljaninz na referência [1], um dos principais problemas nas redes de comunicação é a crescente demanda de largura de banda para as transmissões das aplicações. Dessa forma, faz-se relevante à investigação de teorias e tecnologias para inovar o processo de utilização das redes de comunicação, notadamente a otimização da largura de banda

Para contornar este problema de largura de banda, surge uma nova proposta denominada de codificação de rede, que utiliza o processamento computacional barato para aumentar o desempenho da rede.

Nesse contexto, faz-se mister o estudo de redes codificadas visando, posteriormente, a inserção de um novo serviço no mercado, que venha atender as demandas de transmissão de dados no planeta. Enfatiza-se, assim, que a codificação de rede está relacionada às técnicas para melhoria do desempenho da rede. A partir desse princípio, torna-se necessário mostrar a relevância de realizar um estudo detalhado sobre as teorias e algoritmos que permeiam a codificação de rede.

Assim, este trabalho tem como objetivo realizar um estudo teórico das redes codificadas, buscando elucidar seus conceitos, características, benefícios e limitações, para o processo de transmissão de dados. Como resultado deste estudo teórico, foi implementado um algoritmo que simula a codificação de rede em tráfego HTTP. Tal

estudo ficará disponível para que trabalhos futuros de implementação da mesma possam ser executados.

Este trabalho está assim estruturado: no Capítulo 1 são apresentados alguns conceitos básicos, histórico, áreas de atuação e desafios da codificação de rede. No Capítulo 2 são apresentados os benefícios da codificação de rede, tais como a maximização da vazão, minimização de energia por *bit*, minimização do atraso e análise da utilização da codificação de rede sobre algumas topologias. No Capítulo 3 são apresentados conceitos fundamentais de codificação de rede linear. No Capítulo 4, são apresentadas algumas possibilidades na aplicação da codificação de rede, tais como: *download* de arquivo; vídeo sob demanda; armazenamento distribuído, redes *wireless*, dentre outros. No Capítulo 5 apresenta-se um algoritmo de codificação de rede e sua utilização na simulação para tráfego HTTP, o que permite avaliar o seu desempenho. E por fim, no Capítulo 6, as conclusões deste trabalho.

1.2 Perspectiva Histórica

De acordo Fragouli e Soljaninz na referência [1], a premissa fundamental inerente por trás da operação de todas as redes de comunicação hoje reside na forma como a informação é tratada. Tanto para redes de comutação de pacotes, quanto para sinais de uma rede de telefonia, originárias de fontes diferentes, cada fluxo de informação é mantido e tratado em separado dos demais. São baseados neste princípio de tratamento da informação, o roteamento, comutação, armazenamento de dados e controle de erro. Geralmente, todas as funções de uma rede operam desta forma.

Conforme a referência [1], a codificação de rede é entendida como uma técnica inovadora com relação ao tratamento da informação. Baseia-se em algoritmos nos quais o objetivo é melhorar a vazão e o desempenho da rede. Está técnica surgiu em meados do ano 2000, de acordo com as referências [1] e [2], no intuito de se tornar uma importante tecnologia para as redes. A ideia da técnica de codificação de rede surgiu para que sistemas de comunicação, tais como telefonia, Internet e redes móveis, aproveitem melhor a capacidade do canal de comunicação entre o transmissor e o receptor.

De acordo com Cai e*t al*. em [2], ela foi introduzida pela primeira vez por Yeung e Zhang no uso de redes de comunicação via satélite [3], e totalmente desenvolvida por Ahlswede e*t al*. conforme referência [4], onde neste último o termo codificação de rede foi estabelecido e foram demonstradas vantagens sobre a técnica *store-and-forward*.

1.3 Método *Store-and-Forward*

Em uma rede de comunicação ponto-a-ponto, os dados são transmitidos a partir do nó de origem, passando por nós intermediários até chegar ao nó de destino. Por cada nó em que os dados trafeguem, eles são armazenados e depois encaminhados. Essa técnica é conhecida como *store-and-forward* [2]. A informação irá percorrer os canais de comunicação entre os nós utilizando à capacidade de um determinado canal em um dado momento. Para que esta transmissão aconteça, devem ser estabelecidos alguns requisitos, e suas implementações de forma eficiente. Como exemplo desse requisito, cita-se que há recursos suficientes na rede.

No método *store-and-forward*, os pacotes de dados recebidos de um enlace de entrada de um nó intermediário são armazenados e uma cópia é encaminhada para o próximo nó através de um enlace de saída. Quando ocorrer que o nó intermediário estiver no caminho para múltiplos destinos, envia-se uma cópia dos pacotes de dados em pelo menos um enlace de saída que leva ao destino. Neste método, conforme [2], na rede de dados não há necessidade de tratamento de dados nos nós intermediários, exceto para a replicação de dados.

1.4 Método *Store-Code-Forward*

Em uma rede de computadores, um roteador simplesmente cria rotas e encaminha pacotes, normalmente utilizando o método *store-and-forward*. Recentemente, foi introduzido um novo conceito denominado de codificação de rede, que utiliza o método s*tore-code-forward*. Conforme [4], cada mensagem em um enlace de saída deve ser uma cópia de uma mensagem que chegou mais cedo em um enlace de entrada. A codificação de rede permite que cada nó em uma rede, além de realizar as funções de roteamento e

encaminhamento, realize algum processo computacional. Entende-se neste trabalho que o processo computacional seja a codificação dos dados entrantes em um determinado nó. Portanto, em codificação de rede, cada mensagem enviada pelo enlace de saída de um nó pode ser alguma função ou "mistura" de mensagens que chegaram mais cedo em enlaces de entrada daquele nó, conforme ilustrado na Figura 1.1. Assim, a codificação de rede é definida como a transmissão, a mistura (ou codificação), e re-mistura (ou re-codificação) de mensagens que chegam aos nós dentro da rede, de tal forma que as mensagens transmitidas podem ser descombinadas (ou decodificadas) em seus destinos finais.

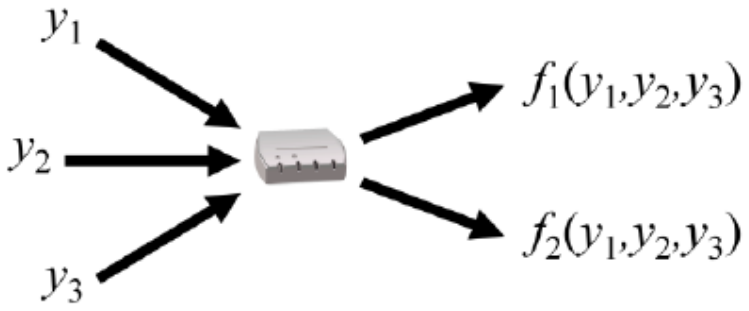


Figura 1.1: Codificação de rede: Nós da rede implementam funções computacionais (f_i) para as mensagens (y_i) que chegam ao nó. Ex.: Adição binária [5].

Com a codificação de rede, uma importante modificação deverá ser feita nas redes de comunicação. Por exemplo, na camada de rede os nós poderão enviar, mas também codificar o fluxo de informações independentes de cada interface de entrada. No processo de codificação os nós intermediários podem realizar a operação ou-exclusivo - XOR entre fluxos ou pacotes de dados independentes, resultando em fluxos combinados independentes uns dos outros. Ao contrário do usual, esses fluxos não são necessariamente mantidos em separados em uma transmissão de dados. Conforme [1], a combinação dos fluxos de dados independentes possui a finalidade de otimizar a utilização da largura de banda. Esta nova abordagem na transmissão de informações pretende atingir várias áreas como compartilhamento de recursos, monitoramento de rede e segurança, controle de fluxo eficiente, no intuito de melhorar o desempenho destas áreas nas redes de comunicações.

1.5 Área de Atuação da Codificação de Rede

A área de atuação da codificação de rede é muito extensa, envolvendo muitos outros temas como teoria da codificação, algoritmos, segurança, armazenamento distribuído, teoria da informação, redes *ad hoc*, monitoramento de redes, dentre outros. A Figura 1.2 ilustra a relação da codificação de rede com vários outros assuntos.

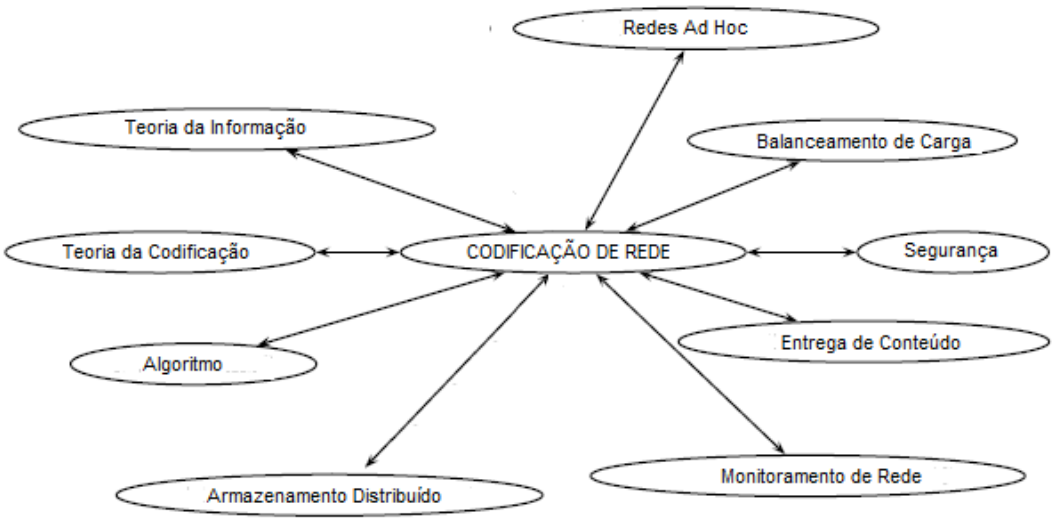


Figura 1.2: Conexão entre codificação de rede com outras áreas.

1.6 Desafios com a Utilização da Codificação de Rede

Por ser uma forma inovadora de transmitir os fluxos de dados, a codificação de rede possui alguns desafios, tais como complexidade, integração com a infra-estrutura existente, segurança.

A complexidade é resultado das funcionalidades adicionais que os nós da rede suportam para a implementação da codificação de rede, tais como requisitos adicionais de memória, realizar operações lógicas do tipo XOR, manter suas próprias informações armazenadas, e em seguida, resolver um sistema linear de equações. Estas funcionalidades adicionais são suportadas pelos nós de codificação.

Por se tratar de uma tecnologia emergente pode ser integrada na rede existente ou coexistir em redes independentes. A integração com a infra-estrutura existente é de suma importância, pois as redes de comunicação precisam incorporar tecnologias emergentes para continuar evoluindo. Este tema pode ser considerado como relevante em uma linha de pesquisa, abordando aspectos de como ser implementada a codificação de rede com todas as suas vantagens, sem ocasionar mudanças drásticas na arquitetura da rede existente.

De acordo com referência [1], os mecanismos de segurança em vigor são projetados em torno do pressuposto de que as únicas entidades elegíves para adulterar os dados são a origem e o destino. Atualmente, os mecanismos de segurança como redes para transações bancárias, implementam este conceito. O conceito de segurança utilizando codificação de rede exige que os nós intermediários executem operações nos pacotes de dados sem afetar a autenticidade, para garantir o mesmo nível de segurança.

Conforme Chou e*t al*. na referência [5], as técnicas de codificação de rede podem ser aplicáveis nas redes existentes, tendo como exemplo principal, a Internet, tanto na camada IP, por exemplo, nos roteadores dos ISP – *Internet Service Provider* e na camada de aplicação. São aplicáveis também nas redes de distribuição de conteúdo. Para comunicações sem fio, a codificação de rede deve suportar aplicações *ad hoc* de múltiplos saltos, redes de sensores sem fio, redes *mesh*, dentre outras.

Diversos estudos na área de codificação de rede estão sendo realizados atualmente, na intenção de difundir melhor esta nova tecnologia e propor soluções para problemas nas redes de comunicação. Alguns destes estudos são apresentados neste trabalho, tais como: comunicação sem fio, distribuição de conteúdo, maximização da vazão, minimização de energia por *bit*, *download* de arquivos, dentre outros.

2 Benefícios da Codificação de Rede

A implementação da codificação de rede traz algumas vantagens sobre o roteamento que utiliza o método *store-and-forward*, tais como, maximização da vazão, minimização do atraso e minimização de energia por *bit*. Esses benefícios serão descritos nas subseções a seguir, juntamente com uma análise sobre a codificação de rede em algumas topologias específicas selecionadas para este fim.

2.1 Maximização da Vazão

Em relação à maximização da vazão, seja o cenário considerado em [5], onde se tem a situação de dois fluxos de informação, ambos a uma taxa de *B bits* por segundo, chegando a um nó, disputando um mesmo enlace de saída, com capacidade de *B bits* por segundo. Com a codificação de rede, pode ser possível aumentar a taxa de transferência, encaminhando ambos os fluxos através do enlace de saída, ao mesmo tempo. Usando-se a codificação de rede, o nó pode misturar os dois fluxos em conjunto através de uma função, OU Exclusivo (XOR), *bit*-a-*bit*, e enviando o fluxo misturado através do enlace. Neste caso, a função XOR é computada no nó. Assim, o uso da função XOR aumenta o rendimento da rede.

Por exemplo, a Figura 2.1 ilustra uma rede com roteadores, enlaces, remetentes e destinatários. Neste exemplo, os enlaces possuem a mesma capacidade de *B bits* por segundo e são direcionais. A fonte *s1* deseja transmitir informações ao destinatário *t1*, e a fonte *s2* deseja transmitir informações ao destinatário *t2*. O caminho para que a fonte *s1* chegue ao seu destino é o $s1 \rightarrow A \rightarrow C \rightarrow D \rightarrow F \rightarrow t1$ e o caminho para que a fonte *s2* chegue ao seu destino é $s2 \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow t2$. As fontes compartilham o enlace de gargalo $C \rightarrow D$. Importante ressaltar que a literatura estudada, não apresenta claramente como ocorrerá a decodificação para sessões *unicasts*.

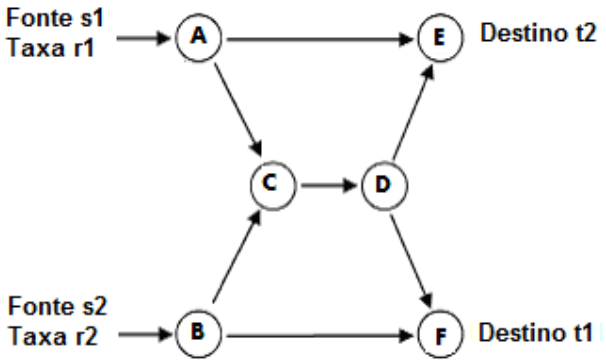


Figura 2.1: Duas sessões unicasts compartilhando um mesmo enlace de gargalo.

Se o mecanismo de roteamento for utilizado, então o enlace $C \rightarrow D$ deve ser compartilhado entre duas sessões, dando origem ao conjunto de taxas atingíveis. Essas taxas atingíveis são as taxas *r1* e *r2*, que variam entre $0 \leq r1max$ e $0 \leq r2max$, respectivamente, e ambas as taxas possuem uma velocidade de $r1 + r2 \leq B$, sendo B o valor total da capacidade dos enlaces. Tal situação é ilustrada na Figura 2.2(a).

No entanto, se os nós da rede implementam a codificação de rede, ambas as sessões podem se comunicar confiantemente com taxa *B*, dando origem ao conjunto de taxas atingíveis, onde *r1* varia entre $0 \leq r1 \leq B$ e *r2* varia $0 \leq r2 \leq B$, como ilustrado na Figura 2.2(b). Para que a taxa *B* seja atingida por ambas as fontes, os dois fluxos devem ser misturados ou codificados no nó *C* usando operação XOR, e decodificados nos nós E e F, usando-se novamente a operação XOR. A figura 2.1 ilustra o fluxo de informações, onde os enlaces $A \rightarrow C$, $A \rightarrow E$, $B \rightarrow C$, $B \rightarrow F$ trafegam os fluxos puros, enquanto os enlaces $C \rightarrow D$, $D \rightarrow E$, $D \rightarrow F$ trafegam o fluxo combinado.

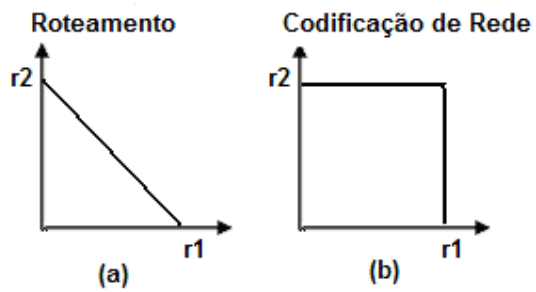


Figura 2.2: Regiões de taxas atingíveis de comunicação para o roteamento e para codificação de rede.

Não é possível que ocorra uma comunicação com uma taxa superior a *B*, uma vez que a fonte e os receptores, em cada sessão são ligados à rede através de um único enlace de capacidade *B*. A Figura 2.2(b) ilustra o conjunto de taxas atingíveis, a chamada região de capacidade para essas sessões nesta rede.

Os primeiros benefícios da codificação de rede foram com relação ao aumento da vazão em uma arquitetura de difusão *multicast*. Como exemplo, a Figura 2.3 ilustra uma rede de comunicação com as fontes *s1* e *s2* e os receptores *T1* e *T2*, onde os vértices são os terminais e as arestas os canais de comunicação. Admite-se que esta rede possui *slots* de tempo. Em cada um dos *slots* de tempo, pode-se enviar um *bit* em cada canal. As fontes *s1* e *s2* geram um *bit* por *slot* de tempo que são denominados de *x1* e *x2*, respectivamente.

Se o receptor *T1* utiliza todos os recursos da rede, ele pode receber dados de ambas as fontes. Pode-se rotear o *bit x1* através do caminho $A \rightarrow T1$ e o *bit x2* da fonte *S2* através do caminho $B \rightarrow C \rightarrow D \rightarrow T1$, como ilustrado na Figura 2.3 (a). Da mesma maneira, se o receptor *T2* utiliza todos os recursos da rede, ele pode receber dados de ambas as fontes. Pode-se rotear o *bit x1* da fonte *s1* através do caminho $A \rightarrow C \rightarrow D \rightarrow T2$, e o *bit x2* da fonte *s2* através do caminho $B \rightarrow T2$, como ilustrado na Figura 2.3 (b).

Supõe-se uma transmissão *multicast*, na qual ambos receptores desejam receber simultaneamente as informações de ambas às fontes. Tem-se então, uma disputa pelo enlace $C \rightarrow D$, onde só é permitido o envio de um *bit* por slot de tempo. Contudo, deseja-se enviar simultaneamente o *bit x1* para o receptor *T2* e o *bit x2* para o receptor *T1*.

No roteamento tradicional, os fluxos de informação de diferentes fontes, são tratados independentemente uns dos outros. Aplicando esta técnica, é necessário que se tome uma decisão sobre qual informação deve-se transmitir primeiro no enlace $C \rightarrow D$, bits *x1* ou bits *x2*. Se, por exemplo, decide-se enviar *bit x1*, então o receptor *T1* só irá receber *x1*, enquanto receptor *T2* irá receber tanto *x1* quanto *x2*.

Com a codificação de rede, os nós intermediários da rede são capazes de processar os fluxos de informação recebida, e não apenas encaminhar para frente. O nó *C* pode conter *bits x1* e *x2* e através da operação lógica XOR, criar um terceiro *bit* $x3 = x1 + x2$, encaminhando assim, este *bit x3* pelo enlace $C \rightarrow D$. O receptor *T1* recebe $\{x1, x1+x2\}$ e o receptor *T2* recebe $\{x2, x1+x2\}$, cada receptor resolve esse sistema de equações através da operação lógica XOR e recupera os seus *bits* de interesse.

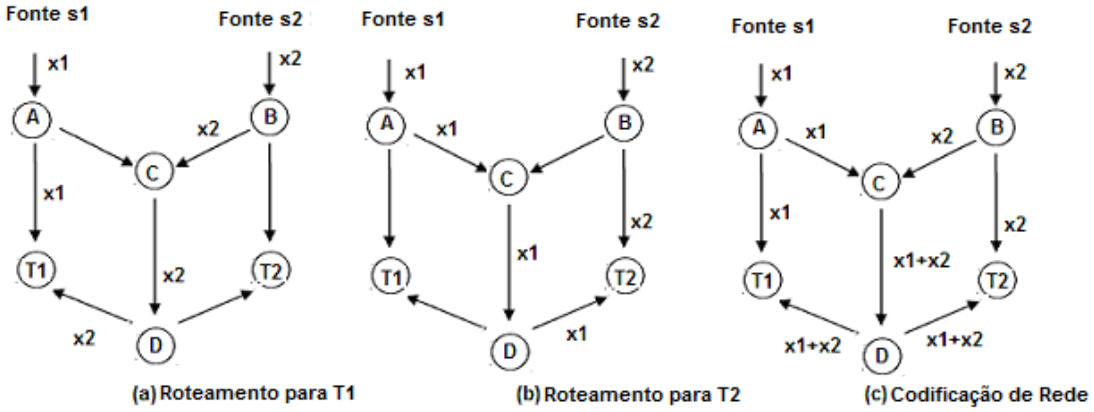


Figura 2.3: Fontes *multicast* s1 e s2 enviam suas informações para os receptores T1 e T2.

2.2 Redes *Wireless*

Em redes sem fio, o roteamento tradicional é ilustrado pela Figura 2.4. Admite-se que o alcance da transmissão de uma estação é de $i + 1$ e $i - 1$, no qual i seja a estação transmissora no momento. Neste cenário, a estação A quer transmitir para estação C e estação C para A. A estação A transmite o pacote $x1$ para a estação B (Figura 2.4(a)). A estação B replica o pacote $x1$ para as estações vizinhas A e C (Figura 2.4(b)). A estação C transmite o pacote $x2$ para a estação B (Figura 2.4(c)). E finalmente, a estação B transmite para as estações vizinhas A e C o pacote $x2$ (Figura 2.4(d)). Portanto, são necessárias quatro transmissões para que ambas as estações recebam os pacotes $x1$ e $x2$.

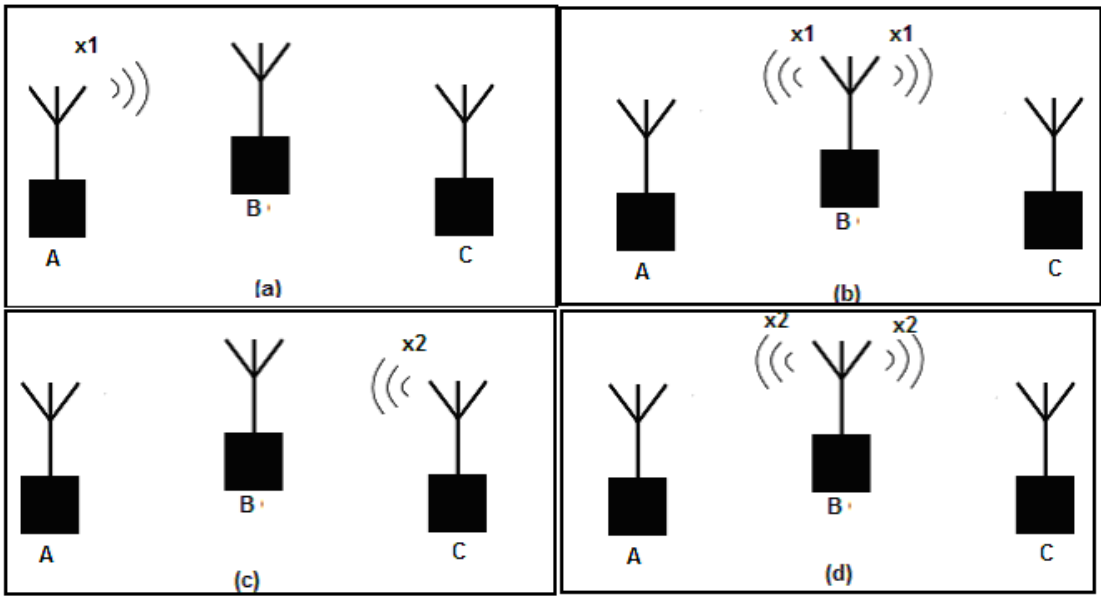


Figura 2.4: Roteamento tradicional de pacotes em redes sem fio.

A Figura 2.5 ilustra o mesmo cenário anterior para o caso da codificação de rede. A estação *A* transmite o pacote *x1* para estação *B*, Figura 2.5(a), em seguida, a estação C transmite o pacote *x2* para estação *B*, Figura 2.5(b), e finalmente, a estação *B* transmite para as estações vizinhas *A* e *C* o pacote *x1 XOR x2*, Figura 2.5(c). Com a codificação de rede, o número de transmissões diminui de quatro para três. Importante ressaltar que a literatura estudada, não apresenta claramente como será realizado a correção de erros.

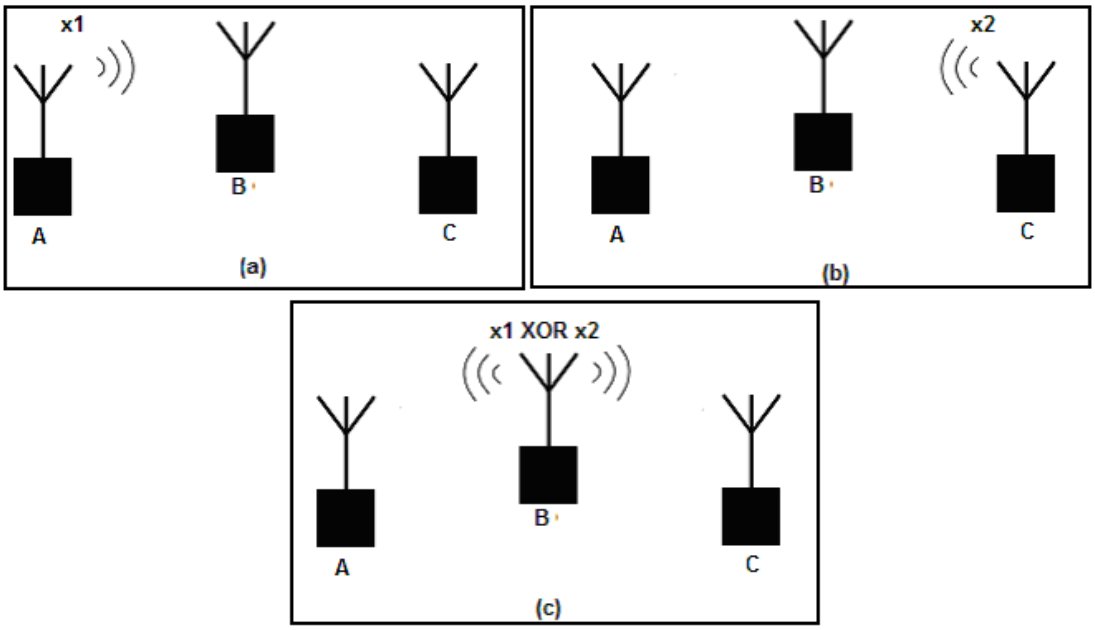


Figura 2.5: Codificação de rede em redes sem fio.

2.2.1 Minimização de Energia por *Bit*

Ainda em redes sem fio, a codificação de rede pode minimizar a quantidade de energia necessária por pacote de informações *multicast*. A Figura 2.6 mostra uma rede sem fio com os nós formando um quadrado. Os nós possuem alcance com seus nós vizinhos na vertical e na horizontal, mas não na diagonal, para obter a comunicação direta. Uma única sessão *multicast* é estabelecida, com um remetente *S* localizado na parte superior central e dois receptores, *R1* e *R2*, localizados no canto inferior esquerdo e canto inferior direito. Supondo-se que cada transmissão necessita de uma unidade de energia, pode-se usar o número de transmissões com um *proxy* para determinar a quantidade de energia que cada pacote *multicast* necessita. Implementando-se o mecanismo de roteamento, é possível

demonstrar que o caminho mínimo é de cinco transmissões *multicast*, entre a fonte *S* e os receptores *R1* e *R2*. Por exemplo, a primeira transmissão envia o pacote da fonte, para os seus dois vizinhos, e quatro outras transmissões para encaminhar o pacote até os dois receptores, como mostra a Figura 2.6(a). Todavia, implementando-se a codificação de rede, apenas 4,5 transmissões por pacote em média são necessárias, com nove transmissões para dois pacotes *a* e *b*. Por exemplo, três transmissões podem encaminhar o pacote *a* para o receptor *R1*, três transmissões podem encaminhar o pacote *b* para o receptor *R2*, duas transmissões podem encaminhar pacotes *a* e *b* a um nó intermediário entre os receptores *R1* e *R2*, e uma transmissão final pode transmitir *a* + *b* de volta aos receptores, conforme ilustrado na Figura 2.6(b).

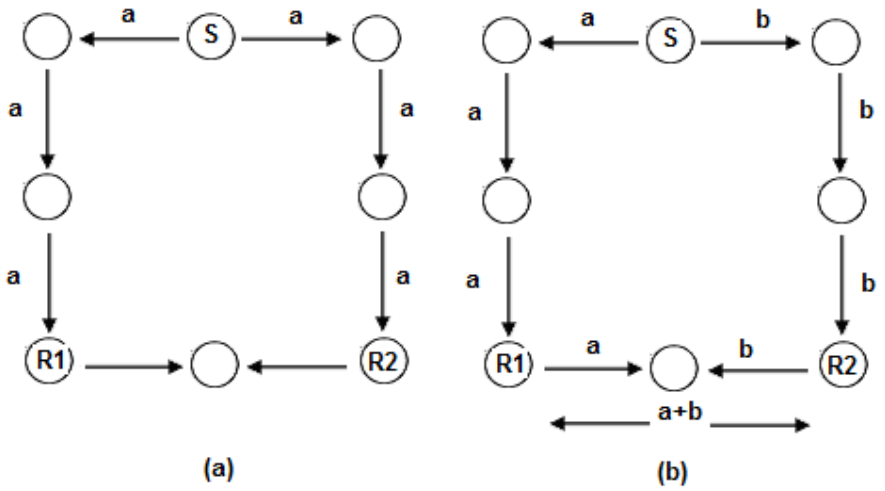


Figura 2.6: Codificação de rede minimizando a energia por pacote.

2.3 Minimização do Atraso

A minimização do atraso também é uma característica da codificação de rede, diminuindo, por exemplo, o número máximo de saltos para um pacote chegar ao destino. A Figura 2.7 ilustra uma rede com quatro nós interconectados, no qual existe um único remetente no topo(nó *S*) e três receptores na parte inferior (nós *A*, *B* e *C*). O valor mínimo de corte (MinCut) entre o emissor e qualquer receptor é dois (enlaces da esquerda e da direita do nó emissor *S*). De acordo com [5], o teorema de Edmonds garante a existência de duas *spanning trees* de nós disjuntos ao longo da qual o remetente pode encaminhar duas unidades de taxa de fluxos (*a* e *b*) aos três receptores. Na Figura 2.7(a), utilizando a técnica de roteamento tradicional nota-se que a profundidade da árvore de S→*B,* *B*→*A* e *A*→*C* é três, o que é, portanto, um eventual atraso se somente o roteamento pode ser usado para

comunicar a taxa de dois. Em contraste, a Figura 2.7(b) mostra que, se a codificação de rede for utilizada, é possível reduzir o atraso para dois, pelo encaminhamento do fluxo *a* ao longo do caminho S→*A* e *A*→*C,* fluxo *b* ao longo do caminho S→*B, B*→*A* e sua mistura *a* + *b* ao longo do caminho S→*C, C*→*B*.

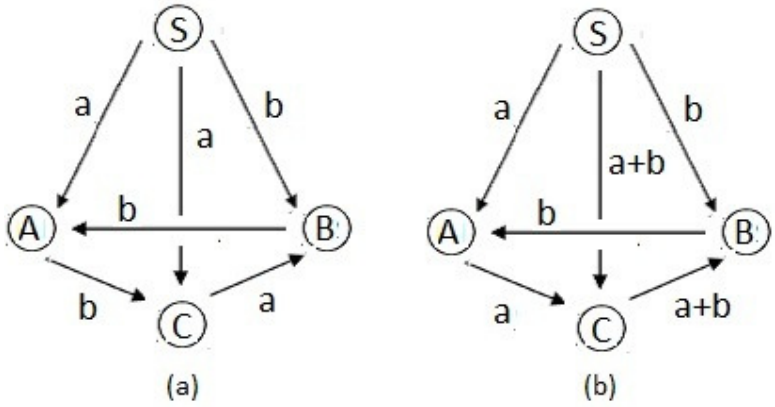


Figura 2.7: Minimização de atraso com Codificação de rede.

2.4 Análise da Codificação de Rede em Algumas Topologias de Interesse

Nesta subseção, mostraremos vários exemplos que demonstram os benefícios da aplicação da codificação de rede em redes específicas. Para a transmissão no meio, considera-se o protocolo de múltiplo acesso TDMA – *Time Division Multiple Access*. Vamos analisar o ganho resultante da aplicação da técnica de codificação de rede ao invés de métodos tradicionais de roteamento. Os cenários onde foi implementada a codificação de rede estão demonstrados por Pappas em [6].

2.4.1 N nós conectados ao mesmo ponto de acesso

Neste cenário, temos *n* nós conectados a um mesmo *Access point*. Cada nó contém um bit de informação (ou um símbolo de informação) que deseja transmitir para outros nós na rede através do *access point*. Vamos analisar inicialmente uma rede com 2 nós, ou seja, $n = 2$, como ilustra a Figura 2.8.

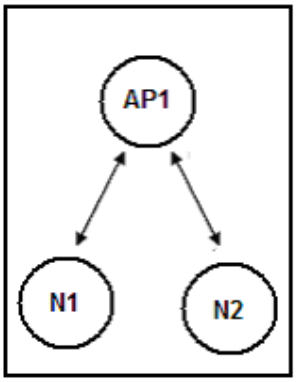


Figura 2.8: Dois nós conectados ao mesmo *access point*.

O nó *N1* contém o *bit* de informação *x*, que será transmitido ao nó *N2* e o nó *N2* possui o bit de informação *y*, que será transmitido ao nó *N1*. Nos dois primeiros *time slots*, o nó *N1* transmite o *bit x* para o *AP1* e, similarmente, o nó *N2* transmite o *bit y* para o *AP1*, como ilustra a Figura 2.9(a) e 2.9(b), respectivamente. Após o recebimento dos *bits x* e *y*, o *AP1* transmite para ambos os nós *N1* e *N2* via *multicast* o *bit* $(x \oplus y)$, conforme ilustrado pela Figura 2.10.

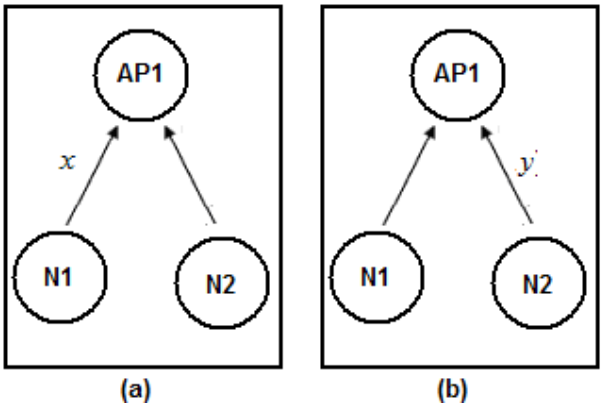


Figura 2.9: Transmissão dos nós para o *access point*.

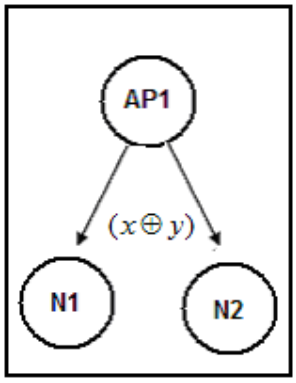


Figura 2.10: Transmissão do *access point* para os nós.

O nó *N1*, que já possuía o *bit x* e agora recebeu $(x \oplus y)$, pode, após o recebimento, calcular o *bit y* através de $y = x \oplus (x \oplus y)$. Da mesma maneira, *N2* obtém o *bit x* através de $x = y \oplus (x \oplus y)$. Portanto, com uma transmissão *multicast* do *AP1* para os nós *N1* e *N2*, são transmitidas as informações necessárias para os dois nós. O ganho, utilizando a codificação de rede, foi de uma transmissão a menos (ganho de 50% de largura de banda para o downlink). Se o método clássico de encaminhamento fosse utilizado, o *AP1* transmitiria o *bit y* para o nó *N1* e o *bit x* para o nó *N2*, ao invés de somente um *bit* $(x \oplus y)$.

Seja agora, o caso com $n = 3$ nós, conforme ilustrado pela Figura 2.11.

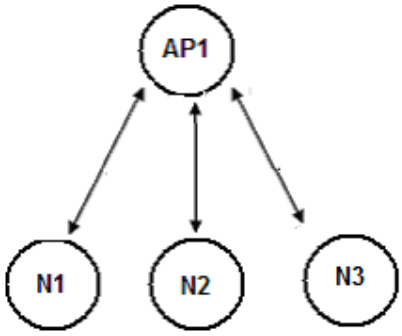


Figura 2.11: Três nós conectados ao mesmo *access point*.

A Figura 2.12 ilustra a seguinte explicação. Nos três primeiros *time slots*, os nós *N1*, *N2*, *N3* transmitem os *bits* de informação *x*, *y* e *z* respectivamente, para o *access point*. Após o recebimento, o *AP1* codifica os *bits x* e *y* em $(x \oplus y)$ e envia para todos os nós. Agora, os nós *N1* e *N2* possuem os *bits* de informação *x* e *y* depois da decodificação, mas o nó *N3* não possui informação suficiente para decodificar os *bits x* e *y*, então, ele armazena o *bit* $(x \oplus y)$ para uso futuro. O *access point AP1* codifica os *bits y* e *z* em $(y \oplus z)$ e o transmite para os nós. Depois desta transmissão, os nós *N1* e *N2* possuem os *bits* de informação *x*, *y* e *z*. O nó *N3*, depois de receber $(y \oplus z)$, pode decodificar o *bit* de informação *y* e logo após, utiliza o *bit* de informação *y* para decodificar o *bit* de informação armazenado $(x \oplus y)$ e encontrar o *bit* de informação *x*. No caso de $n = 3$ nós, o ganho utilizando a codificação de rede foi de uma transmissão a menos (ganho de 33% de largura

de banda para o downlink). Se o método clássico de encaminhamento fosse utilizado, o *AP1* transmitiria o *bit x* para os nós N2 e N3, o *bit y* para os nós N1 e N3, e o *bit z* para os nós *N1* e *N2*.

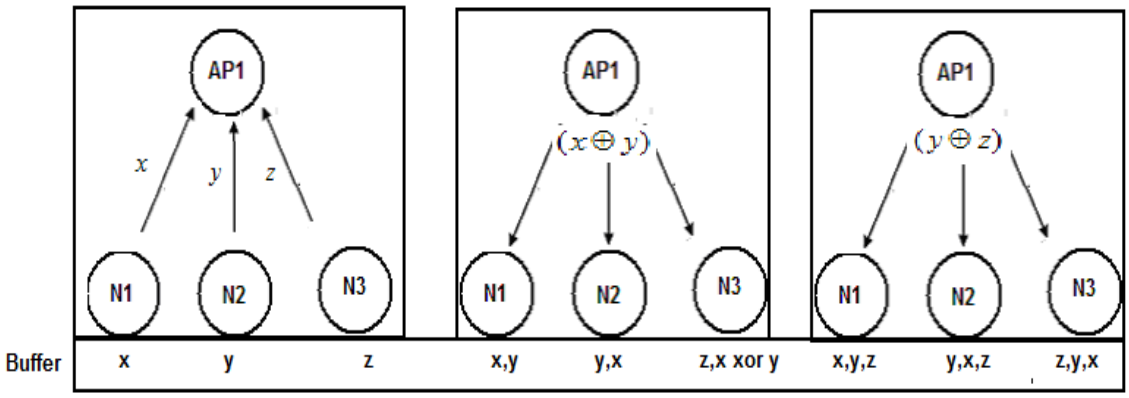


Figura 2.12: Transmissões de uma rede com três nós ligados em um *access point*.

Através destes cenários, para $n = 2$ e $n = 3$ nós, pode-se extrapolar e concluir que quanto maior o número de nós, menor é o ganho. Podemos notar que a codificação de rede é útil para um pequeno número de nós. Mas se a estrutura acima é combinada com outras para formar redes mais complexas. Veremos que a codificação de rede é benéfica. No próximo cenário, iremos analisar as redes tandem.

2.4.2 N nós conectados em mais de um Access Points

Seja uma rede com *n* nós e $n-1$ *access points*, como ilustra a Figura 2.13.

Figura 2.13: Rede tandem.

Uma rede com dois nós conectados ao mesmo *access point*, é a mesma rede do cenário mostrado na Figura 2.8 com $n = 2$. No caso de $n = 3$ nós, os nós *N1*, *N2* e *N3* possuem os *bits* de informação *x*, *y* e *z* respectivamente. Temos dois pontos de acesso *AP1* e *AP2*. A Figura 2.14, ilustra uma rede tandem com $n = 3$ nós junto com as transmissões e os *time slots* em que elas acontecem. Assumimos que algumas transmissões podem ocorrer simultaneamente.

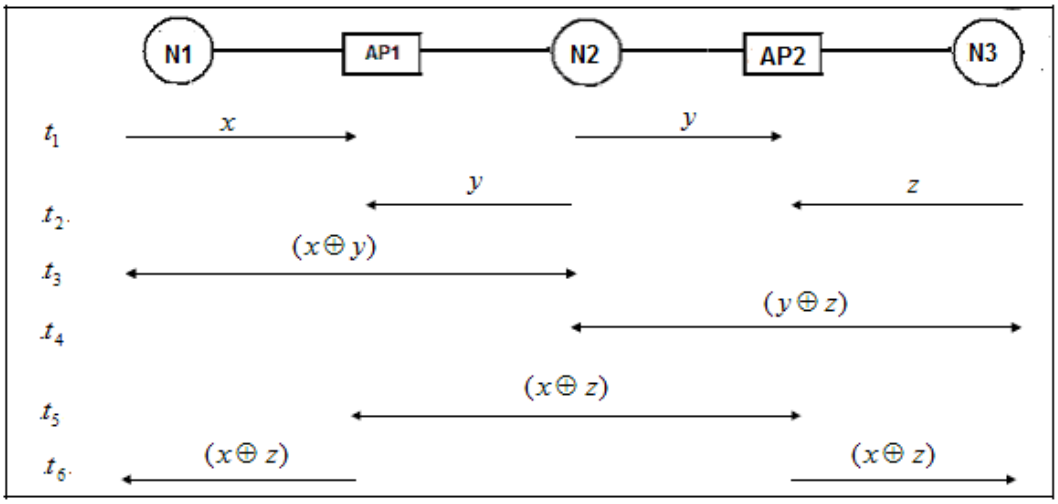


Figura 2.14: Transmissões para a rede tandem com três nós e dois a*ccess points.*

Tabela 2.1 : Conteúdo dos *buffers* em cada unidade de tempo.

	N1	AP1	N2	AP2	N3
	x		y		z
t_1	x	x	y	y	z
t_2	x	x, y	y	y, z	z
t_3	$x, (x \oplus y)$	x, y	$y, (x \oplus y)$	y, z	$z, (y \oplus z)$
t_4	$x, (x \oplus y)$	x, y	$y, (x \oplus y),$ $(y \oplus z)$	y, z	$z, (y \oplus z)$
t_5	$x, (x \oplus y)$	$x, y,$ $(x \oplus z)$	$y, (x \oplus y),$ $(y \oplus z)$	$y, z,$ $(x \oplus z)$	$z, (y \oplus z)$
t_6	$x, (x \oplus y),$ $(x \oplus z)$	$x, y,$ $(x \oplus z)$	$y, (x \oplus y),$ $(y \oplus z)$	$y, z,$ $(x \oplus z)$	$z, (y \oplus z), (x \oplus z)$

A Tabela 2.1 mostra o conteúdo dos *buffers* em cada unidade de tempo da Figura 2.14. Iremos analisar somente os *times slots* onde há ganho com a utilização da codificação de rede. Nos *time slots* t_3, t_4 e t_5, ilustrados na Figura 2.14, tem-se uma transmissão ao invés de duas em cada *time slot*, pois é enviado o pacote codificado, e não os pacotes originais individualmente. Sem a codificação de rede, seriam necessários mais três *time slots* para realizar a transmissão por completo. Portanto, com a codificação de rede, uma rede tandem com três nós e dois *access points*, tem três transmissões a menos.

2.4.2.1 O caso com dois *access points* e dois nós conectados em cada *access point*.

A Figura 2.15 ilustra uma estrutura com dois *access points* e dois nós conectados em cada *access point*, ou seja, a rede contém dois *access point* interconectados e quatro nós conectados a eles. Os nós *N1* e *N2* são conectados ao *access point* AP1 e os nós N3 e N4 ao AP2. A Tabela 2.2, resume as ações em cada unidade de tempo.

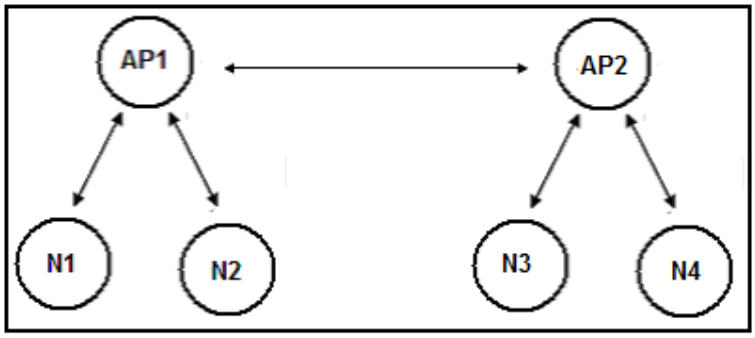


Figura 2.15: Dois *access point* e dois nós conectados em cada *access point*.

Tabela 2.2: Conteúdo dos *buffers* em cada unidade de tempo.

Tempo	Ação	AP1	AP2	N1	N2	N3	N4
t_0				w	x	y	z
t_1	N1→AP1: w N3→AP2: y	w	y	w	x	y	z
t_2	N2→AP1: x N4→AP2: z	w, x	y, z	w	x	y	z
t_3	AP1→N1,N2: $w \oplus x$ AP2→N3,N4: $y \oplus z$	w, x	y, z	w, x	w, x	y, z	y, z
t_4	AP1→AP2: w	w, x	w, y, z	w, x	w, x	y, z	y, z
t_5	AP2→AP1: y AP2→N3,N4: w	w, x, y	w, y, z	w, x	w, x	w, y, z	w, y, z
t_6	AP1→AP2: x AP1→N1,N2: y	w, x, y	w, x, y, z	x	w, x, y	w, y, z	w, y, z
t_7	AP2→AP1: z AP2→N3,N4: x	w, x, y, z	w, x, y, z	x	w, x, y	w, x, y, z	w, x, y, z
t_8	AP2→N1,N2: z	w, x, y, z	w, x, y, z	w, x, y, z	w, x, y, z	w, x, y, z	w, x, y, z

Os nós *N1*, *N2*, *N3* e *N4* possuem os *bits* de informação *w*, *x*, *y* e *z*, respectivamente. Para efeito de análise, cada nó da rede deve possuir todos os *bits* de informação através das transmissões após um período de tempo. No primeiro *time slot*, em t_1, o nó *N1* envia para o *access point AP1* o *bit* de informação *w*, e simultaneamente, o nó *N3* envia para o *AP2* o *bit* de informação *y*. No próximo *time slot*, em t_2, o nó *N2* e *N4* enviam para *AP1* e *AP2* os *bits* de informação *x* e *z*, respectivamente. Agora, os *access point AP1* e *AP2* possuem os *bits* de informação *w*, *x* e *y*, *z*, respectivamente. No próximo *time slot*, em t_3, *AP1* transmite para *N1* e *N2* o *bit* de informação $w \oplus x$ e simultaneamente o *AP2* transmite para *N3* e *N4* o *bit* de informação $y \oplus z$. Os nós *N1* e *N2* decodificam *x* de $w \oplus x$ e *w* de $w \oplus x$, respectivamente, e os nós *N3* e *N4*, decodificam *z* e *y* de $y \oplus z$, respectivamente. Todas as outras ações são demonstradas na Tabela 2.2 acima. Neste cenário, comprova-se que a codificação de rede não tem grande vantagem

sobre o método clássico. O ganho foi de apenas duas transmissões no *time slot* t_3, uma transmissão para cada *access point.* A Figura 2.16 mostra as ações da Tabela 2.2.

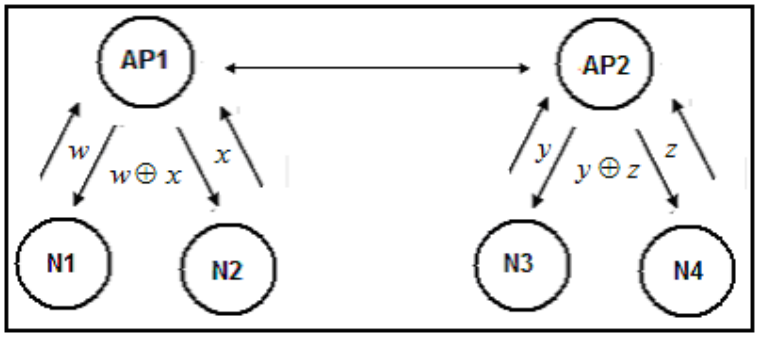


Figura 2.16: Ações da Tabela 2.2.

2.4.2.2 O caso com dois *access points* e dois nós em cada, conectados em um servidor

Os nós *N1*, *N2*, *N3*, *N4* possuem os *bits* de informação *w*, *x* e *y*, *z* respectivamente. Para efeito de análise, cada nó da rede deve possuir todos os *bits* de informação através das transmissões após um período de tempo. A Figura 2.17 ilustra a estrutura da rede com dois *access points* e dois nós em cada, conectados à um servidor e as Figuras 2.18 (a) e (b) ilustram as ações dos *time slots* t_1 a t_3 e t_4 a t_7, respectivamente, também demonstradas pela Tabela 2.3. As ações de t_8 e t_9 são demonstradas na Tabela 2.3.

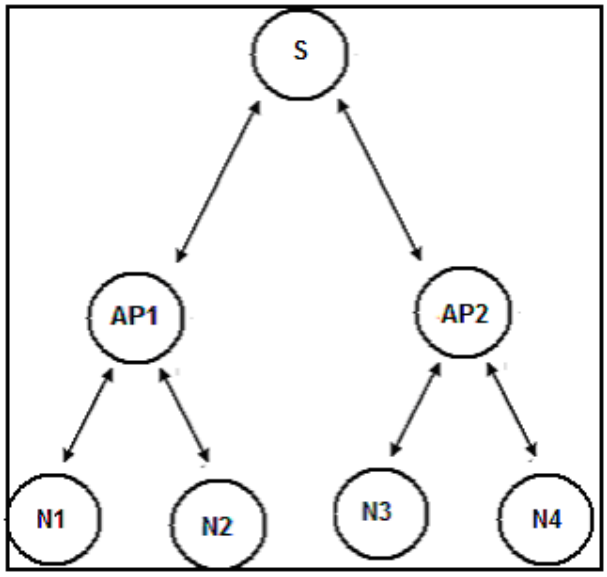


Figura 2.17: Estrutura da rede com dois *access points* e dois nós em cada, conectados em um servidor.

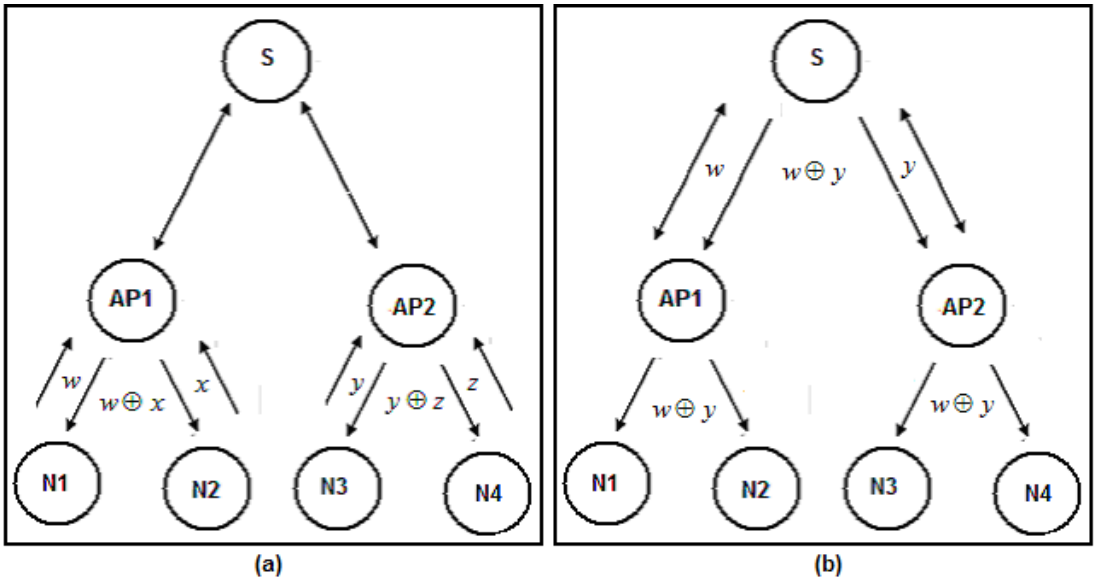


Figura 2.18: (a)ações dos *time slots* t_1 a t_3, (b) ações dos *time slots* t_4 a t_7.

Tabela 2.3 :. Conteúdo dos *buffers* em cada unidade de tempo.

Tempo	Ação	AP1	AP2	S	N1	N2	N3	N4
t_0					w	x	y	z
t_1	N1→AP1: w N3→AP2: y	w	y		w	x	y	z
t_2	N2→AP1: x N4→AP2: z	w, x	y, z		w	x	y	z
t_3	AP1→N1,N2: $w \oplus x$ AP2→N3,N4: $y \oplus z$	w, x	y, z		w, x	w, x	y, z	y, z
t_4	AP1→S: w	w, x	y, z	w	w, x	w, x	y, z	y, z
t_5	AP2→S: y	$w, x,$	y, z	w, y	w, x	w, x	y, z	y, z
t_6	S→AP1,AP2: $w \oplus y$	$w, x,$ y	$w, y,$ z	w, y	w, x	w, x	y, z	y, z
t_7	AP1→N1,N2: $w \oplus y$ AP2→N3,N4: $w \oplus y$ AP1→S: x	$w, x,$ y	$w, y,$ z	$w, x,$ y	$w, x,$ y	$w, x,$ y	$w, y,$ z	$w, y,$ z
t_8	AP2→S: z	$w, x,$ y	$w, y,$ z	$w, x,$ y, z	$w, x,$ y	$w, x,$ y	$w, y,$ z	$w, y,$ z
t_9	S→AP1,AP2: $x \oplus z$	$w, x,$ y, z	$w, x,$ y, z	$w, x,$ y, z	$w, x,$ y	$w, x,$ y	$w, y,$ z	$w, y,$ z
t_{10}	AP1→N1,N2: $x \oplus z$ AP2→N3,N4: $x \oplus z$	$w, x,$ y, z	$w, x,$ y, z	$w, x,$ y, z	$w, x,$ y, z	$w, x,$ y, z	$w, x,$ y, z	$w, x,$ y, z

Neste exemplo, foi demonstrado que a codificação de rede tem ganho de quatro transmissões, no *time slot* t_3, uma transmissão para cada *access point*, ganho de uma transmissão em t_6 e ganho de uma transmissão em t_9.

3 Codificação de Rede

3.1 Introdução

A codificação de rede é um método que visa alcançar o máximo fluxo de informações possíveis em uma rede. Este método surgiu da teoria da informação e teoria da codificação, tendo como foco obter uma melhor eficiência de transmissão de dados em uma rede.

Reforça-se aqui, a idéia de que os nós intermediários de uma rede, que tradicionalmente apenas encaminham fluxos recebidos de informação, com a codificação de rede serão capazes de combinar esses fluxos, a fim de criar um novo fluxo combinado através da operação XOR. A combinação ou processamento desses fluxos consiste em uma combinação linear de um determinado número de pacotes, com coeficientes aleatórios, para posterior encaminhamento. Em seguida, o destino recebe várias combinações lineares advindas de diferentes caminhos que resulta em um sistema linear de equações.

Este método é diferente do encaminhamento tradicional de pacotes, no qual cada pacote é reenviado independentemente pela rede até o destino, que por sua vez, é responsável por reunir todos os pacotes originais para encontrar o conteúdo enviado.

Antes de apresentar os conceitos fundamentais de codificação de rede linear, será feita uma descrição sobre o campo finito e a teoria dos grafos.

3.2 Campo Finito

Um campo finito possui um número finito de elementos. O campo finito é um conjunto com duas operações, normalmente adição e a multiplicação, obtendo-se os seguintes axiomas:

Para todo *a*, *b* em *F*, ambas as operações $a+b$ e $a \cdot b$ estão em *F*, ou seja, adição e multiplicação são operações binárias em *F*;

Para todos *a*, *b* e *c* em F, as seguintes igualdades devem ser mantidas: $a+(b+c)=(a+b)+c$ e $a \cdot (b \cdot c)=(a \cdot b) \cdot c$;

Para todo *a* e *b* em *F*, as seguintes igualdades devem ser mantidas: $a+b=b+a$ e $a \cdot b=b \cdot a$;

Existe um elemento em *F*, chamado de elemento de identidade aditivo e é denotado por 0, tal que para todo *a* em *F*, $a+0=a$. Similarmente, existe um elemento de identidade multiplicativo denotado por 1, tal que para todo a em *F*, $a \cdot 1=a$;

Para todo *a*, *b* e *c* em *F*, a seguinte igualdade deve ser mantida: $a \cdot (b+c)=(a \cdot b)+(a \cdot c)$.

Um exemplo de campo com número finito de elementos é o de *Galois*, que será descrito a seguir.

3.2.1 Campo Finito de *Galois*

O campo de *Galois* é um campo finito que possui $q=p^n$ elementos, no qual p é um número primo e o tamanho do campo é $GF(q)=\{0,1,...,q-1\}$, onde q é a ordem do campo de *Galois*. Por exemplo, $GF(2)=\{0,1\}$ e $GF(3)=\{0,1,2\}$. Um campo de *Galois* de ordem q é um conjunto de q elementos com duas operações binárias módulo-p de adição e multiplicação. As operações binárias para $GF(2)$, $GF(2^2)$ e $GF(3)$ são demonstradas a seguir.

Tabela 3.1: Operações binárias para o campo finito $GF(2)$.

GF(2)					
+	0	1	X	0	1
0	0	1	0	0	0
1	1	0	1	0	1

Tabela 3.2: Operações binárias para o campo finito $GF(2^2)$.

GF(2^2)									
+	00	01	10	11	X	00	01	10	11
00	00	01	10	11	00	00	00	00	00
01	01	00	11	10	01	00	01	00	01
10	10	11	00	01	10	00	00	10	10
11	11	10	01	00	11	00	01	10	11

Tabela 3.3: Operações binárias para o campo finito $GF(3)$.

GF(3)							
+	0	1	2	X	0	1	2
0	0	1	2	0	0	0	0
1	1	2	0	1	0	1	2
2	2	0	1	2	0	2	1

3.2.1.1 Espaço vetorial

Seja $V_n = GF(2)^n$ um espaço vetorial com *n* dimensões. Os elementos de V_n são chamados de vetores e os elementos de $GF(2)$ são chamados de escalares com valores 0 e 1. Para um melhor entendimento, seja *n* = 2. O espaço vetorial V_2 de todas as 2-tuplas em $GF(2)$, consiste em $2^2 = 4$ vetores.

Um conjunto de vetores $v_1, v_2, ..., v_k$ em um espaço vetorial V sobre o campo de *Galois* é dito ser linearmente dependente se e somente se existirem *k* escalares $a_1, a_2, ..., a_k$ de *F*, nem todos zero, tal que $a_1 v_1, a_2 v_2, ..., a_k v_k = 0$.

O conjunto de vetores $v_1, v_2, ..., v_k$ é dito linearmente independente se não for dependente linearmente. Isso é, se $v_1, v_2, ..., v_k$ são linearmente independentes, ou seja, $a_1 v_1, a_2 v_2, ..., a_k v_k \neq 0$, a menos que $a_1 = a_2 = ... = a_k = 0$.

Um conjunto de vetores é dito de span em um espaço vetorial V se cada vetor em V é uma combinação linear de vetores no conjunto. Em qualquer espaço vetorial ou

subespaço existe pelo menos um conjunto *B* de span de vetores linearmente independentes no espaço. Esse conjunto é chamado de base do espaço vetorial. O número de vetores nesta base do espaço vetorial é chamado de dimensão do espaço vetorial. Considerando o espaço vetorial $V_n = GF(2)^n$, este espaço vetorial tem por base as seguintes n-tuplas:

$e_0 = (1,0,0,...,0,0)$
$e_1 = (0,1,0,...,0,0)$
...
$e_{n-1} = (0,0,0,...,0,1)$

Então, cada n-tupla $(a_0, a_1, ..., a_{n-1})$ em V_n pode ser expresso pela combinação linear de $e_0, e_1, ..., e_{n-1}$ dada a seguir:

$$(a_0, a_1, ..., a_{n-1}) = a_0 e_0 + a_1 e_1 + ... + a_{n-1} e_{n-1} \quad (3.1)$$

Portanto $e_0, e_1, ..., e_{n-1}$ é span do espaço vetorial V_n de todas as n-tuplas em $GF(2)$.

3.3 Teoria dos Grafos

A teoria dos grafos é o estudo de grafos que representam a relação entre objetos em uma determinada coleção. Graficamente, é representado por uma figura com uma coleção de nós ou vértices, significando os objetos, e um conjunto de arestas que ligam pares de nós, configurando assim, a relação entre os objetos. Um grafo é denotado por $G = (V, E)$ no qual V é o conjunto de nós e E é o conjunto de arestas. Uma aresta $e \in E$ onde $e = (x, y)$ se x e $y \in V$. Um grafo direto é um grafo em que o fluxo ao longo da aresta pode ser realizado apenas em um sentido. Contudo, pode-se substituir uma aresta que possui fluxo nos dois sentidos por duas arestas em sentidos contrários. Um grafo indireto é dito se a aresta que liga dois nós possui dois sentidos, como por exemplo, uma estrada ligando duas cidades, via de mão dupla.

Em um grafo dirigido ou um dígrafo $G = (V, E)$, uma aresta $e = (x, y)$ é considerada dirigida se a partir de *x* a *y* permite fluxo positivo em uma direção e fluxo zero

na direção oposta. Um grafo não dirigido é um grafo em que as relações entre os nós são simétricas, ou seja, arestas $e1 = (x, y)$ e $e2 = (y, x)$ são idênticas e ambos são representados por uma linha indireta ligando *x* e *y*, o acesso aos nós não possui uma direção indicada.

Um grafo ainda pode ser acíclico ou cíclico. O grafo é dito acíclico se ele não contém ciclos dirigidos como ilustrado pela Figura 3.1(a). Já, o grafo cíclico, contém pelo menos um ciclo dirigido, a Figura 3.1(b) ilustra dois ciclos dirigidos, A $\rightarrow$C, C $\rightarrow$D, D $\rightarrow$A e B $\rightarrow$C, C $\rightarrow$D e D $\rightarrow$B.

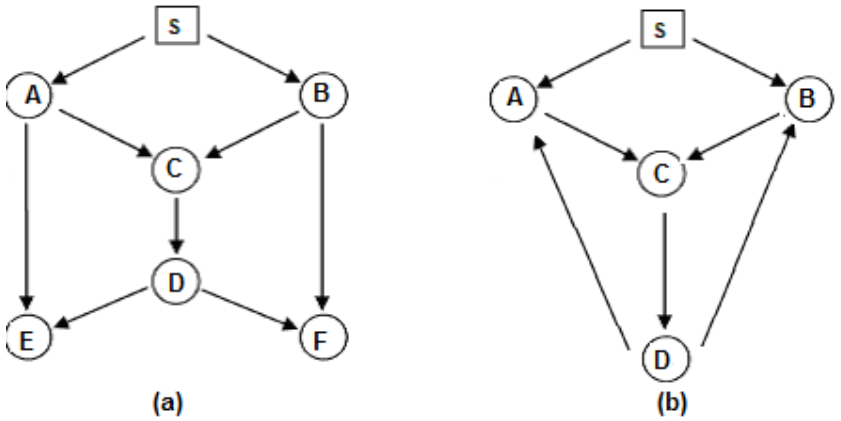


Figura 3.1: Rede acíclica(a) e rede cíclica(b).

Quando duas arestas compartilham o mesmo nó tem-se as Arestas adjacentes. Da mesma maneira, dois nós são adjacentes quando compartilham uma aresta em comum que faz a ligação entre os nós.

Para representar uma rede de comunicação, a teoria dos grafos utiliza o grafo valorado atribuindo valores nas arestas direcionadas do grafo. Tais valores são a capacidade enviar um fluxo ao longo da aresta não excedendo sua capacidade. Um fluxo deve satisfazer a restrição que determina que a quantidade de fluxo na entrada do nó é igual à quantidade de fluxo na saída do nó. Observa-se que esta restrição não é válida para a fonte da rede, pois ela possui mais fluxo de saída do que fluxo de entrada.

Conforme a referência [6], supondo um grafo finito $G = (V, E)$ dirigido em cada aresta $(u,v) \in E$ com uma capacidade real não negativa $c(u,v)$. Se $(u,v) \notin E$, então,

assume-se que $c(u,v) = 0$. Distinguem-se dois elementos s, $t \in V$, fonte e receptor respectivamente. O fluxo da rede é uma função real se $f : VxV \rightarrow R$ possuindo as seguintes propriedades para todos nós u e v:

1 Restrições de capacidade: $f(u,v) \leq c(u,v) \ \forall (u,v) \in E$. O fluxo ao longo da aresta não pode exceder essa capacidade;

2 Desvio de Simetria: $f(u,v) = - f(u,v)$. O fluxo de rede de u para v deve ser o oposto do fluxo de v para u;

3 Conservação do fluxo: $\sum_{u \in V} f(u,w)$ para todo nó u diferente de s e de t. O fluxo de rede para o nó é zero, exceto para a fonte, que produz o fluxo e o receptor que recebe o fluxo.

3.3.1 Teorema do Fluxo Máximo Corte Mínimo - *Max-Flow Min-cut*

De acordo Saireddy em [7], um dos problemas das redes de comunicação é de descobrir a quantidade máxima de informação que pode ser transmitida em um determinado momento. Por exemplo, engenheiros de tráfego querem saber a taxa de fluxo máximo de veículos nas estradas. Esta informação será usada para tomar uma decisão sobre a largura das estradas. Outro exemplo é descobrir o número máximo de chamadas simultâneas entre duas cidades através de linhas terrestres, satélites e torres de microondas operadas por uma companhia telefônica.

Em uma rede que transmite fluxos de dados da origem S ao destino T, o valor mínimo de corte é o número de enlaces destruídos para interromper esta transmissão. Um corte em um grafo $G = (V,E)$ é uma partição dos nós V em dois conjuntos S e T. Cada aresta $(u,v) \in E$ com $u \in S$ e $v \in T$ é dito ser a passagem de corte e é uma aresta de corte. No fluxo de rede, o tamanho do corte é definido como sendo a soma dos pesos das arestas que cruzam o corte da fonte ao receptor.

Para arestas com capacidade unitária, o valor de corte é chamado de tamanho do corte. Existe um único valor de corte mínimo e, de acordo com o grafo, vários cortes mínimos.

O teorema do fluxo máximo corte mínimo diz que o valor de corte mínimo para interromper uma comunicação entre a fonte S e o destino T é a taxa máxima de

informação que podemos enviar entre ambas as fontes. Considerando-se um grafo $G = (V, E)$ com arestas de capacidade unitária, um nó fonte *S* e um nó receptor *T*, e ainda que, o corte mínimo entre *S* e *T* é igual a *H*. Logo, a informação pode ser enviada de *S* para *T*, a uma taxa máxima igual a *H*. De forma equivalente, existem *H* caminhos disjuntos entre *S* e *T*. Para análise, a Figura 3.2(a) ilustra uma rede em topologia borboleta. Esta rede possui um nó fonte *S* e dois nós receptores *R1* e *R2*. A Figura 3.2(b) e 3.2(c) ilustra claramente a quantidade de caminhos disjuntos entre a fonte e os receptores *R1* e *R2*. A quantidade de cortes (dois cortes) nesta figura será a capacidade de taxa máxima igual a duas unidades por unidade de tempo.

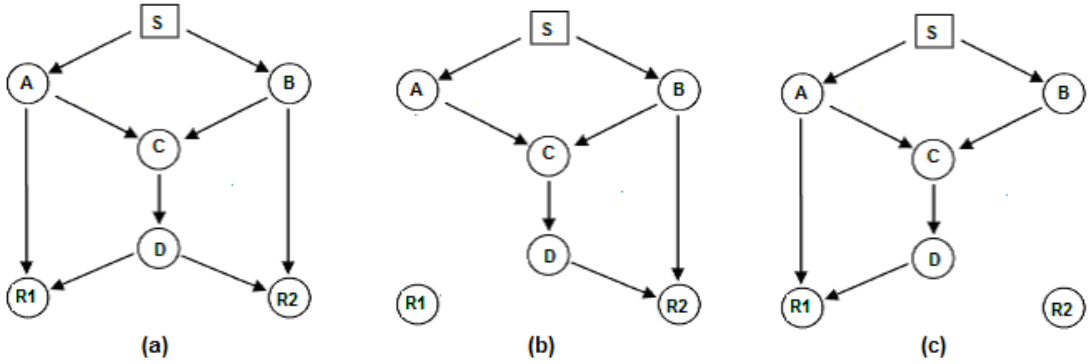


Figura 3.2: Teorema do Fluxo Máximo Corte Mínimo.

3.4 Fundamento Teórico da Codificação de Rede

Utilizando os recursos da Teoria dos Grafos, interpretamos uma rede de comunicação como sendo um grafo direcionado. As arestas do grafo representam os caminhos nos quais os fluxos de informações podem ser enviados. Estes fluxos são os pesos de uma aresta. Estes pesos são utilizados pelo teorema *Max-Flow min-cut*, através deste teorema, é possível determinar a quantidade máxima de informações que a aresta suporta entre dois nós na rede. Segundo Cai e*t al.* em [8], demonstra-se que, com a utilização da codificação de rede linear, o fluxo máximo é atingível. Porém, o método de encaminhamento tradicional de pacotes não consegue atingir esse fluxo máximo. As Figuras 3.4 e 3.3 representam as duas situações, respectivamente.

O funcionamento básico de um nó intermediário é trabalhar com a possibilidade de combinar dados. As mensagens originais são deduzidas à medida que um

receptor tenha acesso a estes dados combinados. Por este método, como dito anteriormente, é possível atingir o fluxo máximo de informações transmitidas entre dois nós na rede.

Primeiramente, deve-se definir uma rede de comunicação (G, S), onde G é grafo finito direcionado e S é o único nó em G sem arestas de entradas. Uma aresta direcionada em G é chamada de canal em uma rede de comunicação (G, S). O nó S é chamado de fonte, nó responsável por gerar os dados a serem transmitidos pela rede de comunicação.

Um canal no grafo G representa um enlace de comunicação sem ruído em que uma unidade de informação, por exemplo, um *bit*, pode ser transmitido por uma unidade de tempo. Os múltiplos canais de um nó x para outro nó y na rede representam a capacidade de direcionar a transmissão de x até y. Cada canal tem a capacidade de uma unidade.

Conforme a referência [6], no nó fonte S, uma quantidade finita de informações é gerada para outros nós da rede, onde cada nó pode passar qualquer um dos seus dados recebidos para outros nós. Em cada nó não fonte, as informações completas geradas por S são recuperadas. Quando a rede de comunicação é acíclica, as operações em todos os nós podem ser sincronizadas de tal forma que, a mensagem é individualmente codificada e propagada do nó *upstream* ao nó *dowstream* na transmissão. Desta forma, o problema da codificação de rede é independente do atraso de propagação nos canais bem como o atraso de processamento nos nós. Mas, quando a rede é cíclica, a propagação e a codificação de uma sequência de mensagens podem estar envolvidas. Neste caso, a quantidade de atraso torna-se uma parte considerável na codificação de rede.

A Figura 3.3 ilustra uma transmissão *multicast* de dois *bits*, *B1* e *B2*, do nó fonte S para os receptores *R1* e *R2*. Os canais $S \rightarrow A$, $A \rightarrow R1$, $A \rightarrow C$, $C \rightarrow D$, $D \rightarrow R2$ transmitem o *bit B1* e os canais $S \rightarrow B$, $B \rightarrow R2$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow R1$ transmitem o *bit B2*. No canal $C \rightarrow D$, encaminha-se o *bit B1* e logo após o *bit B2*, pois só é permitida a transmissão de um *bit* por unidade de tempo. Note que neste esquema, o nó intermediário envia um *bit* de dados somente se ele recebe o mesmo *bit* a partir de outro nó. Por exemplo, o nó A recebe o *bit B1* e envia a cópia para cada um dos dois canais $A \rightarrow R1$ e $A \rightarrow C$. Similarmente, o nó B recebe o *bit B2* e envia cópia para cada um dos canais $B \rightarrow R2$ e $B \rightarrow C$.

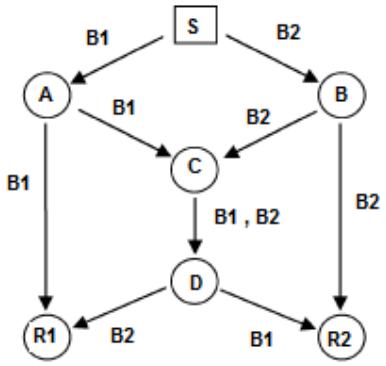


Figura 3.3: Funcionamento do encaminhamento tradicional de pacotes.

A Figura 3.4 ilustra a transmissão *multicast* de dois *bits B1* e *B2* utilizando a codificação de rede linear, tendo o nó *S* como a origem dos dados e os nós *R1* e *R2* como destinatários. Os canais *S*→*A*, *A*→*R1* e *A*→*C* transmitem o *bit B1* e os canais *S*→*B*, *B*→*R2* e *B*→*C* transmitem o *bit B2*. O canal *C* codifica os *bits B1* e *B2* através da operação ou-exclusivo – XOR, e encaminha para o nó *D*, o mesmo replica *B1* XOR *B2* para os receptores *R1* e *R2*. O receptor *R1* recebe {*B1*, *B1*+*B2*} e o receptor *R2* recebe {*B2*, *B1*+ *B2*}, cada receptor resolve esse sistema de equações lineares através da operação lógica XOR fazendo a decodificação e recupera os seus *bits* de interesse. Neste modelo de codificação de rede utilizando a codificação linear, assume-se que não há atraso de processamento nos nós intermediários e que as informações podem ser replicadas ou codificadas. O esquema de codificação e decodificação é suposto para ter sido acordado previamente. Sem a codificação de rede acima, é impossível transmitir dois bits por unidade de tempo a partir da fonte *S* para os destinos *R1* e *R2*. Ficou demonstrada então a vantagem da codificação de rede utilizando a codificação de rede linear.

Vale ressaltar que a replicação e a codificação da informação não aumentam o conteúdo da informação a ser transmitida pela rede e que a taxa de chegada de pacotes por unidade de tempo em um determinado nó é a mesma da taxa de saída de pacotes por unidade de tempo desse mesmo nó. O conteúdo codificado que é transmitido por um determinado nó é a combinação linear dos pacotes recebidos por este nó.

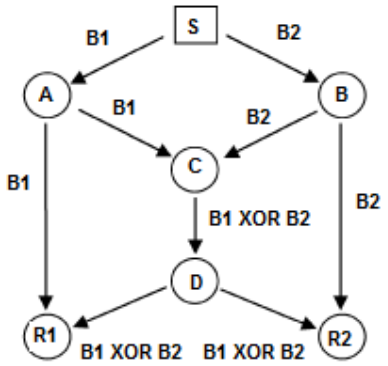


Figura 3.4: Funcionamento do encaminhamento de pacotes utilizando a codificação de rede.

3.5 Modelo Analítico da Codificação de Rede

O modelo analítico da codificação de rede descrito a seguir, foi proposto por [2]. Para cada nó *T*, o conjunto de canais na entrada deste nó é denotado por *IN*(*T*) e o conjunto de canais de saída do nó *T* é *OUT*(*T*). No nó fonte *S*, responsável por gerar as mensagens a serem transmitidas, *IN*(*S*) é um conjunto de canais imaginários que terminam no nó fonte *S*. Os canais imaginários são provenientes de outros nós imaginários. Cada canal imaginário na fonte *S* é considerado como uma mensagem a ser transmitida pela mesma. O número de canais imaginários em *S* é *w*. A Figura 3.5 ilustra uma rede de comunicação acíclica com $w = 2$.

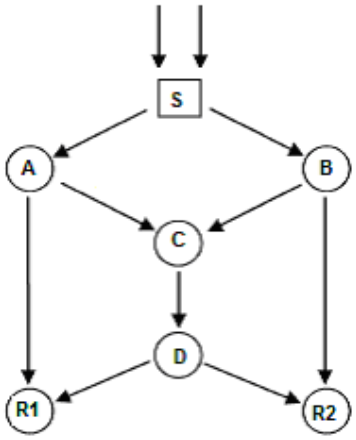


Figura 3.5: Rede de comunicação acíclica com $w = 2$.

O campo finito de *Galois* possui os elementos que representam um símbolo de informação, por exemplo, se *F* é um GF(2), então o símbolo da informação é um *bit*. A

mensagem consiste de *w* unidades de dados e é representado por um vetor linha $x \in F^W$ de dimensão *w*. O nó fonte *S* gera uma mensagem *x* e envia um símbolo através de seus canais de saída. Através da transmissão do símbolo em cada canal *e* na rede, a propagação da mensagem é atingida. Este símbolo transmitido através do canal *e* é uma função da mensagem *x*, $\tilde{f}_e(x) \in F$. Essa função simplifica o mapeamento de símbolos de entrada a partir dos canais de entrada para o símbolo de cada canal de saída. Cada canal possui um mecanismo de codificação que especifica o código de rede. Para o caso de uma rede de comunicação acíclica com uma fonte *S*, existem duas maneiras equivalentes para definir o código de rede, o mapeamento de codificação local e o mapeamento de codificação global.

3.5.1 Mapeamento de Codificação Local

Um código de rede para uma rede de comunicação acíclica com dimensão *w* e valor *F* consiste em um mapeamento de codificação local $\tilde{k}_e : F^{|IN(T)|} \to F$ para cada nó *T* na rede e cada canal $e \in OUT(T)$, sendo F um campo finito e *w* um inteiro positivo.

A topologia de rede acíclica faz o mapeamento de codificação local para construir os valores de $\tilde{f}_e(x)$ transmitidos por todos os canais e através de um procedimento *upstream-to-downstream*. A definição do código de rede acima não dá explicitamente os valores de $\tilde{f}_e(x)$. A definição de $\tilde{f}_e(x)$ é demonstrada abaixo, descrevendo um código de rede com mapeamento de codificação local e os valores derivados obtidos de $\tilde{f}_e(x)$ de forma recursiva.

3.5.2 Mapeamento de Codificação Global

Seja *F* um campo finito e *w* um inteiro positivo. Um código de rede para uma rede de comunicação acíclica com dimensão *w* e valor *F* consiste em um mapeamento de codificação local $\tilde{k}_e : F^{|IN(T)|} \to F$ e o mapeamento de codificação global $F_e : F^w \to F$ para cada canal *e* na rede tal que:

1) Para cada nó T e cada canal $e \in OUT(T)$, $\tilde{f}_e(x)$ é unicamente determinada por $\tilde{f}_d(x), d \in IN(T)$ e $\tilde{k}_e$ é mapeado via $\tilde{f}_d(x), d \in IN(T)$ α $\tilde{f}_e(x)$;

2) Para w canais imaginários e, o mapeamento $\tilde{f}_e(x)$ são projeções a partir do espaço F^w de w diferentes coordenadas.

Para ilustrar os conceitos de mapeamento de codificação local e global, a Figura 3.6 mostra uma rede de comunicação acíclica, onde uma mensagem $x = (b1,b2) \in F^2$.

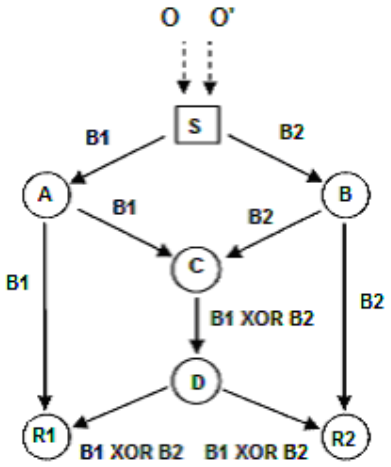


Figura 3.6: Conceitos de mapeamento de codificação local e global.

O código de rede binário de duas dimensões ilustrado segue o mapeamento de codificação global a seguir:

$$\tilde{f}_e(x)\begin{cases} b_1, e = OS, SA, AC, AR1 \\ b_2, e = O'S, SB, BC, BR2 \\ b_1 XOR b_2, e = CD, DR1, DR2 \end{cases}$$

no qual OS e O'S denotam os dois canais imaginários. Os correspondentes mapeamentos de codificação locais $\tilde{k}_e : F^{|IN(T)|} \rightarrow F$ são:

$$\tilde{k}_e \begin{cases} \tilde{k}_{SA}(b_1, b_2) = b_1 \\ \tilde{k}_{SB}(b_1, b_2) = b_2 \\ \tilde{k}_{AR1}(b_1) = b_1 \\ \tilde{k}_{AC}(b_1) = b_1 \\ \tilde{k}_{BR2}(b_2) = b_2 \\ \tilde{k}_{BC}(b_2) = b_2 \\ \tilde{k}_{CD}(b_1 XOR b_2) = b_1 XOR b_2 \\ \tilde{k}_{DR1}(b_1 XOR b_2) = b_1 XOR b_2 \\ \tilde{k}_{DR2}(b_1 XOR b_2) = b_1 XOR b_2 \end{cases}$$

Quando o mapeamento de codificação local $\tilde{k}_e$, no qual $e \in OUT(T)$ é linear, ele corresponde a um vetor coluna $\tilde{k}_e$ de dimensão $|IN(T)|$ tal que $\tilde{k}_e(y) = y \cdot k_e$, onde $y \in F^{|IN(T)|}$ é representado por um vetor linha de símbolos recebidos no nó *T*. Similarmente, quando o mapeamento de codificação global $\tilde{f}_e$ é linear, ele corresponde a um vetor coluna f_e de dimensão *w* tal que $\tilde{f}_e(x)$ é o produto de $x \cdot f_e$, onde um vetor linha *x* de dimensão *w* representa a mensagem gerada por *S*. Abaixo é formulado um código de rede linear onde todos os mapeamentos de codificação local e global são lineares.

3.5.3 Mapeamento de Codificação Local para o Código de Rede Linear

Seja *F* um campo finito e *w* um inteiro positivo. Um código de rede para uma rede de comunicação acíclica com dimensão *w* e valor *F* consiste em uma escalar $k_{d,e}$, chamado de núcleo de codificação local, para todo par adjacente (*d*,*e*). O núcleo de codificação local no nó *T* forma a matriz $|IN(T)|x|OUT(T)|$, ou seja,

$$k_t = [k_{d,e}] d \in IN(T), e \in OUT(T). \qquad (3.2)$$

3.5.4 Mapeamento de Codificação Global para o Código de Rede Linear

Seja *F* um campo finito e *w* um inteiro positivo. Um código de rede para uma rede de comunicação acíclica com dimensão *w* e valor *F* consiste em uma escalar $k_{d,e}$, para todo par adjacente (*d*,*e*) na rede bem como um vetor coluna f_e de dimensão *w* para cada canal tal que:

1) $f_e = \sum_{d \in IN(T)} k_{d,e} \times f_d$, no qual $e \in$ OUT(T);
2) Os vetores f_e correspondem aos *w* canais imaginários $e \in IN(S)$ forma a base natural do vetor espaço F^w.

O vetor f_e é chamado de núcleo de codificação global para cada canal *e*.

Assume-se que a fonte gera a mensagem *x* na forma de um vetor linha de dimensão *w*. O nó *T* recebe símbolos $\tilde{f}_d(x) = x \cdot f_d$, no qual $d \in$ IN(T), a partir do qual calcula o símbolo $\tilde{f}_e(x) = x \cdot f_e$ para enviar para cada canal $e \in$ OUT(T) via a fórmula dada a seguir:

$$\tilde{f}_e(x) = x \cdot f_e = x \cdot \sum_{d \in IN(T)} k_{d,e} \times f_d = \sum_{d \in IN(T)} k_{d,e} \times (f_d \cdot x) = \sum_{d \in IN(T)} k_{d,e} \times f_d(x) \qquad (3.3)$$

Dado o núcleo de codificação local para todos os canais em uma rede acíclica, o núcleo de codificação global pode ser calculado recursivamente em qualquer ordem *upstream-to-downstream*.

A Figura 3.7 ilustra a construção de um código linear para uma rede de comunicação acíclica. Assume-se a ordem alfabética entre os canais *OS*, *O'S*, *SA*, *SB*, *AC*, *AR1*, *BC*, *BR2*, *CD*, *DR1* e *DR2*. As matrizes nos nós que formam os núcleos de codificação local são:

$$K_{d,e} \begin{cases} K_s = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ K_A = K_B = K_C = \begin{pmatrix} 1 & 1 \end{pmatrix} \\ K_D = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{cases}$$

Os vetores de codificação global são:

$$f_e \begin{cases} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, e = OS, SA, AC, AR1 \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, e = O'S, SB, BC, BR2 \\ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, e = CD, DR1, DR2 \end{cases}$$

Os vetores de codificação global e o núcleo de codificação local são ilustrados na Figura 3.7. Os vetores e os núcleos permanecem os mesmos, para qualquer escolha do campo base.

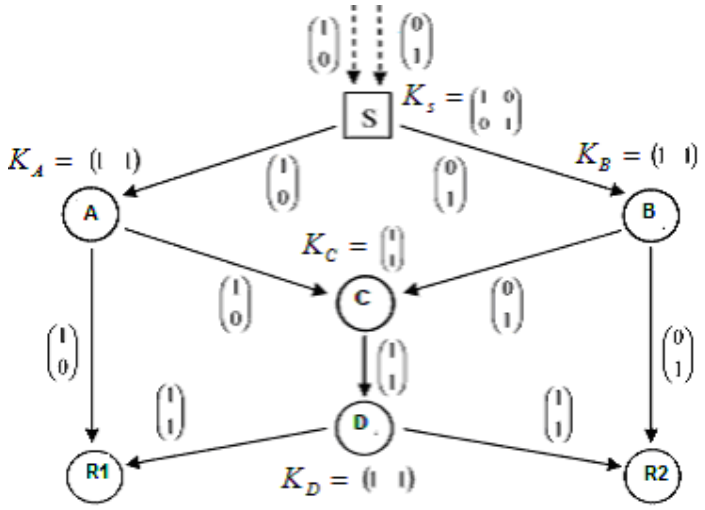


Figura 3.7: Construção de um código linear para uma rede de comunicação acíclica.

Logo quando os dados chegam aos receptores, os mesmos montam as seguintes matrizes:

$$R1 = \begin{pmatrix} b_1^T \\ b_1 XOR b_2^T \end{pmatrix} = \begin{pmatrix} f_{AR1}^T \\ f_{DR1}^T \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

$$R2 = \begin{pmatrix} b_1 XOR b_2^T \\ b_2^T \end{pmatrix} = \begin{pmatrix} f_{BR2}^T \\ f_{DR2}^T \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Para que ocorra a decodificação nos receptores, o receptor *R1* que conhece o b_1, aplica operação lógica XOR ao $b_1 XOR b_2$ e encontra o b_2, e similarmente, o receptor *R2* que conhece o b_2, aplica operação lógica XOR ao $b_1 XOR b_2$ e encontra o b_1.

Para a generalização do caso de duas arestas imaginárias na entrada do nó fonte *S*, a Figura 3.8 ilustra a codificação no núcleo dos nós utilizando a codificação linear de rede.

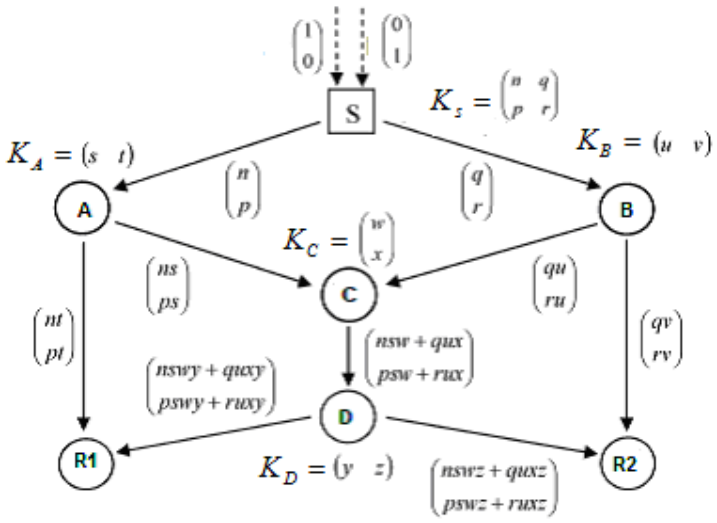


Figura 3.8: Generalização do caso de duas arestas imaginárias na entrada do nó fonte S.

3.6 Propriedades Desejáveis de um Código de Rede Linear

Conforme as referências [1] e [8], o fluxo de dados com quaisquer esquemas de codificação em um nó intermediário devem seguir com a lei da conservação da informação: o conteúdo das informações na saída de qualquer nó não fonte é no máximo igual ao conteúdo de informações recebidas na entrada do nó. O conteúdo das informações de saída pelo nó não fonte deve ser derivado do conteúdo na entrada do mesmo nó. Na subseção teoria dos grafos, vimos que o fluxo máximo da fonte *S* para um receptor *T* é dado pelo *maxflow*(*T*). Pelo teorema do fluxo máximo corte mínimo, a taxa de informações recebidas pelo receptor *T* não pode exceder *maxflow*(*T*). Alguns fatores influenciam para que este valor limite possa ser atingido, como a topologia da rede, a dimensão *w* e o esquema de codificação. Existem três classes de códigos de rede linear, multicast linear,

broadcast linear e dispersão linear. Neste trabalho será abordado somente o multicast linear.

3.6.1 Multicast Linear

Os vetores f_e denotam os núcleos de codificação global de um código de rede linear para uma rede de comunicação acíclica com dimensão *w* e valor *F*. Escrevendo $V_T = \langle \{ f_e : e \in IN(T) \} \rangle$, no qual $\langle \{ f_e : e \in IN(T) \} \rangle$ denota um span linear de um conjunto de vetores, então o código de rede linear qualifica como *multicast* linear se a seguinte afirmação é verdadeira: $\dim(V_T) = w$ para cada nó não fonte *T* com $maxflow(T) \geq w$.

Para que um nó receptor *T* obtenha informação suficiente para decodificar uma mensagem, o nó fonte *S* tem que transmitir uma mensagem de *w* unidades de dados na rede e se e somente se $\dim(V_T) = w$, que é um pré-requisito necessário para que $maxflow(T) \geq w$. Portanto, um *multicast* linear de dimensão *w* é eficiente em um *multicasting* de *w* unidades de dados de informação a todos nós não fonte *T* se atenderem essas condições.

3.7 O Teorema Principal da Codificação de Rede

Considerando um grafo acíclico dirigido $G = (V, E)$, no qual as arestas possuem uma capacidade unitária, com *H* canais imaginários, ou seja, taxa de *H* unidades na fonte e *T* receptores. Assumindo que o valor de corte mínimo de cada receptor é *H*, então, existe um esquema de transmissão multicast sobre um largo campo finito $GF = (q)$, no qual os nós de rede intermediários combinam linearmente as informações recebidas de símbolos $GF = (q)$, este campo fornece as informações provenientes das fontes simultaneamente de cada receptor a uma taxa igual a *H*.

No subseção 3.3 de teoria dos grafos, vimos o teorema do fluxo máximo corte mínimo, por isso sabemos que existem *H* arestas de caminhos disjuntos entre a fonte *S* e cada receptor *T*. Assim, se qualquer dos *T* receptores, por exemplo, somente o receptor T_i está usando a rede, então a informação de *H* canais imaginários pode ser encaminhada para

T_i através de um conjunto de H caminhos disjuntos. Quando vários receptores estão usando a rede de comunicação simultaneamente, os seus conjuntos de caminhos da fonte até eles podem sobrepor. Por isso, os receptores são obrigados a compartilhar os recursos da rede, o que leva a taxas reduzidas, como, por exemplo, compartilhar a aresta de sobreposição, geralmente, este é o canal de gargalo na rede de comunicação ou compartilhar o acesso para a aresta no tempo. Porém, o teorema principal da codificação de rede diz que, é permitido que os nós intermediários na rede combinem os fluxos de informação recebidos, além de somente encaminhá-los, assim, cada um dos receptores T_i estará recebendo os fluxos de informações na mesma taxa como se tivesse acesso exclusivo de recursos da rede. De acordo com a referência [6], este teorema afirma também que, as operações lineares de adições e multiplicações sobre um campo finito $GF = (q)$ são suficientes para os nós intermediários, estabelecendo a existência de códigos de rede lineares sobre um determinado campo finito $GF = (q)$.

4 Aplicações da Codificação de Rede

Como discutido no Capítulo 2, a técnica de codificação de rede pode oferecer importantes benefícios em termos de aumento da vazão, minimização do atraso e redução do consumo de energia. No entanto, a implementação da codificação de rede em redes reais incorre em uma comunicação com um certo *overhead* computacional. Como resultado, uma análise cuidadosa com relação ao custo-benefício deve ser realizada, para avaliar a aplicabilidade da técnica para qualquer determinada configuração de rede.

Conforme Spriintson em [9], a construção de uma rede de codificação em nível de roteadores IP na Internet é improvável que seja prática em um futuro próximo. Isso devido a uma variedade de razões, como exemplos: a implementação da técnica nos roteadores de núcleo de rede, devido à alta taxa de transmissão de dados no núcleo da rede; a necessidade de compatibilidade com uma infra-estrutura já implantada; e a necessidade de múltiplos caminhos de roteamento para fazer a codificação de rede eficaz. Assim, encontrar a configuração de rede que pode se beneficiar da técnica de codificação de rede é um problema desafiador, por si só.

No entanto, de acordo com [5], a construção de uma rede de codificação em redes *overlay* (ou rede de sobreposição) é bastante viável. Em redes *overlay*, os nós são programas que estão sendo executados em computadores em nível de aplicação. Já, as arestas são as conexões existentes, entre computadores, de forma lógica, em nível de transporte. Ou seja, é uma rede lógica construída sobreposta há uma rede física já existente.

Redes *overlay* possuem a sua infra-estrutura base como uma típica rede de distribuição de conteúdo. Também podem ser redes *ad-hoc* ou *peer-to-peer*(P2P) de *hosts* reunidos temporariamente para cumprir determinadas aplicações de comunicação, como transmissão ao vivo, mídia sob demanda, *download* de arquivo, mensagem instantânea, armazenamento, telefonia, conferência ou jogos.

Na subseção 4.1, é discutida a implementação em um sistema de distribuição de conteúdo utilizando a técnica de codificação de rede proposta por [5] e por Wu e*t al*. em [10]. Nas subseções 4.2 a 4.5, é feita uma descrição sobre as aplicações de codificação de rede conforme [5], terminando com aplicações em redes sem fio. Os princípios da implementação prática abordada neste trabalho também foram adotados por Gkandsidis em [11] e por sistemas comerciais como o Microsoft Avalanche.

4.1 Implementação em um Sistema de Distribuição de Conteúdo

Em um sistema de distribuição de conteúdo, todos os terminais devem receber um fluxo de *bits* gerado por uma única fonte de informação. Os *bits* são combinados em símbolos. Cada símbolo, tipicamente, inclui 1 ou 2 *bytes* e representa um elemento de um campo finito $GF(q)$. Os símbolos, por sua vez, são combinados em pacotes, de forma que o pacote p_i é composto de N símbolos $\sigma_1^i, \sigma_2^i, ..., \sigma_N^i$. Já, os pacotes são combinados em gerações, cada geração inclui h pacotes. Conforme [9], nas configurações típicas, os valores de *h* podem variar entre 20 e 100. A Figura 4.1 demonstra o processo de criação de símbolos e pacotes a partir do fluxo de *bits*.

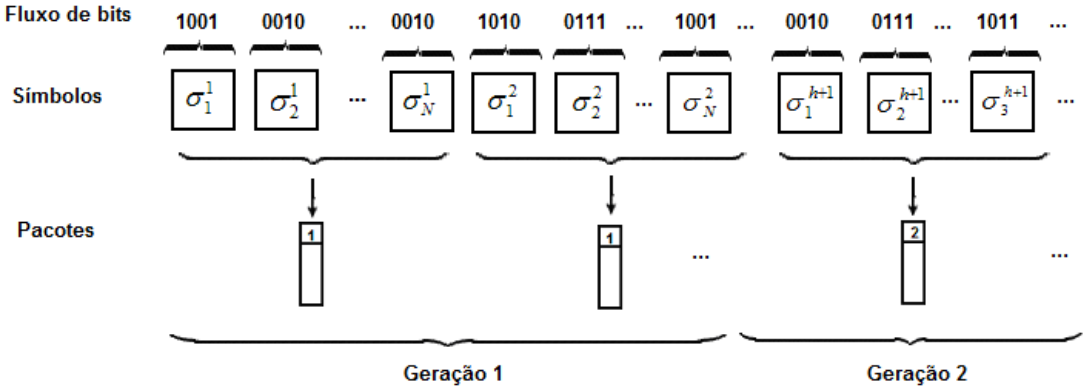


Figura 4.1: Processo de criação do pacote: formando símbolos a partir dos *bits* e pacotes a partir dos símbolos.

A implementação prática da codificação de rede possui uma ideia chave. O esquema proposto é o de misturar pacotes que pertencem à mesma geração, resultando em outro pacote pertencente a esta mesma geração. Essa mistura ou codificação é feita através de símbolos individuais, em vez do pacote inteiro. Ou seja, a codificação é feita símbolo por símbolo no intuito de gerar um novo pacote de mesma geração. Com este esquema, os coeficientes de codificação local pertencem ao mesmo campo como os símbolos, ou seja,

$GF(q)$. Por exemplo, suponha que dois pacotes p_i e p_j são combinados em um novo pacote p_l com os coeficientes de codificação local $\beta_1 \in GF(q)$ e $\beta_2 \in GF(q)$. Então, para $1 \leq y \leq N$ o símbolo y de p_l é uma combinação linear de símbolos y de p_i e símbolos y de p_j, ou seja,

$$\sigma_y^l = \beta_1 \cdot \sigma_y^i + \beta_2 \sigma_y^j \tag{4.1}$$

Este sistema de codificação de rede é baseado na técnica de codificação linear aleatória que escolhe os coeficientes de codificação local uniformemente sobre $GF(q)$ (excluindo o zero). Cada pacote enviado pela rede possui um campo que afirma que seus símbolos são combinações lineares dos símbolos correspondentes dos pacotes originais, ou seja, os pacotes gerados pelo nó de origem. Assim, cada pacote p_l pode ser associado a um vetor de codificação global $\tau_l = \{\gamma_1^l, ..., \gamma_N^l\}$ que capta a dependência entre os símbolos p_l e os símbolos dos pacotes originais.

Outra ideia-chave deste esquema é o de vincular os coeficientes globais de codificação para o pacote. Estes coeficientes são essenciais para o nó terminal ser capaz de decodificar os pacotes originais. Este método é adequado para as configurações com coeficientes de codificação aleatória local. O leiaute dos pacotes é mostrado na Figura 4.2. Observe que cada pacote também inclui o seu número de geração.

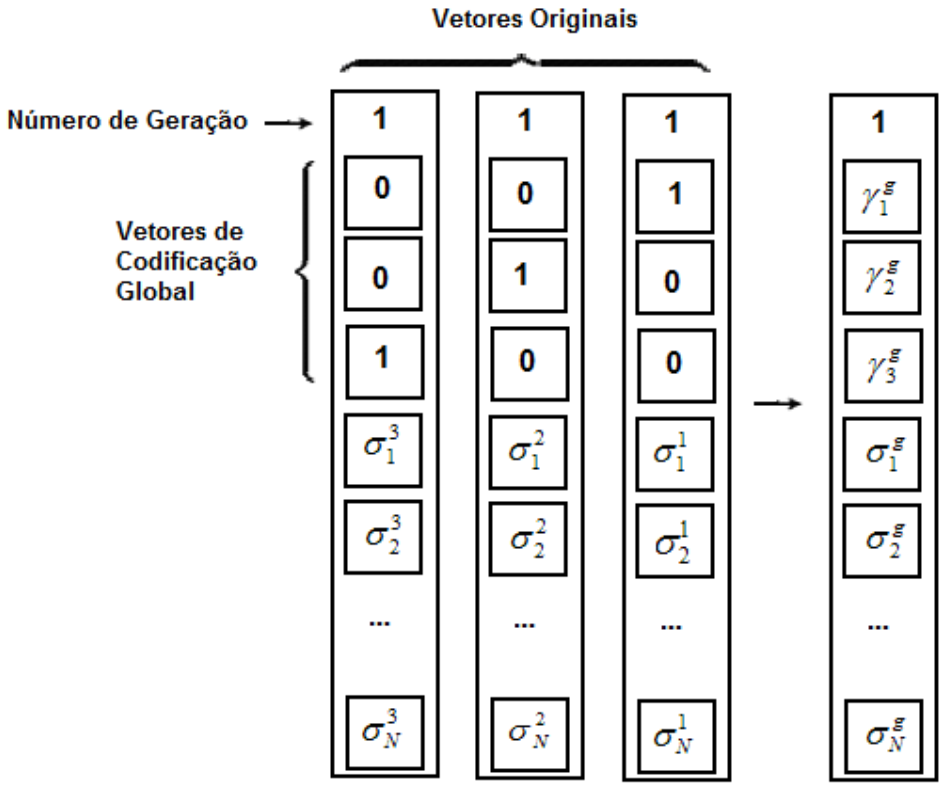


Figura 4.2: Estrutura de um pacote.

Com a adição do vetor de codificação global ao pacote, ocorre um aumento da informação de controle e, consequentemente, há também um aumento de *overhead* de comunicação. O tamanho deste *overhead* depende do tamanho do campo finito. Na verdade, o número de *bits* necessários para armazenar os vetores de codificação global é igual a $h \cdot q$. No caso prático considerado por [9], h é igual a 50 e o tamanho do campo q é igual a dois *bytes*, resultando em um *overhead* total de 100 *bytes* para os pacotes. Com o tamanho do pacote de 1400 *bytes*, o *overhead* constitui cerca de 6% do tamanho total do pacote. Se o tamanho do campo é reduzido a um *byte*, então o *overhead* é reduzido para apenas 3% do tamanho do pacote.

Para que o nó de destino seja capaz de decodificar os pacotes originais é necessário receber h ou mais pacotes linearmente independentes que pertencem à mesma geração. Com a codificação de rede aleatória, a probabilidade de receber pacotes linearmente independentes é elevada, mesmo se alguns dos pacotes são perdidos. A principal vantagem do esquema proposto é que não exige qualquer conhecimento da topologia da rede e lida eficientemente com as mudanças em uma rede dinâmica, por exemplo, devido a falhas de enlace.

A operação de um nó de rede intermediário é mostrada na Figura 4.3. Através de seus *enlaces* de entrada, um nó intermediário recebe pacotes pertencentes a diferentes gerações. Após o recebimento, os pacotes são então, armazenados no *buffer*, e classificados de acordo com seu número de geração. Em qualquer momento, para cada geração, o *buffer* contém um conjunto de pacotes linearmente independentes. Um novo pacote transmitido pelo nó é formado por uma combinação linear aleatória dos pacotes que pertencem à geração atual.

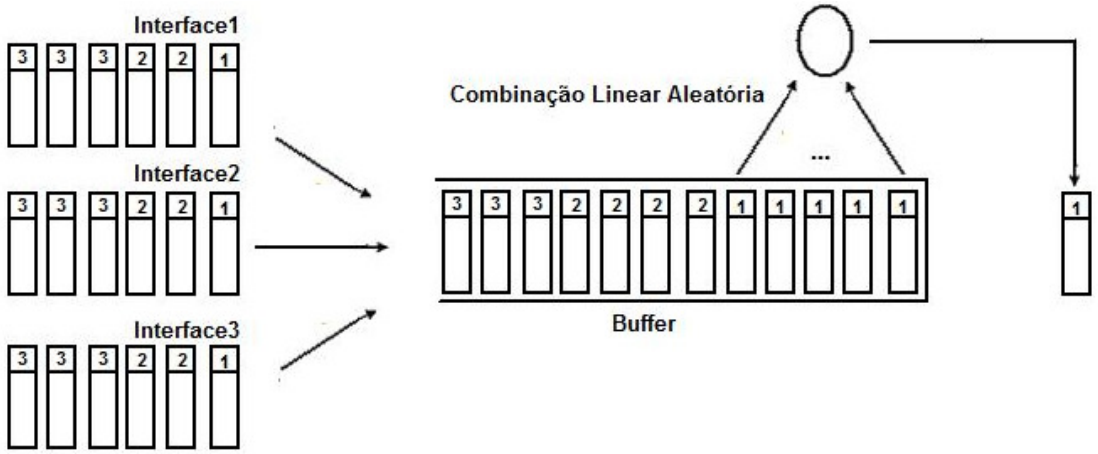


Figura 4.3: Operação em um nó de rede intermediário.

Uma importante decisão de projeto do nó de codificação é a *flushing policy*. O *flushing policy* determina quando uma nova geração de pacotes passa a ser a geração atual. Existem várias *flushing policies* que podem ser considerados. Uma possibilidade é a mudança da geração atual logo que um pacote que pertence a uma nova geração chegar pelos links de entrada. Uma política alternativa é a mudança de geração, quando todos os links de entrada recebem pacotes que pertencem à nova geração. O desempenho das diferentes *flushing policy* podem ser avaliados por uma simulação ou por um estudo experimental, que foge ao escopo deste trabalho de dissertação.

4.2 *Download* de Arquivo

Uma das tarefas de comunicação mais comum de rede é baixar um arquivo de um servidor para um computador cliente. Este arquivo pode ser uma página da *web*, uma imagem, uma música, um filme, um programa, ou outro tipo de documento. Embora, tradicionalmente, as sessões de *download* sejam *unicast*, para sessões *multicast*, a codificação de rede pode potencialmente aumentar o rendimento e, consequentemente, reduzir o tempo médio de *download*.

Para ilustrar como isso pode ser feito, considere o *download* do arquivo em uma rede P2P formada por todos os nós envolvidos no *download* do arquivo. Nós que chegam recentemente à rede, aderem à mesma se conectando a um subconjunto dos nós existentes. Conforme Cohen em [12], o protocolo original e ainda mais popular para *download* de arquivos P2P é o BitTorrent. No BitTorrent, o arquivo é dividido uniformemente em h "pedaços". Cada nó negocia a obtenção de pedaços do arquivo de seus vizinhos, até que o nó obtenha todos os h pedaços, e assim, pode deixar a rede. Depois de um nó obter um novo pedaço, ele anuncia a aquisição aos seus vizinhos, de modo que cada nó sabe quais partes cada vizinho possui. Ao solicitar um determinado pedaço de um vizinho, um nó normalmente pede um pedaço local mais raro, ou seja, uma parte que é menos comum entre todos os vizinhos do nó. Isso garante que os pedaços são propagados uniformemente através da rede, evitando gargalos de informação.

De acordo com [11], um novo protocolo P2P para baixar arquivos com base na codificação de rede é o Avalanche. Conforme Fragouli em [13], no Avalanche, os blocos enviados pelo servidor são combinações lineares aleatórias de todos os blocos originais. Do mesmo modo, os pares enviam combinações lineares aleatórias de todos os blocos disponíveis para eles. Um nó pode também determinar quantos blocos inovadores podem transmitir a um vizinho, comparando a sua própria matriz de coeficientes de decodificação com a matriz do vizinho, ou pode simplesmente transmitir blocos codificados até que o vizinho receba o primeiro bloco não-inovador. O nó em seguida, deixa de transmitir a este vizinho até que ele receba mais blocos inovadores de outros nós. Os coeficientes de codificação são transmitidos junto com os blocos. Como os blocos geralmente têm um tamanho de centenas de *kilobytes*, essa sobrecarga é desprezível.

Em sistemas de distribuição P2P, a codificação de rede ajuda a minimizar o tempo de *download*. Um sistema P2P em grande escala, implica em uma maior complexidade para encontrar o escalonamento ideal de pacotes, particularmente, se os *hosts* participantes deste sistema possuem informações limitadas sobre a topologia da rede subjacente. Com a utilização da técnica de codificação de rede, o desempenho do sistema depende muito menos da topologia específica de uma rede *overlay* e do escalonamento. Devido à diversidade de blocos codificados, uma solução baseada na codificação de rede é muito mais robusta no caso de quando o servidor sai mais cedo, antes de todos os pares terminarem seu *download* ou quando há altas taxas de rotatividade de *nós* participantes, ou seja, os nós só participam por um período de tempo ou deixam imediatamente após terminarem seu *download*.

4.3 Vídeo sob Demanda, *Broadcast* ao Vivo de Mídia e Mensagem Instantânea

No caso do vídeo sob demanda, faz-se um *download* de um arquivo que é então dividido em pacotes. Eles devem chegar em ordem e serem decodificados em tempo real, após algum pequeno atraso. Neste caso, a codificação de rede pode ser aplicada ao quebrar o arquivo em gerações que podem ser baixadas de forma seqüencial. Para *broadcast* ao vivo, uma técnica similar pode ser aplicada. As mensagens instantâneas também são semelhantes ao *broadcast* ao vivo, entretanto, as taxas de *bits* típicas e de rajadas de mensagens de texto são baixas e sofre atrasos mais suaves. Porém, as mensagens instantâneas estão cada vez mais incluindo mensagens maiores, tais como imagens e clipes de áudio. A inundação, que geralmente é usada para mensagens instantâneas em redes P2P, é ineficaz quando se usa arquivos maiores.

Conforme Chou e Wu citados em referência [5], a codificação de rede pode ser aplicada em todos estes casos no intuito de melhorar a eficiência das redes *overlay*.

4.4 Armazenamento Distribuído

Um arquivo pode ser seguramente armazenado de forma distribuída. Para isso, pode usar uma coleção de nós não confiáveis e ter redundância suficiente. Conforme [5], usando um código *Reed-Solomon*, que pode ser visto em [14], um arquivo de tamanho M pode ser particionado em k pedaços. Cada pedaço possui o tamanho de M/k,

codificados em $n > k$ pedaços $y_1,\ldots,y_n$. Cada um desses pedaços codificados também possui o tamanho de M/k. Esses pedaços são distribuídos nos n nós de armazenamento. Dessa forma, o arquivo original pode ser recuperado ao longo de pelo menos k nós ativos, havendo leitura de um pedaço codificado de tamanho M/k de cada um dos k nós, e depois efetuando a decodificação. A manutenção deste nível de confiabilidade em face da desistência de um nó pode ser um desafio. Especificamente, sempre que um nó falhar de maneira permanente, ele deve ser substituído por outro nó. O novo nó deve regenerar um pedaço codificado de tamanho M/k lendo um pedaço codificado de tamanho M/k de cada um dos outros k nós, decodificando e recodificando. Isso resulta na comunicação essencialmente de todo o arquivo através da rede para cada nó falho. Quando o valor de n é grande, gera uma alta quantidade de comunicação referente à manutenção, já que os nós falham com freqüência.

Outras aplicações de codificação de rede para armazenamento distribuído podem ser encontradas nas referências [15] e [16].

4.5 Redes *Wireless*

Nesta subseção, são descritas algumas aplicações em redes sem fio nas quais se torna possível a implementação da codificação de rede, tais como tráfego bidirecional, redes *mesh*, codificação de rede na camada física, *broadcast*.

4.5.1 Tráfego Bidirecional

No Capítulo 2 foi mostrado como a codificação de rede pode melhorar o rendimento quando dois nós sem fio se comunicam através de uma estação base comum. Segundo Wu e*t al.* em [17], esta definição pode ser estendida para o caso de roteamento multi-saltos em uma rede sem fio (ou qualquer outra rede com camada de transmissão física), onde o tráfego entre os dois nós finais é bidirecional, e ambos os nós possuem um número semelhante de pacotes de troca. Em roteadores vizinhos, a transmissão é alternada até que todos os roteadores intermediários possuam pacotes armazenados em *buffer* de transmissão em ambos os sentidos de caminhos. Sempre que uma oportunidade de transmissão surge, um roteador combina dois pacotes, um para cada direção, com uma simples operação lógica ou-exclusivo (XOR) e transmite aos seus vizinhos. Ambos os roteadores que recebem este pacote codificado, já conhecem algum pacote que compõem o

pacote codificado, enquanto outro pacote é novo. Assim, cada transmissão permite que dois roteadores recebam um novo pacote, dobrando a capacidade do caminho.

4.5.2 Redes Mesh

Como descrito na Subseção 4.5.1, a redução do número de transmissões utilizando a codificação de rede pode ser estendida para vários saltos. De acordo com [5], se um nó sem fio *s* envia um fluxo de pacotes através de uma longa série de saltos até chegar ao nó sem fio *t*, então *t* pode enviar um fluxo de taxa igual de pacotes no sentido inverso ao de *s* sem ônus, ou seja, sem quaisquer transmissões adicionais sobre os saltos intermediários, proporcionando uma economia aproximada de 2:1. O autor desta técnica foi Wu *et al.* conforme a referência [17], e deu o nome de *piggybacking physical* usando Xor para combinar pacotes em um nó sem fio para decodificação local por seus vizinhos com a informação obtida por estar no raio de transmissão. Conforme a referência [18], Katti et al. estendeu o trabalho de [17], quando a informação é obtida por "ouvir indevidamente" o meio, ou seja, estar na área de alcance de transmissão do nó vizinho.

A Figura 4.8 ilustra um fluxo de pacotes *a* sendo transmitidos de $s1$ a $t1$ e outro fluxo de pacotes *b* sendo transmitidos de $s2$ a $t2$, ao longo de caminhos que se cruzam em um nó central sem fio chamado de A. Vizinhos do nó central escutam os pacotes *a* e *b* que são transmitidos para o nó central como indicado. O nó central codifica ambos os pacotes *a* e *b* em um único pacote $a+b$, que pode ser decodificado imediatamente por ambos vizinhos usando os pacotes "ouvidos" como informação, denominado codificação oportunista.

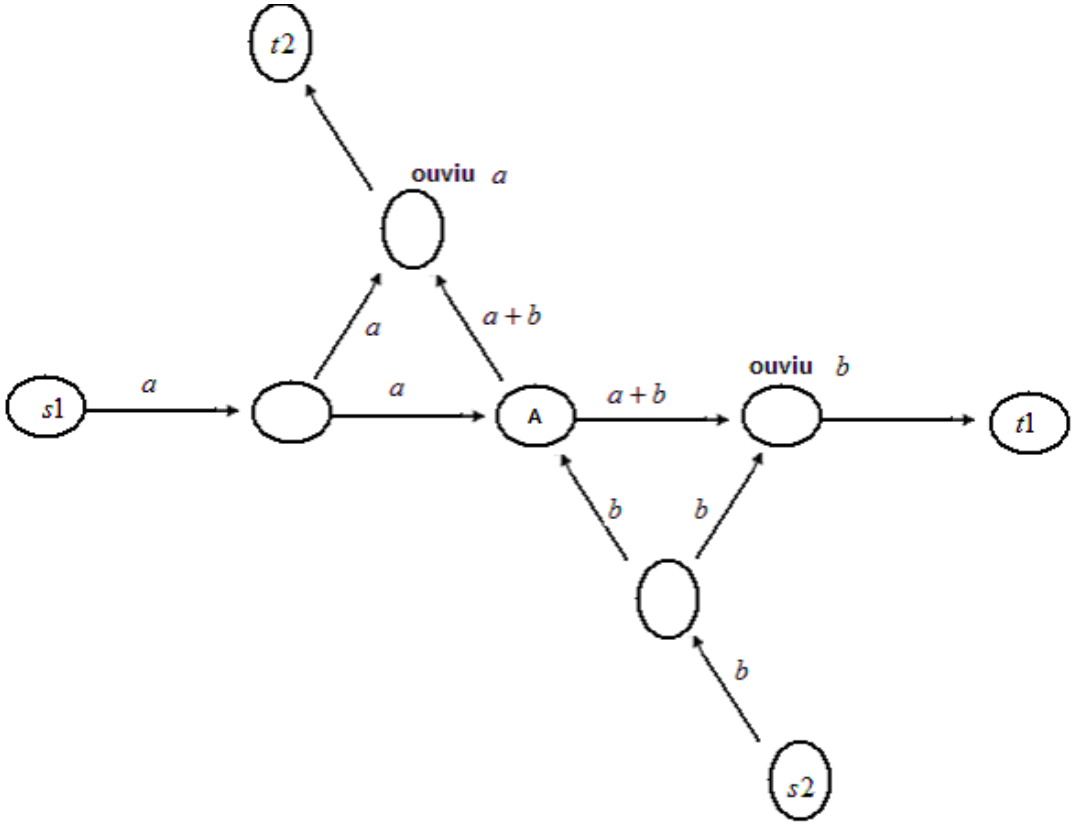


Figura 4.4: Codificação de rede oportunista em redes sem fio *mesh*.

Em [18] foi proposta a COPE (*Opportunistic Code*), em que cada nó mantém uma fila de pacotes não codificados $p_1, p_2, \ldots, p_n$ destinados para o próximo salto de receptores $r_1, r_2, \ldots, r_n$. Essa fila de pacotes não codificados é analisada e então é escolhida a melhor codificação para que os nós no próximo salto possam decodificar imediatamente a mistura resultante. Um receptor pode decodificar a mistura dos pacotes codificados se ele conhece ao menos um pacote não codificado. Por exemplo, a codificação dos pacotes $p_1 + p_3 + p_4$ se o nó r_1 conhece p_3 e p_4, o nó r_3 conhece p_1 e p_4, e o nó r_4 conhece p_1 e p_3. O nó r_2 possui quatro pacotes na fila, cujos próximos saltos estão listados na Figura 4.5(b). Cada vizinho de r_2 possui alguns pacotes armazenados como ilustrado na Figura 4.5(a). O nó r_2 pode tomar um número de decisões de codificação conforme ilustrado pela Figura 4.5(c), mas escolhe somente uma no intuito de maximizar o número de pacotes entregue em uma simples transmissão.

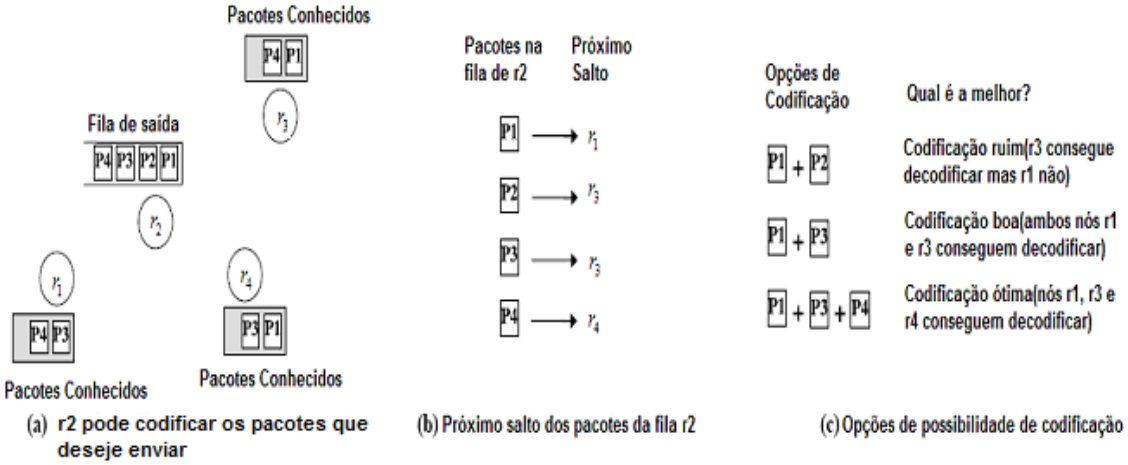


Figura 4.5: Exemplo de codificação oportunista.

Conforme as referências [18] e [19], além de aumentar a vazão, a codificação de rede oferece benefícios adicionais nas redes equalizando as taxas de transmissão e contribuindo assim, para protocolos da camada MAC e o TCP operarem de maneira mais eficiente. Na verdade, esses protocolos, na tentativa de serem justos, dividem igualmente a largura de banda entre os nós concorrentes. No entanto, para o exemplo da Figura 4.4, sem a codificação de rede o nó A precisa transmitir duas vezes mais rápido como os nós restantes para garantir o fluxo rápido de informações. Utilizando o TCP, o nó A não teria permissão para tal, e, portanto, seria um gargalo. A codificação de rede alivia este problema, já que agora o nó A precisa transmitir somente uma vez, o mesmo que o resto dos nós. A Figura 4.6 abaixo demonstra o desempenho de uma rede com a utilização do COPE e sem a utilização do COPE.

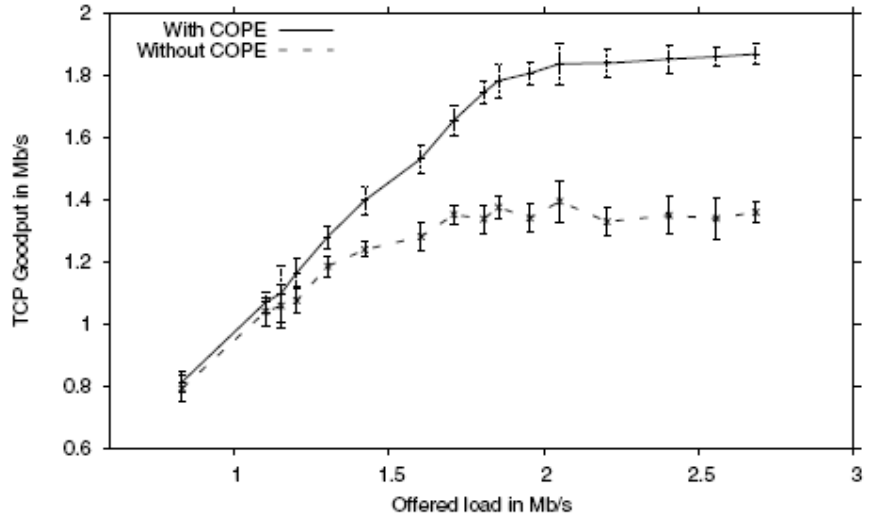


Figura 4.6: Desempenho da codificação oportunista – COPE.[18].

4.5.3 Codificação de Rede na Camada Física

Em redes de transmissão, é de conhecimento que os nós retransmissores não precisam necessariamente decodificar a informação original, mas simplesmente encaminhar aos nós receptores informação suficiente para ajudar na decodificação. Codificação de rede na camada física trata essa ideia ao observar que as transmissões simultâneas que causam interferência não necessariamente precisam ser tratadas como um ruído de tráfego para o outro: um transmissor poderia receber a superposição de sinais na camada física, e simplesmente transmitir essa superposição ao nó receptor.

O exemplo da Figura 4.7 ilustra esse ponto. Na rede de codificação, o transmissor combina linearmente os sinais recebidos $x1$ e $x2$ e transmite o sinal resultante $x1+x2$ para ambos os destinos. A combinação linear ocorre algebricamente, uma vez que ambos $x1$ e $x2$ são recebidos. Na codificação de rede da camada física, os nós *A* e *C* transmitem simultaneamente seus sinais para o receptor *B*. O nós *A* e *C* podem ser, por exemplo, antenas transmitindo seus sinais para um destino comum. O nó de retransmissão *B* observa na camada física a superposição de duas ondas eletromagnéticas coerentemente. Este sinal será retransmitido pelo transmissor, combinando os sinais, realizado pelo canal físico. Algoritmos para construção baseados neste exemplo indicam que esta abordagem pode de fato ser implementada na prática e aumentar a vazão alcançável em redes *wireless ad-hoc*. A abordagem de codificação de rede aproveita a capacidade natural de canais sem fio para transmissão visando proporcionar benefícios em termos de utilização de recursos.

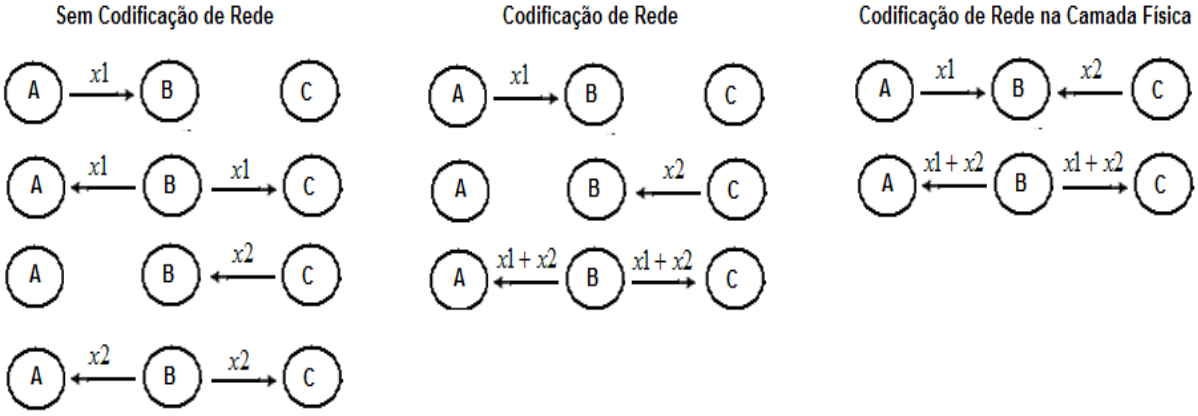


Figura 4.7: Exemplo de codificação de rede na camada física.

Segundo Fragouli em [19], é evidente que este exemplo simplificado abre uma série de questões teóricas e desafios de implementação. Questões teóricas abertas incluem: Como esta abordagem pode ser executada? O que é possível alcançar em taxas em redes arbitrárias? E como codificar levando em conta a interferência? Uma abordagem sistemática que surgiu recentemente faz promessas de progresso para responder a tais perguntas. Esta abordagem tenta levar em conta tanto à radiodifusão quanto as interferências, a fim de calcular taxas alcançáveis, bem como estratégias de codificação sobre redes sem fio. Ela utiliza canais determinísticos para modelar as interações entre os sinais na rede, e ignora o efeito do ruído. Mais detalhes sobre esta abordagem, podem ser encontrados em [19].

4.5.4. *Broadcast* Muitos para Muitos

De acordo com Widmer e*t al*. em [20] e Cai e*t al*. em [21], uma transmissão *broadcast* é usada para um número de propósitos em redes *ad-hoc* (por exemplo, descobrir a melhor rota) e podem ser aplicadas com muito mais eficiência com a codificação de rede. Um simples algoritmo distribuído para a codificação de rede aleatória reduz o número de transmissões por um fator de dois ou mais, levando a uma significativa economia de energia. De acordo com [13], nesse cenário, o amplo poder de transmissão direcionado traduz em uma redução no mesmo número de transmissões necessárias, o que permite uma economia interessante de energia. O gasto energético é também distribuído uniformemente entre os nós. Há uma maior flexibilidade na distribuição das necessidades de energia em relação aos algoritmos convencionais.

4.5.5 Desafios para a Codificação de Rede Sem Fio

De acordo com [19], a implantação da codificação de rede exigem recursos, tais como, sincronização e mecanismos de segurança e funcionalidades nos nós da rede, como as operações sobre campos finitos e capacidades de armazenamento. Por ser uma tecnologia nova, além de todos esses recursos citados acima, a codificação de rede requer também a concepção de novos protocolos e arquiteturas ou adaptação das infra-estruturas existentes de redes sem fio, de modo que as funcionalidades de codificação de rede estejam habilitadas. Deve-se refletir nos seguintes pontos: em qual camada referente ao modelo OSI (*Open Systems Interconnection*) a codificação de rede deve funcionar, que tipo de

conexões devem suportar, e que mecanismos devem ser criados ou alterados, formando-se assim, um conjunto de problemas de pesquisa a serem investigados em protocolos de rede e projetos de arquitetura.

Por exemplo, de acordo com referência [19], os protocolos da camada MAC, tais como 802.11, suportam apenas transmissões não-confiáveis. Isto é, quando os nós transmitem no modo *broadcast*, não há nenhum mecanismo para evitar a colisão, logo, não há o recebimento de reconhecimentos. Não está claro como adicionar esta funcionalidade na camada MAC, ou nas camadas superiores, e como implementar sem incorrer em uma enorme complexidade.

Outros desafios incluem o suporte às exigências de aplicações específicas. Por exemplo, aplicações em tempo real, tais como áudio e vídeo, possuem exigências estritas de Qualidade de Serviço - QoS em termos de vazão e atraso. Estes requisitos são muitas vezes conflitantes. Para alcançar a vazão ideal, com a codificação de rede, pode requerer que os nós da rede misturem n pacotes. Um exemplo de atraso proibitivo é o caso de um nó receptor, que pode precisar receber n pacotes para que ocorra a decodificação da informação fonte, para grandes valores de n.

Equilibrar esses requisitos pode requerer a abordagem *cross-layer*, que pode ser vista em [22].

Além disso, funcionalidades como escalonamento e roteamento precisam ser reconsideradas. Por exemplo, para uma rede que suporta múltiplas sessões *unicast*, não está claro como a informação, através de diferentes sessões, será misturada, e em que ordem às transmissões *broadcast* devem ser escalonadas.

5 Análise de Desempenho da Codificação de Rede para Tráfego HTTP

Com relação ao tráfego total que é gerado na Internet, o tráfego *web* é estimado como sendo mais de 70% e continua crescendo. Conforme Sanatos em [23], estudos demonstraram que o tráfego na Internet é autosimilar, isto é, apresenta estatísticas similares em diferentes escalas de tempo. Assim, modelos de simulação de fonte de tráfego *web* tem sido propostos com base em um extenso trabalho de campo, com diversas medidas em rede com tráfego real.

Nesta seção será analisado o desempenho da técnica de codificação de rede considerando fontes de tráfego. O tráfego utilizado é o HTTP (*Hypertext Transfer Protocol*), cujo modelo de fonte é proposto pelo 3GPP2 em [24]. Devido a sua utilização em redes IP, o protocolo da camada de transporte utilizado foi o TCP (*Transmission Control Protocol*), cujo modelo de tráfego é proposto por [23], uma versão simplificada do modelo de tráfego TCP de [24].

O cenário utilizado para análise é apresentado na Figura 5.1, onde tem-se um servidor *S* e dois clientes *C1* e *C2* e os pacotes são transmitidos via uma árvore *multicast* estabelecida. Essa árvore consiste das seguintes possibilidades de transmissão, $S \rightarrow A$, $A \rightarrow C1$, $A \rightarrow C$, $C \rightarrow D$, $D \rightarrow C2$ e $S \rightarrow B$, $B \rightarrow C2$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow C1$. O objetivo é analisar o desempenho da técnica de codificação de rede proposta em um nó de gargalo, como por exemplo, o nó *C* da Figura 5.1. Este desempenho será comparado ao desempenho deste mesmo nó C trabalhando no método *store-and-forward*.

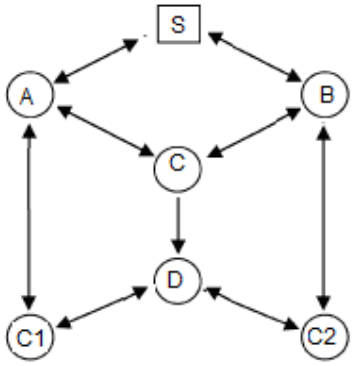


Figura 5.1: Cenário para análise de desempenho.

5.1 Modelo TCP

A versão do modelo TCP simplificado não considera o mecanismo de controle de congestionamento conhecido como partida lenta. O modelo TCP proposto por Santos em [23] foi adaptado para a análise de desempenho do nó de gargalo *C,* no enlace de $C \rightarrow D$. A Figura 5.2 ilustra esse modelo, no qual, uma vez estabelecida à conexão entre os clientes e o servidor com a utilização da *flag* SYN e do ACK, passando por diversos nós na rede, a transmissão dos dados é iniciada. Ao término desta transmissão, a conexão é encerrada com a utilização da *flag* FYN e do ACK. Na análise, será admitido que a conexão seja estabelecida e encerrada via nó de gargalo C.

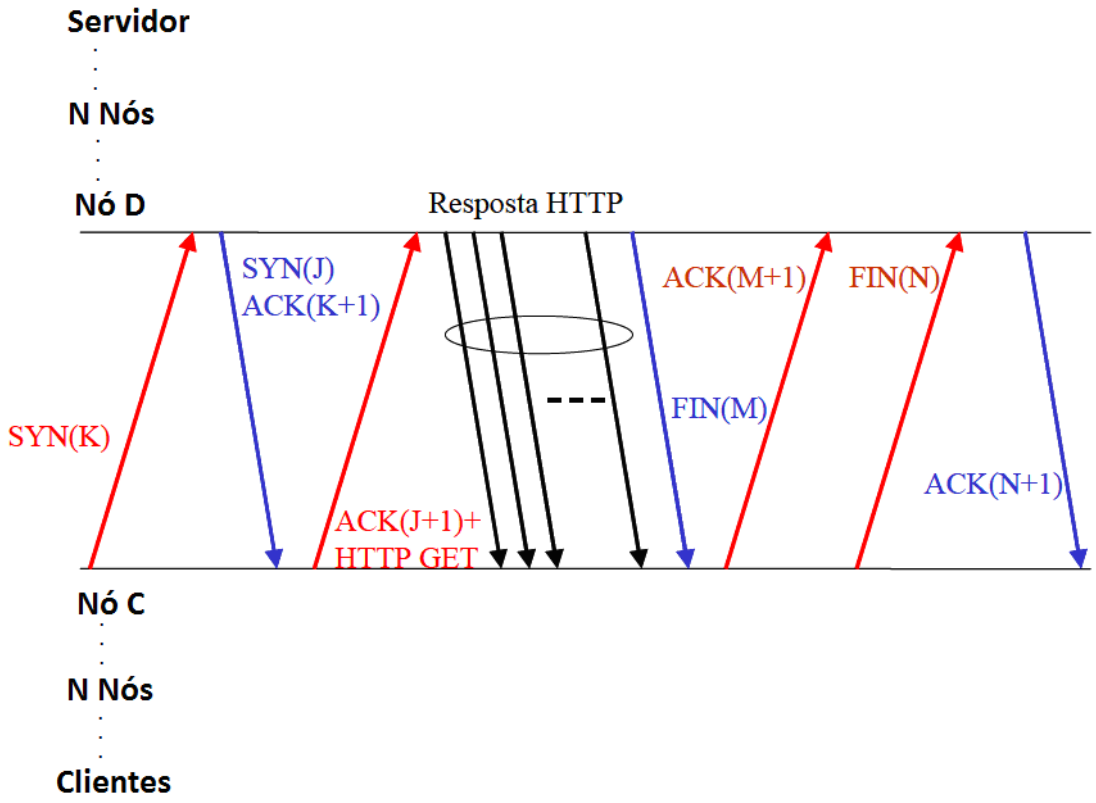


Figura 5.2: Estabelecimento e encerramento de uma conexão TCP.

O algoritmo proposto para o modelo TCP em [23] para o método *store-and-forward* é representado pela Figura 5.3 e pelo seguinte algoritmo de passo:

1. A variável S armazena o tamanho do objeto em *bytes*. O número de pacotes é dado por $N = [S/(MTU - 40)]$. Caso o número de pacotes não seja um número inteiro, ele assumirá o valor do próximo número inteiro superior.[1]

2. Os pacotes a serem transmitidos são enfileirados de acordo com a ordem de chegada. Sendo P a variável que armazena o número de pacotes a serem transmitidos, se $P == 0$, vá para o passo 6.

3. Aguarda até que um pacote do objeto seja transmitido.

4. Escalona a chegada do próximo pacote do objeto após um intervalo de tempo de propagação. Faz $P - 1$.

[1] 40 *bytes* de cada MTU são dedicados para códigos de redundância e cabeçalhos TCP e PPP.

5. Se $P > 0$, vá para o passo 3.

6. Fim.

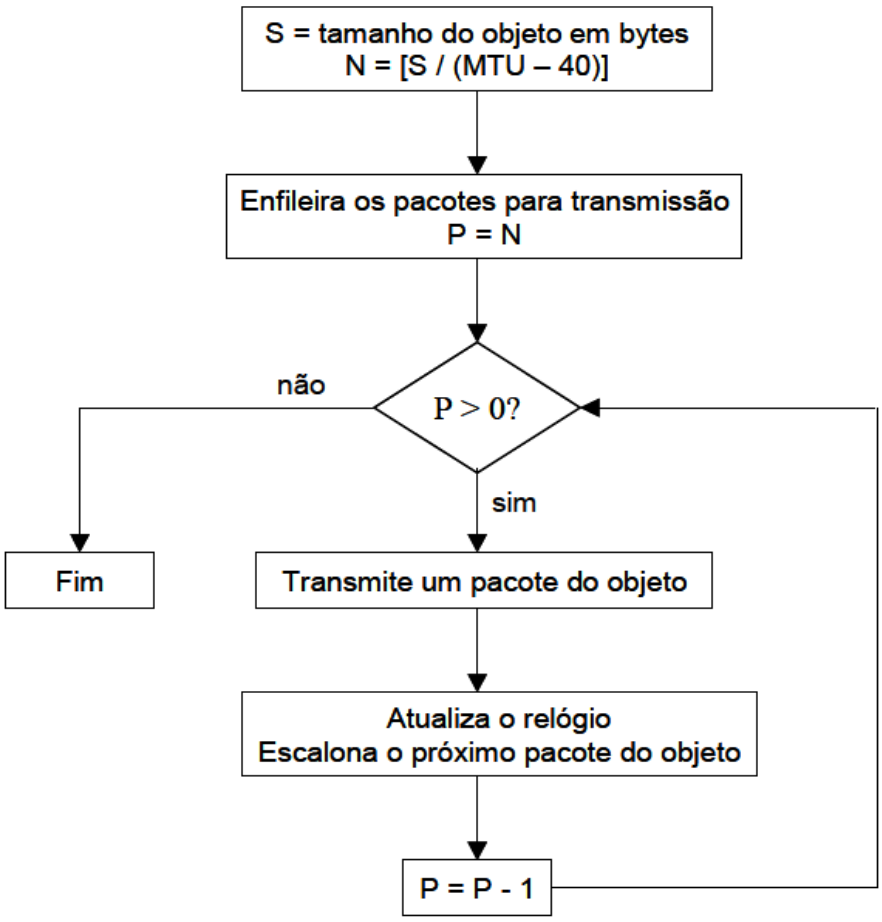


Figura 5.3: Processo de chegada dos pacotes no nó *C* sem codificação de rede [23].

Neste trabalho, é proposto um algoritmo para o método *store-code-forward*, ou seja, um algoritmo para o modelo TCP com codificação de rede. O processo de chegada dos pacotes no nó C pode ser representado pela Figura 5.4 e pelo seguinte algoritmo de passo:

1. A variável S armazena o tamanho do objeto em bytes. O número de pacotes é dado por $N = [S/(MTU - 90)]$. Caso o número de pacotes não seja um número inteiro, ele assumirá o valor do próximo número inteiro superior.[2]

2. Os pacotes a serem transmitidos são enfileirados de acordo com a ordem de chegada. Sendo P a variável que armazena o número de pacotes a serem transmitidos, se $P == 0$, vá para o passo 12.

3. Se $P == 1$, vá para passo 4, senão, vá para o passo 7.

4. Aguarda até que um pacote do objeto seja transmitido.

5. Escalona a chegada do próximo pacote do objeto após um intervalo de tempo de propagação. Faz $P - 1$.

6. Se $P > 0$, vá para o passo 3.

7. Codifica $PXorP - 1$.

8. Descarta os pacotes *P* e *P – 1*.

9. Transmite o pacote codificado.

10. Escalona a chegada dos próximos dois pacotes do objeto após um intervalo de tempo de propagação. Faz $P - 2$.

11. Se $P > 0$, vá para o passo 3.

12. Fim.

[2] Dos 90 *bytes* de controle, 40 *bytes* de cada MTU são dedicados para códigos de redundância e cabeçalhos TCP e PPP e os outros 50 *bytes* são utilizados para identificarem os pacotes codificados baseados na referência [9].

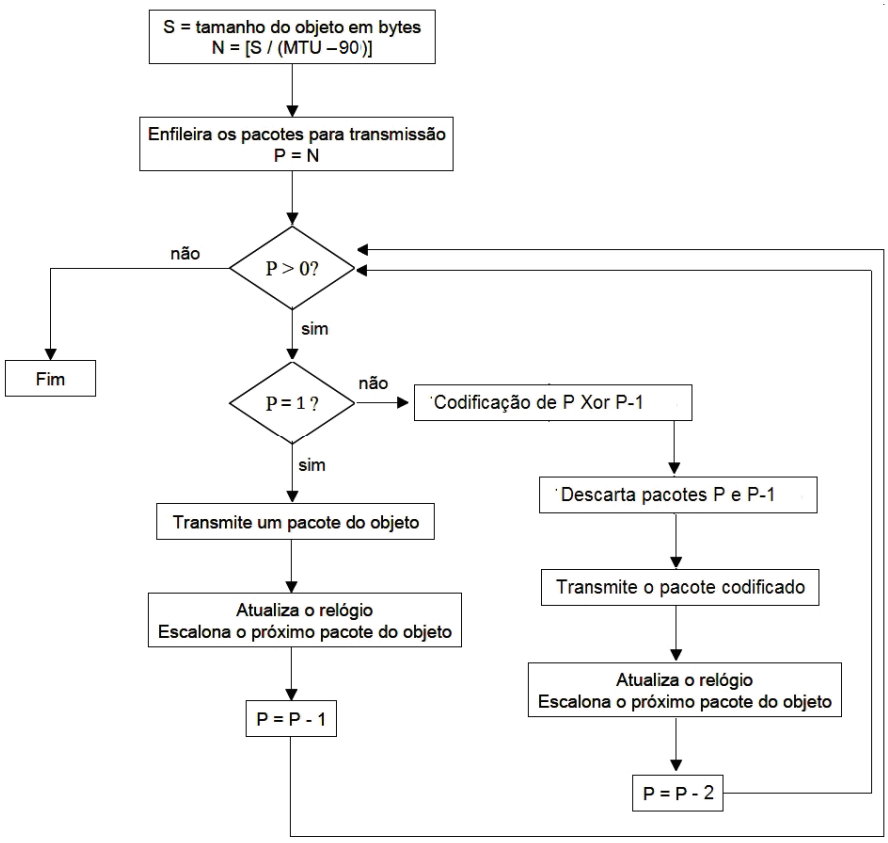


Figura 5.4: Processo de chegada dos pacotes no nó *C* com codificação de rede.

De acordo com 3GPP2 em [24], os tamanhos dos segmentos de dados TCP podem ser de 1500 *bytes* ou 576, ou seja, esse é o tamanho da unidade máxima de transferência MTU. Sem a utilização da técnica de codificação de rede, utilizando MTU de 1500 *bytes*, a estrutura deste segmento destina 1460 *bytes* para dados, e, utilizando MTU de 576 *bytes*, a estrutura deste segmento destina 536 *bytes* para dados. Os outros 40 *bytes* de cada MTU são dedicados para códigos de redundância e cabeçalhos TCP e PPP. Assim, os segmentos de dados TCP possuem um *overhead* percentual de 2,7% para segmentos com MTU de 1500 *bytes* e 7% para segmentos com MTU de 576 *bytes*.

Com a implementação da técnica de codificação de rede, utilizando MTU de 1500 *bytes*, a estrutura deste segmento destina 1410 *bytes* para dados, e, utilizando MTU de 576 *bytes*, a estrutura deste segmento destina 486 *bytes* para dados. Dos 90 *bytes*

restantes, 40 *bytes* de cada MTU são dedicados para códigos de redundância e cabeçalhos TCP e PPP e os outros 50 *bytes* são utilizados para identificarem os pacotes codificados baseados no Capítulo 4, item 4.1. Assim, os segmentos de dados TCP possuem um *overhead* percentual de 6% para segmentos com MTU de 1500 *bytes* e 15,63% para segmentos com MTU de 576 *bytes*.

5.2 Modelo HTTP

De acordo com [23], uma seção de tráfego *web* apresenta períodos de *downloads* de páginas e períodos de tempo de leitura destas páginas *web*, sendo que esta mesma seção pode haver n períodos de *downloads* e leituras de páginas *web*. Esta divisão em períodos do tráfego HTTP deve-se a interação homem máquina. A Figura 5.5 ilustra uma típica seção *web*.

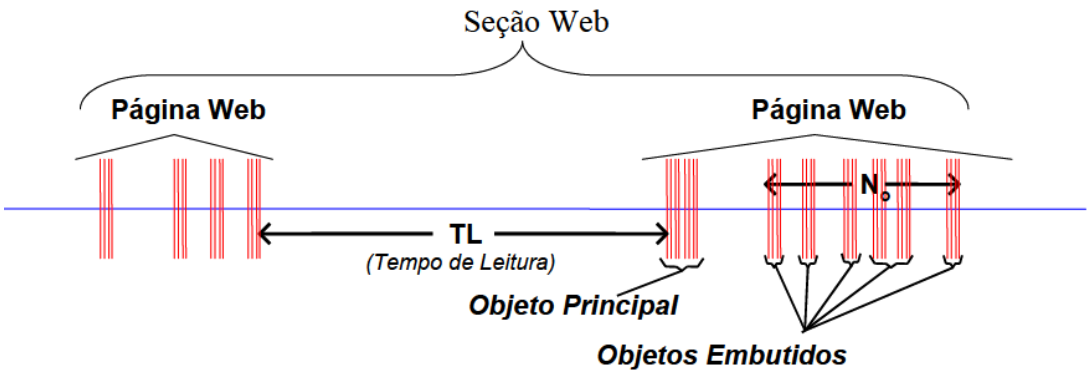


Figura 5.5: Seção de navegação *Web* [23].

Uma página *web* é constituída de vários objetos individualmente referenciados. Em uma transmissão de uma página *web*, primeiramente, usando uma requisição HTTP GET, o navegador solicita o objeto principal da página HTML (*Hyper Text Markup Language*). Esse objeto principal é responsável por determinar o *layout* da página *web*. Após o recebimento do *layout* da página, o navegador irá analisar o código HTML para referências adicionais para incorporar os objetos embutidos, como arquivos de imagem, tais como gráfico e figuras, botões estilizados, entre outros objetos possíveis em uma página *web*. O tempo gasto para determinar o *layout* da página *web* é chamado de tempo de *parsing*. Após esse tempo, o navegador faz a busca e o *download* dos objetos embutidos

que compõem a página *web* e o usuário começa a leitura da mesma. Esse tempo de busca e de *download* dos objetos principal e embutidos são representados pelos períodos *On*, enquanto o tempo de leitura destas páginas, onde ocorre a interação homem máquina, representam o período *Off*. A Figura 5.6 mostra o objeto principal e seus respectivos objetos embutidos da página *web* do jornal *Wall Street*.

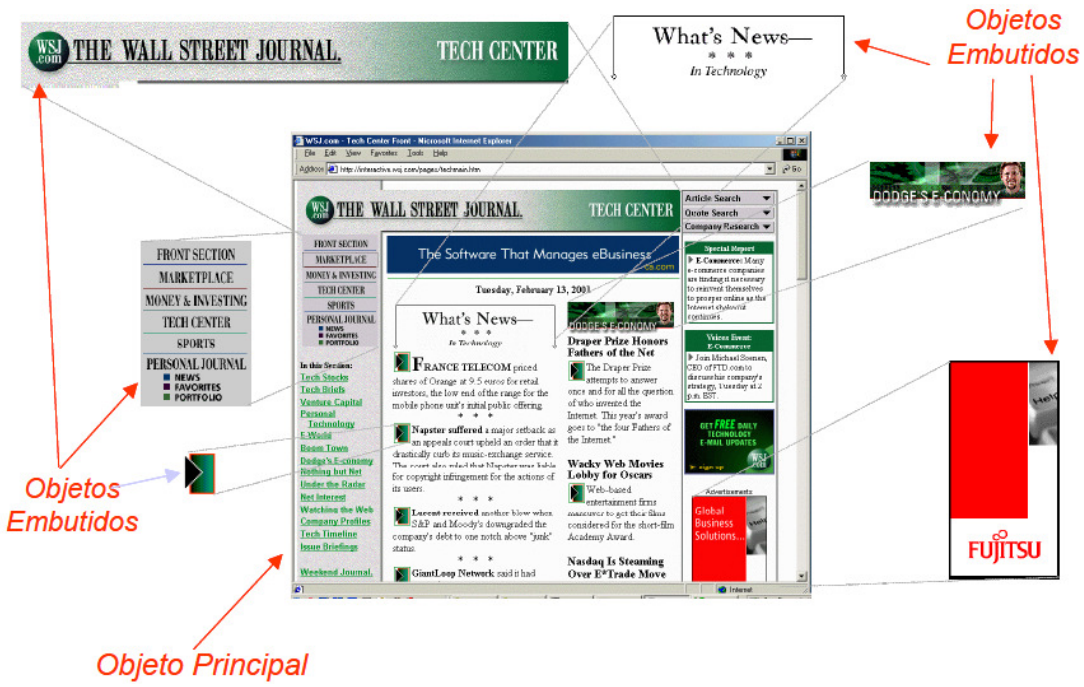


Figura 5.6: Uma típica página *web* e seu conteúdo [23].

As características de tráfego de pacotes dentro de uma seção dependem de como é feito o *download* pelos servidores *web* e navegadores. Duas versões do protocolo HTTP são usadas atualmente na maioria dos servidores e navegadores. Elas se diferem no modo como são usadas as conexões TCP para transferir os objetos principais e embutidos.

A versão HTTP/1.0, também conhecida como HTTP/1.0-*busrt mode transfer*, utiliza uma conexão TCP distinta para *download* de cada objeto principal e embutido. São permitidas até quatro conexões TCP paralelas na maioria dos navegadores.

Já na versão HTTP/1.1, também conhecida como HTTP/1.1-*persistent mode transfer*, são utilizadas conexões TCP persistentes em que os objetos são transmitidos serialmente por meio de uma conexão TCP.

5.2.1 Parâmetros do Modelo de tráfego HTTP

Conforme [24], as distribuições dos parâmetros para o modelo de tráfego de navegação na *web* foi determinada com base na pesquisa da literatura sobre as características de navegação na *web* em [25] e [26].

Como vimos anteriormente, uma página web é composta do objeto principal e vários embutidos. O tamanho do objeto principal segue a distribuição lognormal truncada, variando do mínimo de 100 *bytes* ao máximo de 2 *megabytes*, com média de 10710 *bytes*, com os parâmetros $\sigma = 1.37$ e $\mu = 8.35$. Os objetos embutidos por sua vez, também seguem a distribuição lognormal truncada, porém, o tamanho mínimo de 50 *bytes*, máximo de 2 *megabytes*, valor médio de 7758 *bytes*, com os parâmetros $\sigma = 2.36$ e $\mu = 8.35$.

O número de objetos embutidos por página segue a distribuição pareto truncada com média de 5.64 objetos e máximo de 53 objetos com os parâmetros $k = 2$, $\alpha = 1.1$ e $m = 55$.

O tempo de leitura é o tempo entre duas requisições de página *web*, distribuído exponencialmente com média de 30 segundos. O comportamento do tempo de leitura é diretamente ligada a interação homem máquina, por isso, não será considerado esse tempo para a análise de desempenho comparativa entre o método *store-and-forward* e *store-code-forward*, a análise será feita apenas considerando a interação máquina máquina após os clientes solicitarem a primeira página *web*.

E por último, o tempo *parsing*, tempo gasto para determinar o *layout* da página *web* após a transferência do objeto principal. Ele segue a distribuição exponencial com média de 0.13 segundos. A Tabela 5.1 resume os parâmetros do modelo de tráfego HTTP.

Na Tabela 5.1 são descritas todas as funções densidade probabilísticas usadas para obter o modelo de tráfego HTTP. De acordo com [23], para a geração de números aleatórios com as distribuições lognormal, pareto e exponencial, foi adotado neste estudo o método da transformação inversa, em que as distribuições podem ser invertidas analiticamente. Entretanto, para a geração dos números aleatórios com as distribuições

citadas acima, grande parte já estava implementada no *toolbox stat* da ferramenta de software MatLab, com exceção da distribuição Pareto Truncada obtida em [27].

Tabela 5.1: Parâmetros do modelo de tráfego HTTP [23].

Componente	Distribuição	Parâmetros	PDF
Tamanho do objeto principal	Lognormal Truncada	Média = 10710 bytes Desvio = 25032 bytes Mínimo = 100 bytes Máximo = 2 Mbytes	$f_X = \frac{1}{\sqrt{2\pi}\sigma x}\exp\left[\frac{-(\ln x-\mu)^2}{2\sigma^2}\right], x \geq$ $\sigma = 1.37, \mu = 8.35$
Tamanho do objeto embutido	Lognormal Truncada	Média = 7758 bytes Desvio = 126168 bytes Mínimo = 50 bytes Máximo = 2 Mbytes	$f_X = \frac{1}{\sqrt{2\pi}\sigma x}\exp\left[\frac{-(\ln x-\mu)^2}{2\sigma^2}\right], x \geq$ $\sigma = 2.36, \mu = 6.17$
Número de objetos embutidos por página	Pareto Truncada	Média = 5.64 Max. = 53	$f_X = \frac{\alpha k^{\alpha}}{x^{\alpha+1}}, k \leq x < m$ $f_X = \left(\frac{k}{m}\right)^{\alpha}, x = m$ $\alpha = 1.1, k = 2, m = 55$
Tempo de Leitura	Exponencial	Média = 30 seg	$f_X = \lambda e^{-\lambda x}, x \geq 0$ $\lambda = 0.033$
Tempo Parsing	Exponencial	Média = 0.13 seg	$f_X = \lambda e^{-\lambda x}, x \geq 0$ $\lambda = 7.69$

Conforme visto anteriormente, uma página *web* pode ser segmentada em pacotes de 576 bytes e 1500 bytes. Em [24], as estatísticas mostram uma distribuição de 24% para 576 bytes e 76% para 1500 bytes, bem como uma distribuição de 50% para cada um dos dois modos de *download* disponíveis, o HTTP/1.0 e HTTP/1.1. O processo de geração de tráfego *web* pode ser descrito pelo fluxograma da Figura 5.7. Para o pacote de confirmação ACK do cliente para o roteador, o tempo de propagação é considerado distribuído exponencialmente com média de 50ms.

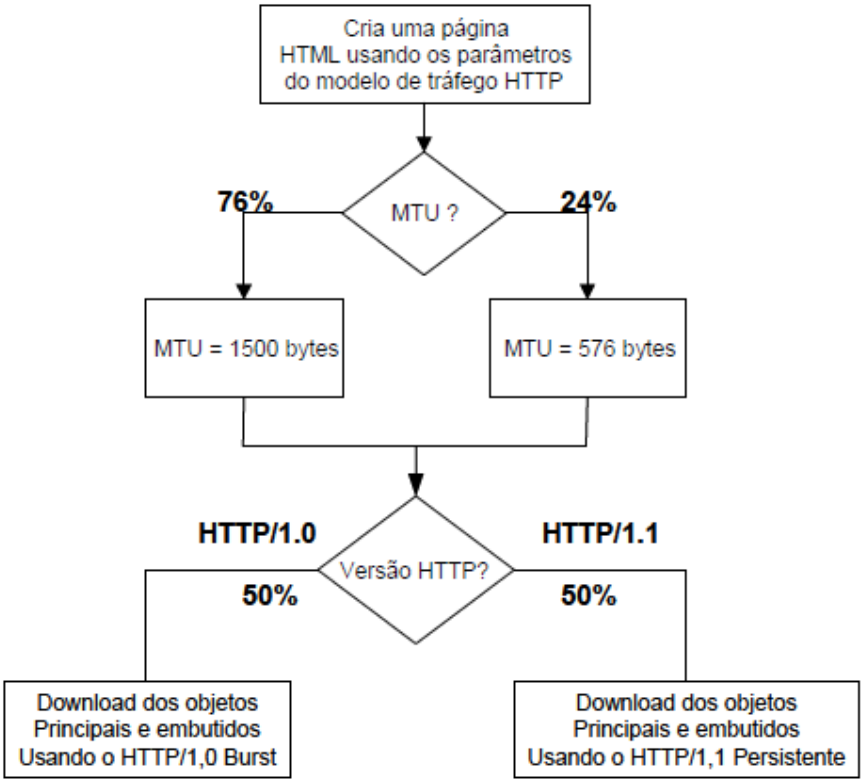


Figura 5.7: Modelagem de *download* de uma página *web* [23].

5.3 Simulação

Toda a simulação foi realizada utilizando o *software* MATLAB, um programa foi desenvolvido e escrito para simular a estrutura de funcionamento dos métodos *store-and-forward* e *store-code-forward*, para depois compará-los.

A análise comparativa entre os métodos de *store-and-forward* e *store-code-forward* enfatiza o ganho em relação ao tempo de transmissão de uma mesma informação e a quantidade de pacotes utilizados por esta mesma transmissão. Nesta simulação não foi considerado o valor da taxa de transferência, apenas o tempo de propagação. Em todas as simulações foi considerado um modelo de filas em que os pacotes de cada geração chegam sequencialmente em um enlace sem erros, e a codificação de rede é feita de dois em dois pacotes, e se por acaso sobrar um pacote na fila dessa mesma geração de tráfego HTTP, ele será transmitido normalmente. Este modelo de codificação de rede respeita o teorema do Fluxo Máximo Corte Mínimo.

A simulação foi feita em três cenários visando verificar o desempenho da codificação de rede para baixo, médio e alto volume de tráfego. Foi considerado baixo volume de tráfego, a geração de 1000 pacotes codificados. Médio volume de tráfego, a geração de 10000 pacotes codificados. E de alto volume de tráfego, a geração de 100000 pacotes codificados. Nesta simulação, a variável de saída deve atingir um determinado valor. Esse valor é calculado pela soma dos pacotes utilizados para a transmissão da informação via *multicast*, com a técnica de codificação de rede.

5.3.1 Cenário 1: 1000 pacotes

As Figuras 5.8 e 5.9 retratam a distribuição do tamanho dos pacotes utilizados no primeiro cenário para o método *store-and-forward* e o *store-code-forward*, respectivamente. Estes resultados foram obtidos através do algoritmo de criação de uma página HTML utilizando os parâmetros de tráfego HTTP, em termos de distribuição do tamanho dos pacotes. Como a intenção é analisar o desempenho do método *store-code-forward*, são analisados somente os pacotes que possuem informações do usuário, ou seja, MTU igual a 576 e 1500 *bytes*. Os demais pacotes com valores inferiores a 576 e 1500 *bytes* são pacotes de controle, ou seja, SYN, FYN e ACK. Na Figura 5.8 foram gerados aproximadamente 80% dos pacotes com tamanho de MTU igual a 1500 *bytes* e aproximadamente 20% igual a 576 *bytes*. Na Figura 5.9 foram gerados aproximadamente 78% dos pacotes com tamanho de MTU igual a 1500 *bytes* e aproximadamente 22% igual a 576 *bytes*.

A Figura 5.10 compara a quantidade de pacotes necessários para transmitir uma determinada quantidade de informação pelo tempo gasto em segundos. No método *store-code-forward* (representado pela sigla SCF) são necessários aproximadamente 1000 pacotes para a transmissão dos dados em aproximadamente 50 segundos. Já com a utilização do método *store-and-forward* (representado pela sigla SAF) são necessários aproximadamente 1700 pacotes para a transmissão dos dados em aproximadamente 85 segundos. Houve um ganho de 35 segundos com a utilização do método *store-code-forward*.

A Figura 5.11 retrata o tamanho total cumulativo de informação em *bytes* contida nos pacotes transmitidos pelo tempo gasto em segundos. O total de informação contida nos aproximadamente 1000 pacotes codificados (representado pela sigla SCF) é de $1.9x10^6$ bytes. Já no método *store-and-forward* (representado pela sigla SAF), são necessários aproximadamente 1700 pacotes para transmitir os mesmos $1.9x10^6$ bytes de informação do usuário.

Neste cenário, houve um melhor desempenho com relação ao atraso na rede do método *store-code-forward* comparado ao método *store-and-forward*. Os dois transmitem a mesma quantidade de informação, porém, o *store-code-forward* é 35 segundos mais rápido e utilizou 700 pacotes a menos.

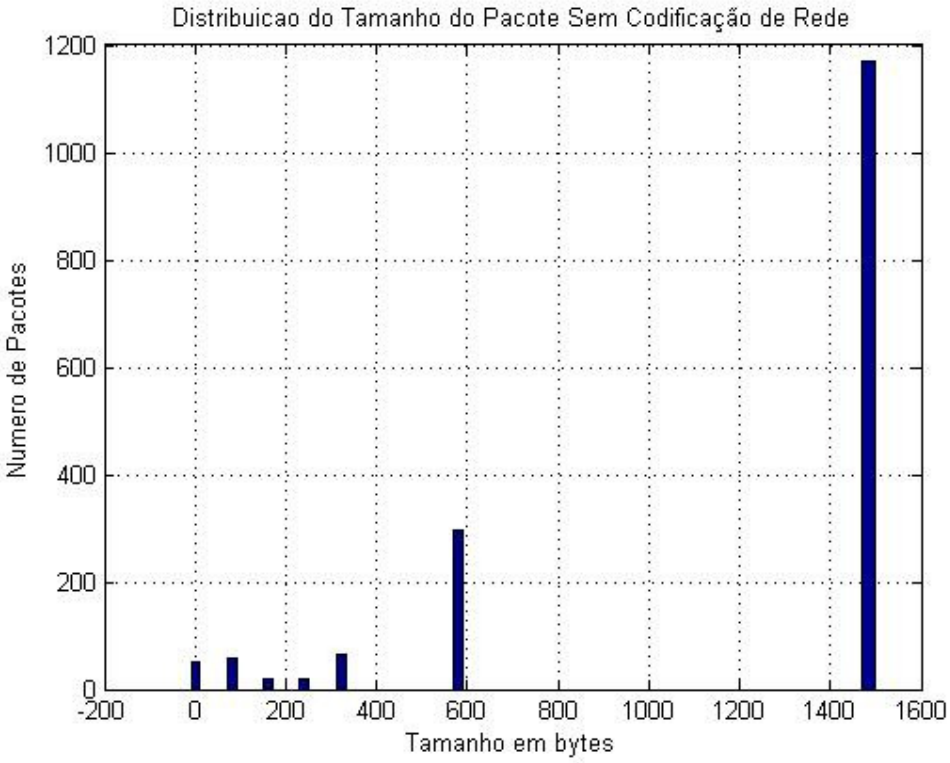


Figura 5.8: Distribuição do Tamanho do Pacote Sem Codificação de Rede.

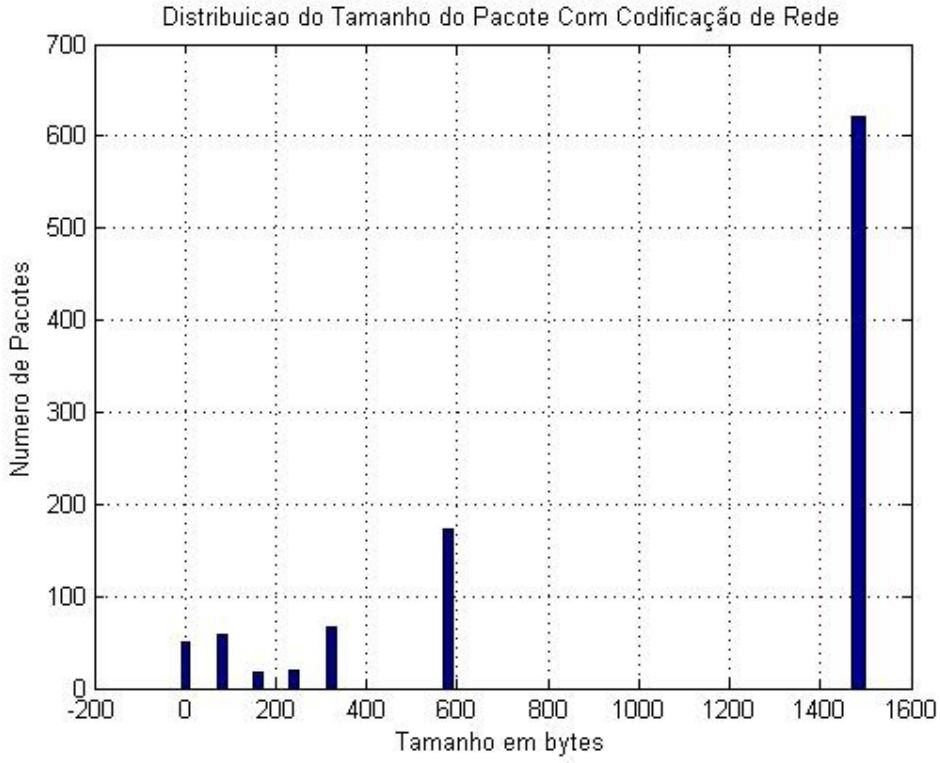


Figura 5.9: Distribuição do Tamanho do Pacote Com Codificação de Rede.

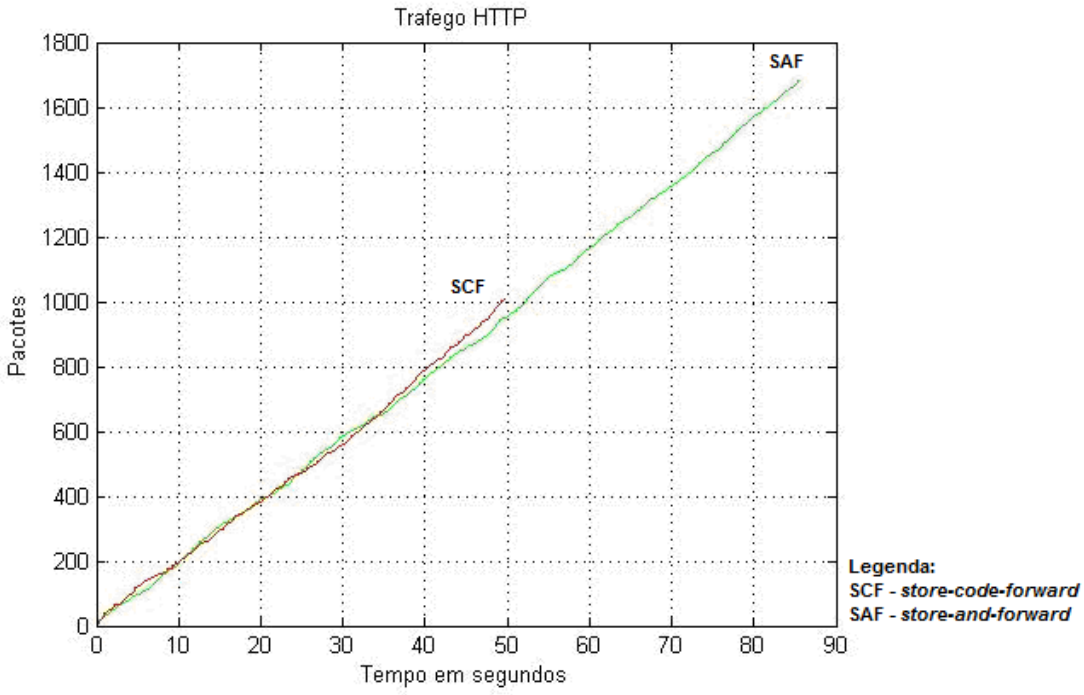
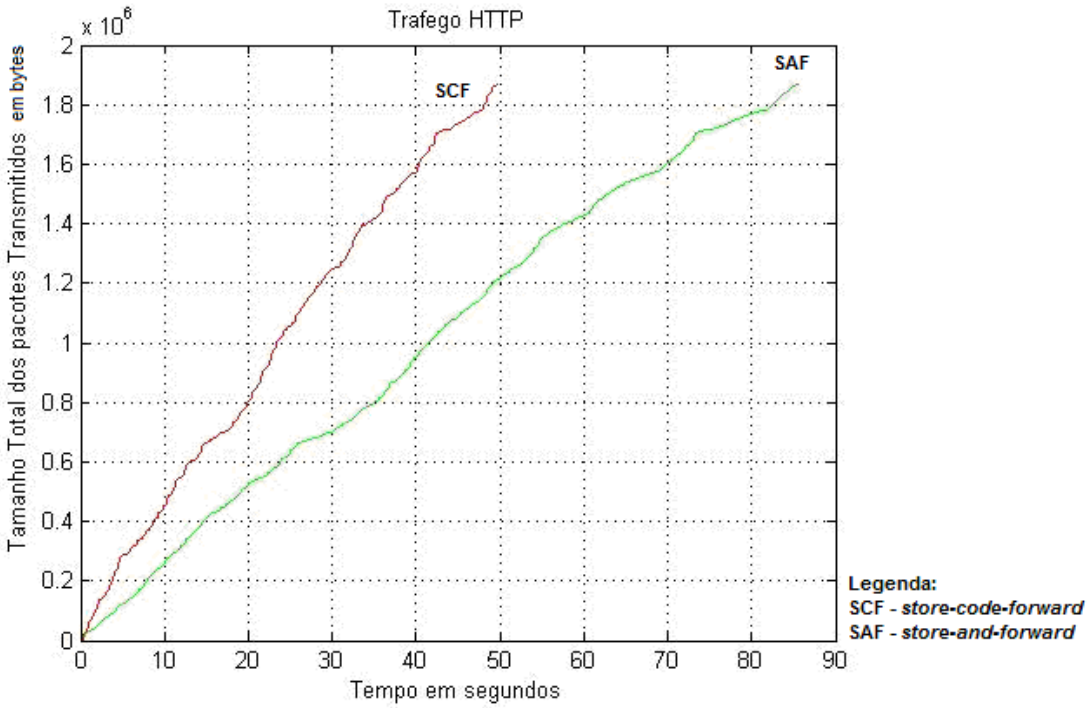


Figura 5.10: Quantidade Total de Pacotes x Tempo em segundos



.

Figura 5.11: Tamanho Total dos Pacotes Transmitidos x Tempo em segundos.

5.3.2 Cenário 2: 10000 pacotes

As Figuras 5.12 e 5.13 retratam a distribuição do tamanho dos pacotes utilizados no segundo cenário para o método *store-and-forward* e o *store-code-forward*, respectivamente. Estes resultados foram obtidos através do algoritmo de criação de uma página HTML utilizando os parâmetros de tráfego HTTP, em termos de distribuição do tamanho dos pacotes. Como a intenção é analisar o desempenho do método *store-code-forward*, são analisados somente os pacotes que possuem informações do usuário, ou seja, MTU igual a 576 e 1500 *bytes*. Os demais pacotes com valores inferiores a 576 e 1500 bytes são pacotes de controle, ou seja, SYN, FYN e ACK. Na Figura 5.12 foram gerados aproximadamente 73% dos pacotes com tamanho de MTU igual a 1500 bytes e aproximadamente 27% igual a 576 *bytes*. Na Figura 5.13 foram gerados aproximadamente 70% dos pacotes com tamanho de MTU igual a 1500 *bytes* e aproximadamente 30% igual a 576 *bytes*.

A Figura 5.14 compara a quantidade de pacotes necessários para transmitir uma determinada quantidade de informação pelo tempo gasto em segundos. No método *store-code-forward* (representado pela sigla SCF) são necessários aproximadamente 10000 pacotes para a transmissão dos dados em aproximadamente 500 segundos. Já com a utilização do método *store-and-forward* (representado pela sigla SAF) são necessários aproximadamente 17500 pacotes para a transmissão dos dados em aproximadamente 870 segundos. Houve um ganho de 370 segundos com a utilização do método *store-code-forward*.

A Figura 5.15 retrata o tamanho total cumulativo de informação em *bytes* contida nos pacotes transmitidos pelo tempo gasto em segundos. O total de informação contida nos aproximadamente 10000 pacotes codificados (representado pela sigla SCF) é de $1.9x10^7$ *bytes*. Já no método *store-and-forward* (representado pela sigla SAF), são necessários aproximadamente 17500 pacotes para transmitir os mesmos $1.9x10^7$ *bytes* de informação do usuário.

Neste cenário, houve um melhor desempenho com relação ao atraso na rede do método *store-code-forward* comparado ao método *store-and-forward*. Os dois transmitem a mesma quantidade de informação, porém, o *store-code-forward* é 370 segundos mais rápido e utilizou 7500 pacotes a menos.

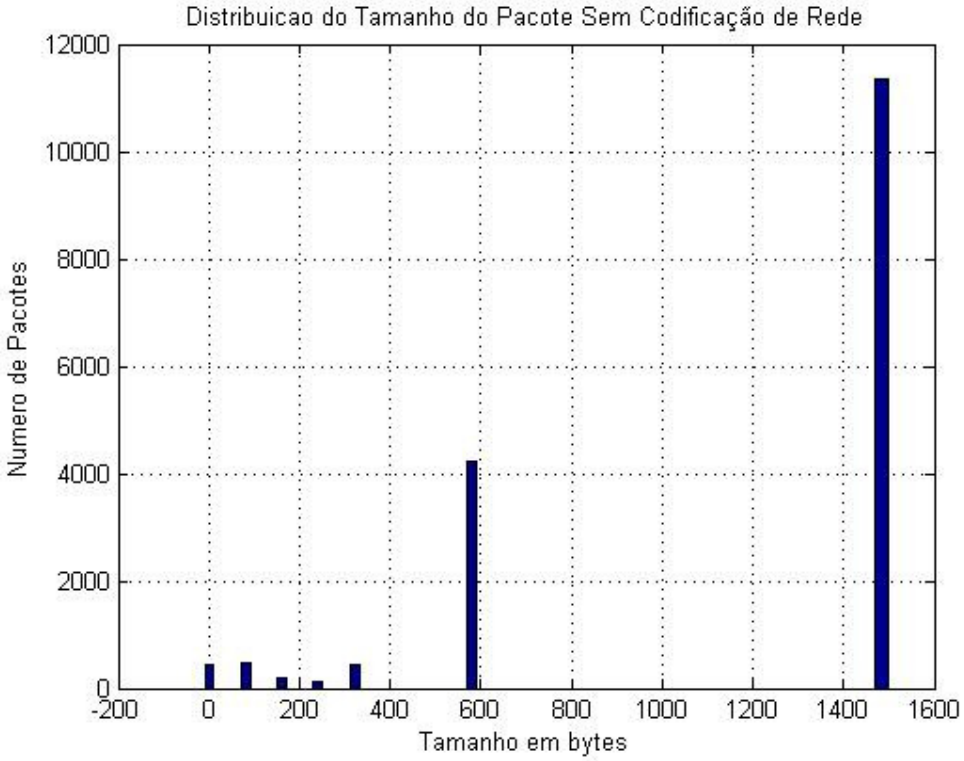


Figura 5.12: Distribuição do Tamanho do Pacote Sem Codificação de Rede.

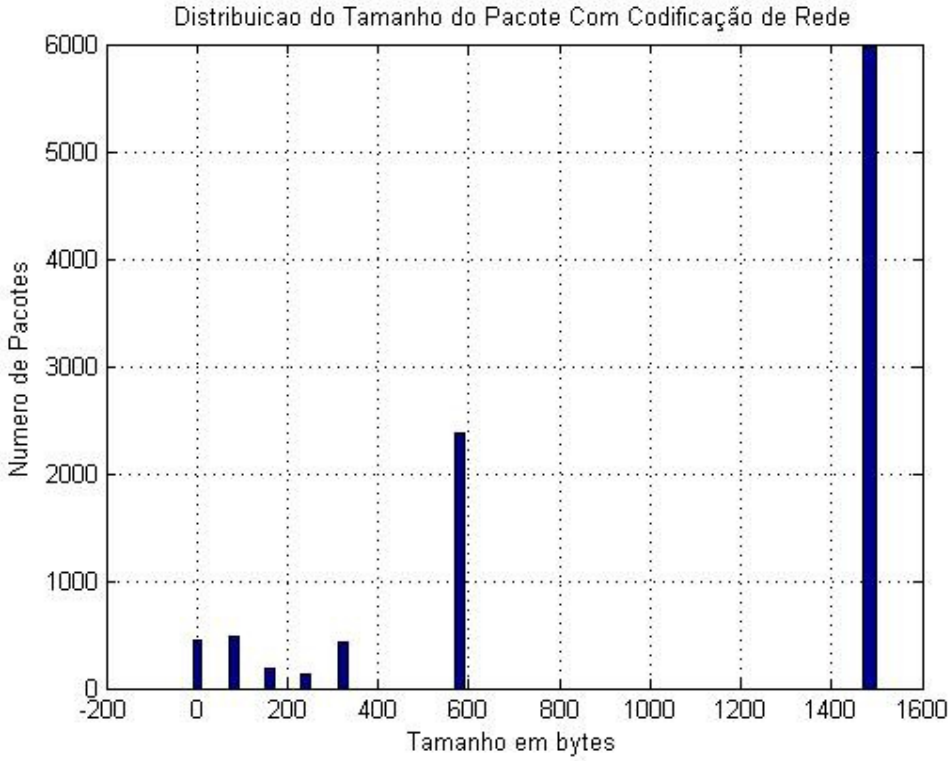


Figura 5.13: Distribuição do Tamanho do Pacote Com Codificação de Rede.

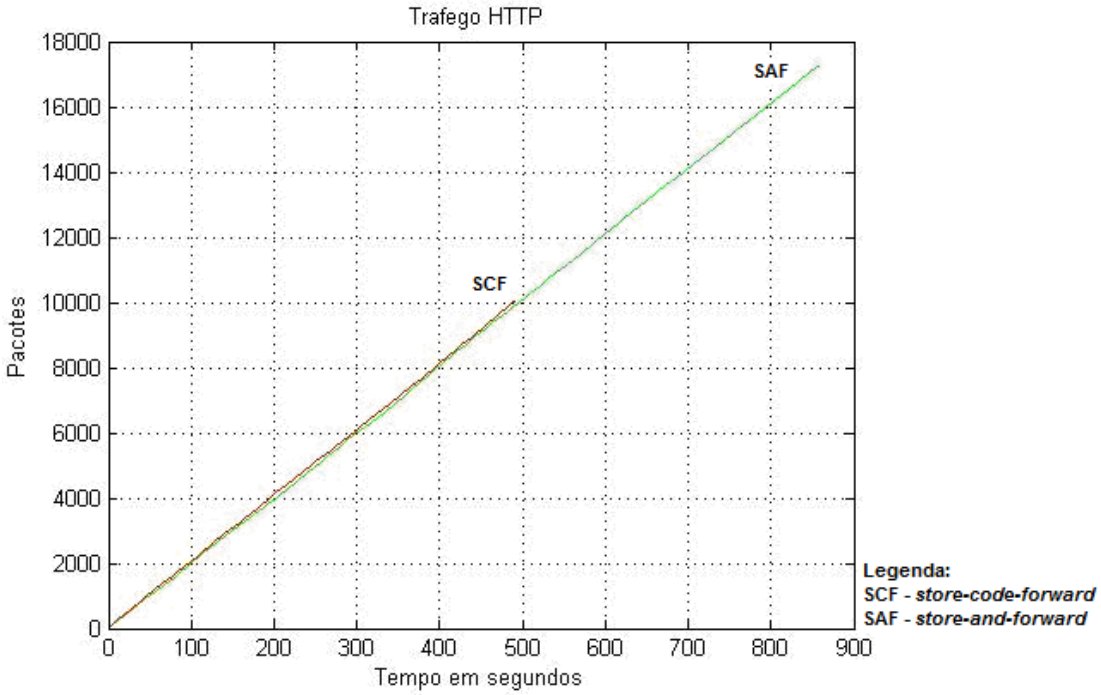


Figura 5.14: Quantidade Total de Pacotes x Tempo em segundos.

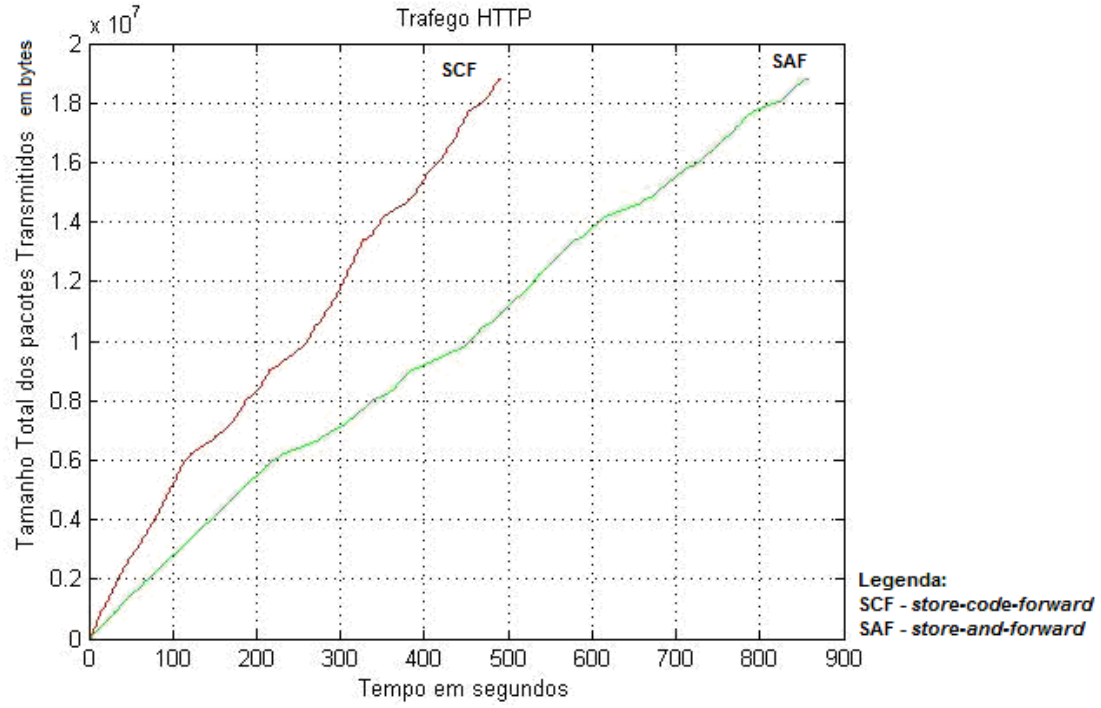


Figura 5.15: Tamanho Total dos Pacotes Transmitidos em *bytes* x Tempo em segundos.

5.3.3 Cenário 3: 100000 pacotes

As Figuras 5.16 e 5.17 retratam a distribuição do tamanho dos pacotes utilizados no primeiro cenário para o método *store-and-forward* e o *store-code-forward*, respectivamente. Estes resultados foram obtidos através do algoritmo de criação de uma página HTML utilizando os parâmetros de tráfego HTTP, em termos de distribuição do tamanho dos pacotes. Como a intenção é analisar o desempenho do método *store-code-forward*, são analisados somente os pacotes que possuem informações do usuário, ou seja, MTU igual a 576 e 1500 *bytes*. Os demais pacotes com valores inferiores a 576 e 1500 *bytes* são pacotes de controle, ou seja, SYN, FYN e ACK. Na Figura 5.16 foram gerados aproximadamente 80% dos pacotes com tamanho de MTU igual a 1500 *bytes* e aproximadamente 20% igual a 576 *bytes*. Na Figura 5.17 foram gerados aproximadamente 80% dos pacotes com tamanho de MTU igual a 1500 *bytes* e aproximadamente 20% igual a 576 *bytes*.

A Figura 5.18 compara a quantidade de pacotes necessários para transmitir uma determinada quantidade de informação pelo tempo gasto em segundos. No método *store-code-forward* (representado pela sigla SCF) são necessários aproximadamente 100000 pacotes para a transmissão dos dados em aproximadamente 5000 segundos. Já com a utilização do método *store-and-forward* (representado pela sigla SCF) são necessários aproximadamente 170000 pacotes para a transmissão dos dados em aproximadamente 8300 segundos. Houve um ganho de 3300 segundos com a utilização do método *store-code-forward*.

A Figura 5.19 retrata o tamanho total de informação em *bytes* contida nos pacotes transmitidos pelo tempo gasto em segundos. O total de informação contida nos aproximadamente 100000 pacotes codificados (representado pela sigla SCF) é de $1.9x10^8$ *bytes*. Já no método *store-and-forward* (representado pela sigla SCF), são necessários aproximadamente 170000 pacotes para transmitir os mesmos $1.9x10^8$ *bytes* de informação do usuário.

Neste cenário, houve um melhor desempenho com relação ao atraso na rede do método *store-code-forward* comparado ao método *store-and-forward*. Os dois transmitem a mesma quantidade de informação, porém, o *store-code-forward* é 3300 segundos mais rápido e utilizou 70000 pacotes a menos.

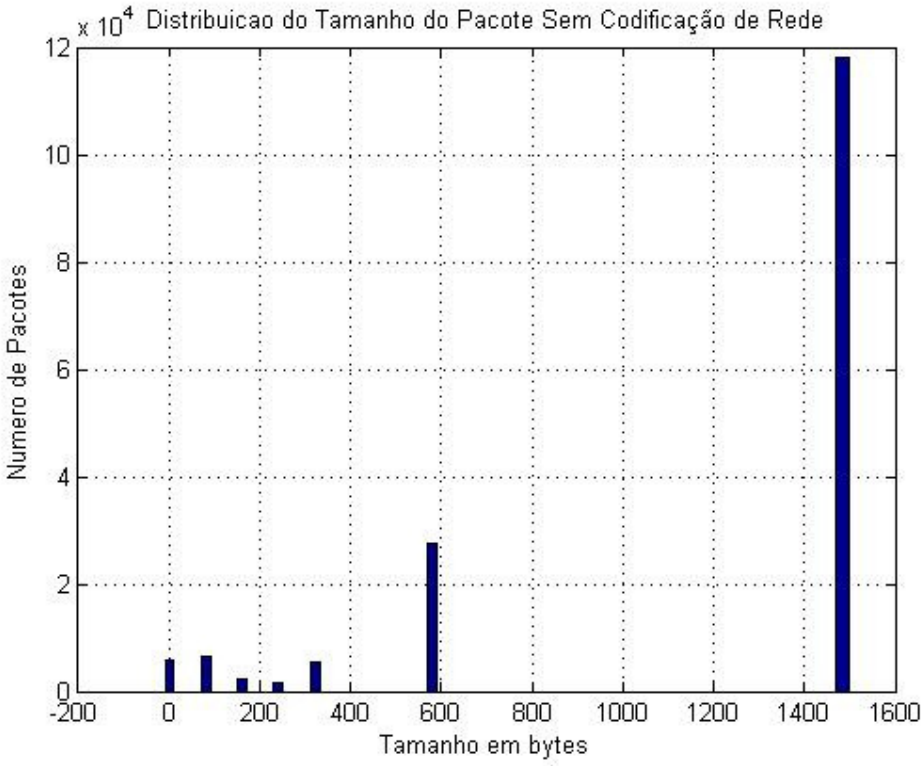


Figura 5.16: Distribuição do Tamanho do Pacote Sem Codificação de Rede.

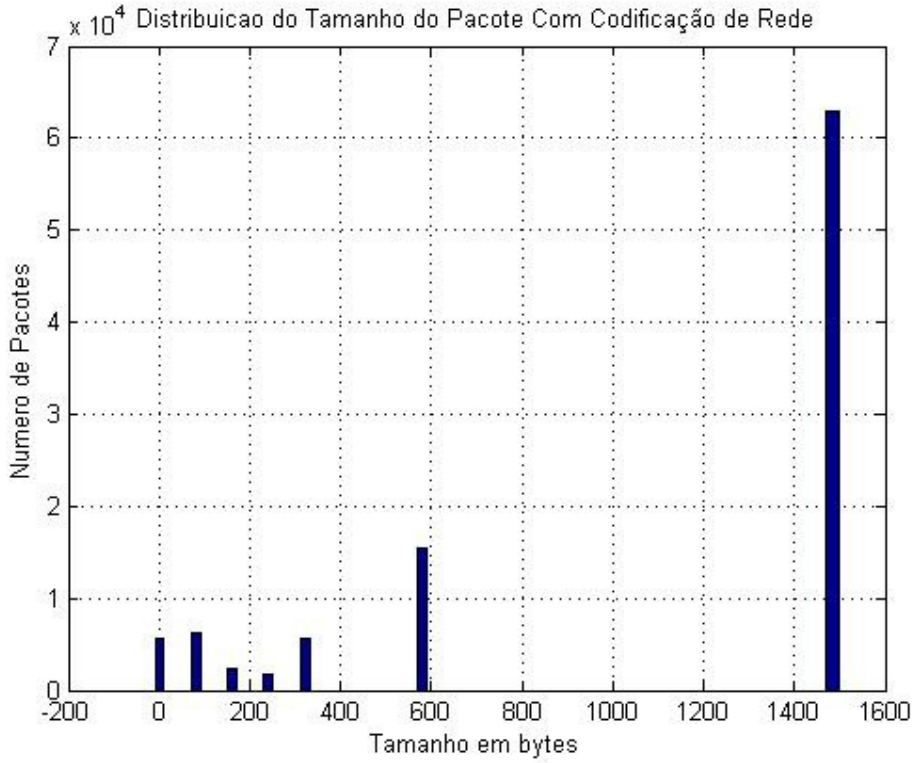


Figura 5.17: Distribuição do Tamanho do Pacote Com Codificação de Rede.

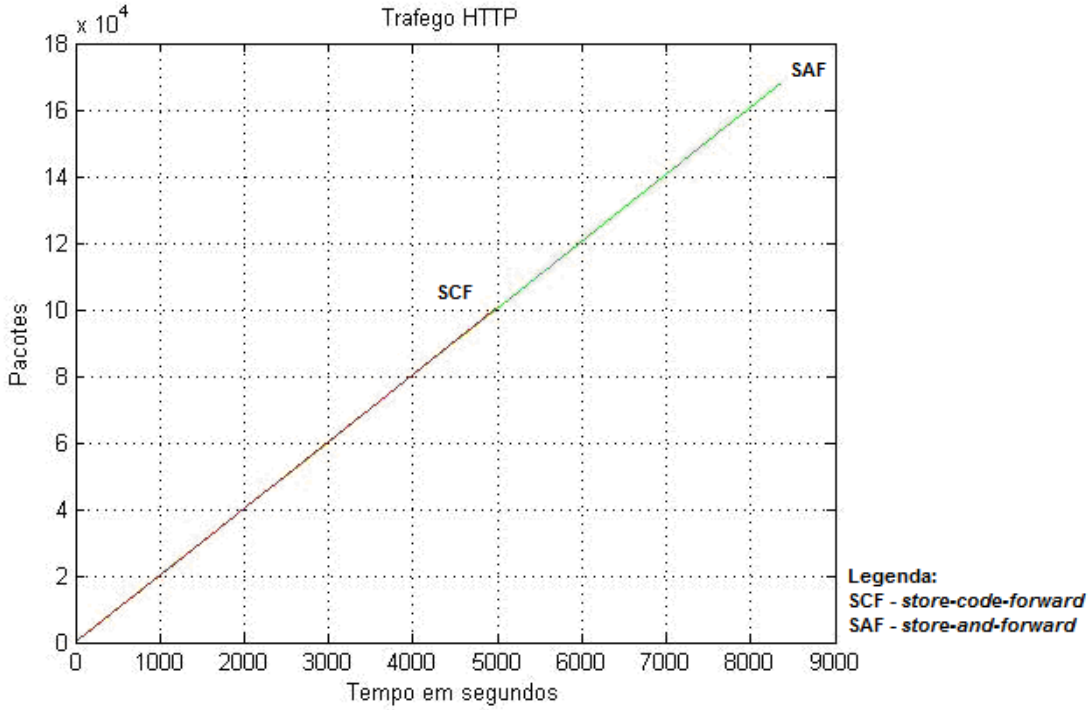


Figura 5.18: Quantidade Total de Pacotes x Tempo em segundos.

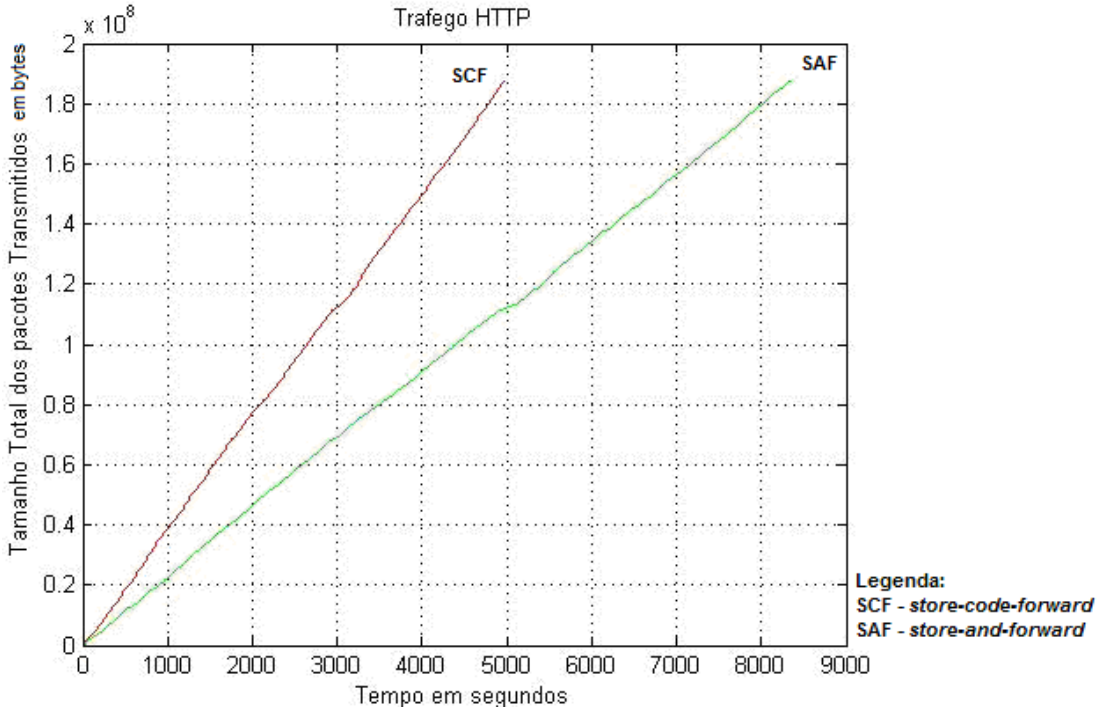


Figura 5.19: Tamanho Total dos Pacotes Transmitidos em *bytes* x Tempo em segundos.

6 Conclusão

A codificação de rede é entendida como uma técnica inovadora com relação ao tratamento da informação. Baseia-se em algoritmos nos quais o objetivo é melhorar a vazão e o desempenho da rede no intuito de atingir o fluxo máximo de informações transmitidas entre dois nós na rede. Para atingir tal melhora de desempenho, o nó precisa combinar os pacotes, ou seja, codificá-los. Esta codificação é feita pelo nó intermediário aplicando a operação ou-exclusivo bit-a-bit nos pacotes recebidos, não alterando, assim o seu tamanho original. O conteúdo codificado que é transmitido por este nó intermediário é a combinação linear dos pacotes recebidos pelo mesmo. No receptor, as mensagens são deduzidas à medida que um receptor tenha acesso a estes dados combinados.

Com esta nova técnica, os nós intermediários precisam ser modificados sobre como transmitir a informação. Além de enviar os dados, agora os nós podem processar o fluxo de informações independentes da entrada e codificá-los. Porém, novas funcionalidades adicionais de codificação de rede devem ser suportadas, novos protocolos ou adaptação dos protocolos existentes, requisitos adicionais de memória, operações lógicas do tipo ou-exclusivo, manter suas próprias informações armazenadas e resolver um sistema linear de equações.

As vantagens que a técnica de codificação de rede oferecem sobre o roteamento tradicional são a maximização da vazão, minimização do atraso e minimização de energia por *bit* conforme demonstrado no Capítulo 2. Estas vantagens são aplicáveis em redes reais, tanto na camada de rede, como os roteadores de um ISP e na camada de aplicação. Porém, devido a dificuldades citadas no Capítulo 4, à implementação da codificação de rede em roteadores IP na Internet é improvável que aconteça em um futuro próximo. Logo, o principal foco da codificação de rede atualmente são as redes *overlay*, devido a uma maior facilidade de sua implementação.

Nesta dissertação foram reunidas e demonstradas em sua teoria, vários tipos de técnicas de codificação de rede propostos para diversos cenários e aplicações em diferentes bibliografias, bem como a simulação de um algoritmo que implementa a técnica de

codificação de rede proposto neste trabalho em uma rede de tráfego HTTP utilizando o software MatLab.

Nas demonstrações de vários cenários e aplicações onde a codificação de rede foi aplicada teoricamente, ficaram comprovadas a melhoria de desempenho em relação ao método tradicional de roteamento. Alguns cenários e aplicações demonstrados, nos quais se percebeu melhoria de desempenho foi o *download* de arquivo, vídeo sob demanda, *broadcast* ao vivo de mídia, mensagem instantânea, armazenamento distribuído e redes *wireless*, tráfego bidirecional em redes wireless, redes *mesh*, *broadcast wireless* muitos para muitos, codificação de rede na camada física.

Por fim, foi proposto um modelo de codificação de rede que foi utilizado para realizar simulações no software MatLab. Nessa simulação, foi feita uma análise de desempenho comparativa do método *store-and-forward* e *store-code-forward* em um nó de gargalo intermediário. O tráfego considerado neste nó foi o HTTP. E, mais uma vez, agora via simulação, ficou comprovada a eficiência da codificação de rede e os ganhos que podem ser obtidos com relação ao tempo gasto para a transmissão de pacotes neste nó de gargalo.

Apesar dos benefícios da codificação de rede demonstrados nesta dissertação, sua implementação em roteadores das redes reais atuais está longe de ser praticada devido a dificuldade de integração com a infra-estrutura existente. As redes de comunicação devem incorporar tecnologias emergentes como a codificação de rede em uma determinada arquitetura de rede já existente, porém, esse processo costuma ser lento. Em redes *overlay*, a aplicação da codificação de rede é bem mais viável. Neste tipo de rede, os nós são programas em nível de aplicação sendo executados em computadores e as arestas são conexões em nível de transporte entre computadores. O fato da viabilidade da codificação de rede em redes *overlay* se deve ao fato de não precisar alterar os roteadores em uma rede de comunicação, já que toda a codificação e decodificação seriam realizadas no nível da aplicação.

Pensando-se nos trabalhos futuros, pode-se considerar o estudo sobre a perda pacotes e retransmissões em redes codificadas, análise de desempenho comparativa do método *store-and-forward* e *store-code-forward* utilizando diferentes tipos de tráfego, tais

como o *File Transfer Protocol* – FTP, *Simple Mail Transfer Protocol* – SMTP, voz, vídeo, aplicações em tempo real.

Referências Bibliográficas

[1] FRAGOULI, CHRISTINA; SOLJANINZ, EMINA. Network Coding Fundamentals, 2007, 133P.

[2] CAI, NING; LI, SHUO-YEN ROBERT; YEUNG, RAYMOND W; ZHANG, ZHEN. Network Coding Theory, 2005, 154p.

[3] YEUNG, R. W.; ZHANG, Z. "Distributed source coding for satellite communications," IEEE Trans. Inform. Theory, vol. IT-45, pp. 1111–1120, 1999.---211.

[4] AHLSWEDE,R.; CAI ,N.; LI ,S.-Y. R.; YEUNG, R. W. "Network information flow," IEEE Trans. Inform. Theory, vol. IT-46, pp. 1204–1216, 2000.---158

[5] CHOU, PHILIP A; WU, YUNNAN. Network Coding for the Internet and Wireless Networks. Junho de 2007.24p.

[6] PAPPAS, NIKOLAOS. Network Coding. Technical Report FORTH-ICS/TR-393– September 2007.100p.

[7] SAIREDDY, BALAJI. Structured Network Coding. Projeto submetido na Universidade do estado do Oregon para obtenção do título de mestrado.2009.54p.

[8] CAI, NING; LI, SHUO-YEN ROBERT; YEUNG, RAYMOND W. Linear Network Coding, IEEE Transactions on Information Teory, Vol. 49, N. º2, pp. 371 – 381, Fevereiro 2003.

[9] SPRIINTSON, ALEX. Network Coding and its Applications in Communication Networks. Texas A&M University, Texas USA. 31p.

[10] WU, Y.; CHOU, A.; JAIN, K. Network Coding for the Internet. In IEEE Communication Theory Workshop, Capri, Itália, 2004.

[11] GKANDSIDIS, C.; RODRIGUEZ, P. Network Coding for Large Scale Content Distribution. Miami. IEEE, March, 2005. INFOCOM.

[12] COHEN, B. Incentives Build Robustness in BitTorrent. Maio,2003. Documento disponível em http://bitconjurer.org/BitTorrent/bittorrentecon.pdf.

[13] FRAGOULI, CHRISTINA; BOUDEC, JEAN. ; WIDMER, JORG. Network Coding: An Instant Primer.LCA-Report-2005.7p

[14] HAYKIN, SIMON. Communications Systems. 4ed.

[15] DIMAKIS, A. G., ET AL. .Network Coding for Peer-to-Peer Storage. IEEE, May 2007. INFOCOM.

[16] ACEDANSKI, S., ET AL. How Good is Random Linear Coding Based Distributed Networked Storage? Abril, 2005. NetCod.

[17] WU, Y.; CHOU, P. A.; KUNG, S.-Y. Information exchange in wireless networks with network coding and physical-layer broadcast. Technical Report MSR-TR-2004-78, Microsoft Research, Ago. 2004.

[18] KATTI, SACHIN; RAHUL, HARIHARAN; HU, WENJUN; KATABI, DINA; MEDARD, MURIEL; CROWCROFT, JON. XORs in The Air: Pratical Wireless Network Coding. University of Cambridge.12p.

[19] FRAGOULI, CHRISTINA; SOLJANINZ, EMINA. Network Coding Applications, 2007, 135p.

[20] WIDMER, J.; FFRAGOULI, C.; LEBOUDEC, J. Y. Energy efficient broadcasting in wireless ad hoc networks.First Workshop on Network Coding, Mar, 2005.

[21] CAI, N; YEUNG, R. W. Secure Network Coding, International Symposium on Information theory (ISIT), 2002.

[22] MENDES, LUCAS DIAS PALHÃO. Algumas Análises de Soluções Cross-Layer para Redes TCP/IP Sem Fio. Dissertação de Mestrado. INATEL. 2009.157p.

[23] SANTOS, C. R., “Proposta e Análise de Desempenho de um Comutador de Pacotes com Enfileiramentos na Entrada e na Saída”. Tese de Doutorado. Unicamp. 2006. 83p.

[24] 3GPP2 Third Generation Partnership Project Two, *1xEV-DV Evaluation Methodology – Addendum (V6)*, Technical Report, WG5 Evaluation AHG, July 2001.

[25] CHOI, H. K. , LIMB, J. O. "A Behavioral Model of Web Traffic", Proceedings of the seventh International Conference on Network Protocols. 1999. Páginas 327-334.

[26] SMITH, F. D., CAMPOS, F. H., JEFFAY, K., Ott, D. "What TCP/IP Protocol Headers Can Tell Us About the Web". Cambridge.

[27] EVANS, M., HASTINGS, N., and PEACOCK, B., *Statistical Distributions*, JohnWiley and Sons, second edition, 1993.

Referências Bibliográficas Auxiliares

I. BANERJEE, SUMAN; RAMANATHAN, PAREMESWARAN; SUNDARAM, NIVEDITHA. Multirate Media Streaming Using Network Coding, 2007, 10p.

II. BRUCK, JEHOSHUA; LANGBERG, MICHAEL; SPRINTSON, ALEXANDER. The Encoding Complexity of Network Coding.20p.

III. Documento disponível em http://www.ifp.illinois.edu/~koetter/NWC/Course/ . Acessado em 20/10/2009.

IV. EFFROS, MICHELLE; HO, TRACEY; KARGER, DAVID R.; KOETTER, RALF; MÉDARD, MURIEL. The Benefits of Coding over Routing in a Randomized Setting.6p.

V. EFFROS, MICHELLE; HO, TRACEY; KOETTER, RALF ; MÉDARD, MURIEL. Network Coding for Correlated Sources. 6p.

VI. EFFROS, MICHELLE; HO, TRACEY; KARGER, DAVID R; KOETTER, RALF; LEONG,BEN; MÉDARD, MURIEL; SHI, JUN. Toward A Randon Operation of Networks.30p.

VII. HARVEY, NICHOLAS J.A.; KLEINBERG, ROBERT.; NAIR, CHANDRA.; WU, YUNNAN. A "Chicken & Egg" Network Coding Problem.5p.

VIII. HO, TRACEY; KOETTER, RALF; MÉDARD, MURIEL. A Coding View of Network Recovery and Management for Single Receiver Communications. Março de 2002. 8p.

IX. KUNG, SUN-YAN; WU, YUNNAN. Distributed Utility Maximization for Network Coding Based Multicasting: A Shortest Path Approach.13p.

X. HO, T.; LUN, D. *Network Coding: An Introduction*. Cambridge University Press, 2008.

XI. PADERBOM, DEREJE. *Network-Coded Cooperation in Wireless Networks: Theoretical analysis and Performance Evaluation*, 2010.161p.

XII. CHOU, PHILIP A; WU, YUNNAN; JAIN, KAMAL. *Pratical Network Codin*g. Dept. of Electrical engineering, Princeton University.10p.

XIII. HO, T. *A random Linear Network Coding Approach to Multicast*. IEEE Trans. Information Theory, Vol.52, 2006.

XIV. SANDERS, P.; EGNER, S.; TOLHUIZEN, L. *Polynomial Time Algorithms for Network information Flow*. Symp. Parallelism in Algorithms and Architectures (SPAA), 2003.

XV. SAIREDDY, BALAJI. Structured Network Coding. Projeto submetido na Universidade do estado do Oregon para obtenção do título de mestrado.2009.54p.

XVI. DIESTEL,R. Graph Theory, Springer - Verlag, 2000.

XVII. BOLLOBAS,B. Modern Graph Theory, Springer-Verlag, 2002.

XVIII. GADDAM, NISHANTH. Network Coding in Wireless Networks. Projeto submetido na Universidade do estado do Iowa para obtenção do título de mestrado.2009.56p.

XIX. GAJIC, BORISLAVA; RIIHIJÃRVI, JANNE; MAHÕNEN, PETRI. Performance evalution of Network Coding: Effects os Topology and Network Traffic for Linear na XOR Coding. RWTH Aachen University.9p.

XX. IRAJI, MOHAMMAD B.; AMERIMEHR, MOHAMMAD H.; ASHTIANI, FARID. A Queueing Model Wireless Tandem Network Coding. Advanced Communications research Institute – ACRI.2009.6p.

xxi. HO, TRACEY; LEONG, BEN; RALF, MÉDARD, Muriel. Distributed Asynchronous Algorithms for Multicast Network Coding. 6p.

xxii. JAFARI, MAHDI; KELLER, LORENZO; FRAGOULI, CHRISTINA; ARGYRAKI, KATERINA. Compressed Network Coding Vectors.EPFL, Laussane, Switzerland.5p.

xxiii. BARROS, JOÃO. Mixing Packets: Pros and Cos of Network Coding. International Symposium on Wireless Personal Multimedia Communications.2008.5p.

xxiv. CHOU, PHILIP A.; WU, YUNNAN ; JAIN, KAMAL. Pratical Network Coding. Dept. Of Electrical Engineering, Princeton University.

xxv. DIMAKIS, A. G.; PRABHAKARAN, V.; RAMCHANDRAN, K**.** Decentralized Erasure Codes for Distributed Networked Storage. IEEE, June 2006, Trans. Information Theory.

xxvi. PETROVIĆ, D.; RAMCHANDRAN, K; RABAEY, J. Overcoming Untuned Radios in Wireless Networks with Network Coding. Riva del Garda : s.n., April 2005. NetCod.

xxvii. WU, Y.; CHOU, A.; JAIN, K. Network Coding for the Internet. In IEEE Communication Theory Workshop, Capri, Itália, 2004.

xxviii. CAMPO, ADRIAN TAUSTE.; GRANT, ALEX. On Random Network Coding for Multicast. Fev. 2007.5p.

xxix. DIMAKIS, ALEXANDROS G.; PETROVIC, DRAGAN; RAMCHANDRAN, KANNAN. From Dumb Wireless Sensors to Smart Networks using netoqork Coding. Department of Electrical Engineering and Computer Science, University of California, Berkeley.2p.

xxx. SILVA, DANILO. Error Control for Network Coding.A thesis submitted in conformity with the requirements for the degree of doctor of Philosophy. Graduate Department of Eletrical and Computer Engineering, University of Toronto.2009.194p.

xxxi. BENTO, RUI DINIS MANGUEIRA. Redes de Distribuição de Conteúdos.:Integração com IPTV e Estudo sobre Codificação de Rede. Dissertação para obtenção do grau de mestre em Engenharia Informática e de Computadores.Set.2007.128p.