UNIVERSIDADE FEDERAL FLUMINENSE
CENTRO TECNOLÓGICO – ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES (PPGEET)

ANA CAROLINA FERREIRA BEAKLINI

Análise objetiva e subjetiva da implementação de mecanismos de correção de erros a nível de pacote sobre a qualidade do vídeo

NITERÓI/RJ
2018

ANA CAROLINA FERREIRA BEAKLINI

Análise objetiva e subjetiva da implementação de mecanismos de correção de erros a nível de pacote sobre a qualidade do vídeo

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

Orientador: Prof. Dr. Ricardo Campanha Carrano

NITERÓI/RJ
2018

Ficha catalográfica automática - SDC/BEE

B365a Beaklini, Ana Carolina Ferreira
Análise objetiva e subjetiva da implementação de mecanismos de correção de erros a nível de pacote sobre a qualidade do vídeo / Ana Carolina Ferreira Beaklini ; Ricardo Campanha Carrano, orientador. Niterói, 2018.
101 f. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2018.

DOI: http://dx.doi.org/10.22409/PPGEET.2018.m.11062803779

1. Comunicação de Dados. 2. Multimídia (Ciência da computação) . 3. Telemedicina. 4. Produção intelectual. I. Título II. Carrano,Ricardo Campanha , orientador. III. Universidade Federal Fluminense. Escola de Engenharia.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

ANA CAROLINA FERREIRA BEAKLINI

Análise objetiva e subjetiva da implementação de mecanismos de correção de erros a nível de pacote sobre a qualidade do vídeo

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

Aprovada em 30 de maio de 2018.

BANCA EXAMINADORA:

Prof. Dr. RICARDO CAMPANHA CARRANO - Orientador
Universidade Federal Fluminense – UFF

Prof. Dr. NATALIA CASTRO FERNANDES
Universidade Federal Fluminense – UFF

Prof. Dr. LISANDRO LOVISOLO
Universidade do Estado do Rio de Janeiro – UERJ

Dedico essa conquista aos meus pais Luiz Emilio e Carmen Regina, e ao meu noivo Fábio Portela pelo carinho e apoio irrestritos nessa trajetória, motivando-me a continuar e propiciando as condições necessárias para a realização deste trabalho, sem o qual não seria possível chegar até aqui e realizar este sonho.

AGRADECIMENTOS

À minha família, que sempre esteve presente em todos desafios e conquistas da minha vida, me apoiando incondicionalmente e me dando o suporte necessário para atingir uma formação sólida e uma carreira de sucesso. Ao meu pai Luiz Emilio Beaklini que sempre admirei e me inspirou na carreira de engenharia, a minha mãe Carmen Regina Ferreira Beaklini que sempre me apoiou, torceu e viabilizou o melhor para que eu conquistasse todos os meus sonhos. Ao meu noivo Fábio Affonso Portela, que mediante a todos os desafios sempre esteve ao meu lado me apoiando, me encorajando e sem ele nada disso seria possível.

Agradeço especialmente ao meu orientador Ricardo Carrano, pela amizade e por além de orientado, ter sempre contribuído pessoalmente e academicamente durante o período do mestrado, demonstrando dedicação e compreensão, os quais foram essenciais para a conclusão deste trabalho.

Agradeço a todos os membros do Programa de Pós-Graduação e do Laboratório MidiaCom, que sempre foram atenciosos e disponíveis para contribuir à minha trajetória.

Aos meus amigos da TV Globo, pelo apoio e incentivo constante a capacitação dos seus funcionários, flexibilizando muitas vezes minha carga horária no trabalho de modo a permitir que eu cumprisse os requisitos do curso de mestrado na Universidade Federal Fluminense, especialmente aos meus gestores Paulo Roberto Santos e Eduardo Ferreira por toda compreensão e apoio durante todo período do curso. Finalizando, gostaria de agradecer a todos que de alguma forma contribuíram para a minha dissertação.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSSÁRIO

EB – Exabytes
FEC – Forward Error Correction
QoS – Quality of Services
IP - Internet Protocol
FAPERJ - Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro
CSV - Consultório de Saúde Virtual
ARQ - Automatic repeat request
CSH - Centro de Saúde Holográfico
SMPTE - Society of Motion Picture & Television Engineers
Pro-MPEG CoP3 - Pro-MPEG Code of Practice #3
Pro-MPEG - Professional MPEG Forum
VPN - Virtual Private Network
CRASI - Centro de Recuperação e Assistência Social Integrada
GOP - Group of Picture
MPEG - Moving Picture Experts Group
UIT - União Internacional de Telecomunicações
VCEG Video Coding Experts Group
TCP - Transmission Control Protocol
UDP - User Data Protocol
RTP - Real time Protocol
XOR - ou-exclusivo
IETF - Internet Engineering Task Force
VBR - Variable Bit Rate
MB - Megabytes
GOP - Group of Pictures
SSIM - Structural Similarity Index
PSNR - Peak-Signal-to-Noise-Ratio
MSE - Mean Squared Error
RFCs - Requests for Comments
NetEM - Network Emulator

RESUMO

Os mecanismos de correção de erros são amplamente utilizados na maioria dos sistemas de comunicação digital para mitigar perdas de dados causadas por transmissões em canais ruidosos. Os codificadores e decodificadores implementam diferentes modelos de correção de erros. Estes podem ser acrescidos nos pacotes ou no fluxo de dados e buscam transmitir os dados de um ponto a outro de forma robusta. Esse trabalho se propõe a implementar e analisar o mecanismo de correção de erros Pro Mpeg CoP#3. Este mecanismo foi escolhido devido a sua simplicidade computacional, sendo uma promissora opção no contexto de atendimento médico remoto proposto pelo projeto Sistema de Saúde Holográfica da Universidade Federal Fluminense.

Aplicações baseadas em vídeo são consideradas críticas quando pensamos em banda de transmissão, na qualidade e no atraso. Por isso este trabalho busca entender as características do canal e parametrizar o mecanismo de correção de erros escolhido adicionando robustez suficiente para garantir uma comunicação de qualidade. Proporcionando ao médico especialista e ao paciente uma consulta à distância provendo uma experiência humanizada mais próxima de uma consulta real mesmo em áreas remotas.

Através do uso de um *hardware* comercial e do mecanismo de correção de erros é possível parametrizar o transmissor e o receptor para que atinjam uma configuração ótima para a aplicação desejada. Baseado nos resultados obtidos empiricamente. Avaliamos para perdas de 5% é possível recuperar totalmente os pacotes perdidos. O resultado obtido é promissor, pois oferece 100% de recuperação ao custo do acréscimo de redundância, o atraso medido pode ser considerado desprezível para a aplicação, porém o *overhead* chega a 50%.

Palavras-chave: FEC; Correção de Erros a nível de pacote; Fluxo multimídia, Redes IP, Pro-MPEG COP3 codes

ABSTRACT

Error correction mechanisms are widely used in most digital communication systems to mitigate data losses caused by transmissions in noisy channels. Encoders and decoders implement different error correction models. These can be added in the packets or in the data stream and seek to transmit the data from one point to another in a robust way. This work proposes to implement and analyze the error correction mechanism Pro Mpeg CoP # 3. This mechanism was chosen due to its computational simplicity, being a promising option in the context of remote medical care proposed by the Holographic Health System project of the Federal Fluminense University.

Video-based applications are considered critical when thinking about transmission bandwidth, quality and delay. Therefore, this work seeks to understand the characteristics of the channel and parameterize the chosen error correction mechanism adding enough robustness to ensure quality communication. Providing the specialist physician and the patient a remote consultation providing a humanized experience closer to a real consultation even in remote areas.

Through the use of commercial hardware and the error correction mechanism, it is possible to parameterize the transmitter and receiver to achieve optimum configuration for the desired application. Based on the results obtained empirically. We evaluate for losses of 5% it is possible to fully recover lost packets. The result obtained is promising because it offers 100% recovery at the cost of adding redundancy, the measured delay can be considered negligible for the application, but the overhead reaches 50%.

Key words: FEC, Packet-Level, Error recovery, Multimedia streaming, IP networks, Pro-MPEG COP3 codes

1. INTRODUÇÃO

A Internet vem crescendo massivamente pelo mundo desde a década de 90, tornando esta rede um dos principais meios de comunicação e acesso à informação da atualidade. Responsável por revolucionar a forma como a informação é consumida, todos os segmentos se adequaram para oferecer serviços que trafeguem por esse meio de comunicação: jornal, telefone, TV e rádio são exemplos de indústrias que se reinventaram para sobreviver a abrupta mudança no perfil dos usuários e nos meios de acesso [1].

A expansão do acesso à Internet alavanca oportunidades em diversos segmentos, viabilizando conceitos inovadores, dentre eles a utilização de sistemas de telepresença holográfica para atendimento médico remoto. Através desta tecnologia é possível levar a áreas carentes medicina especializada. É possível proporcionar ao médico uma visão detalhada do paciente mesmo à distância. Este modelo permite emular a experiência de uma consulta real, mitigando o desconforto experimentado pelos sistemas de vídeo conferência tradicionais [31].

De acordo com a pesquisa realizada pela empresa Estatista [2] em julho de 2017, referente a quantidade de usuários da Internet no mundo no período entre 2005 e 2018, o número de usuários cresceu abruptamente nos últimos 12 anos. Em 2017, a quantidade de usuários ultrapassou a marca de 50% da população mundial, atingindo um total de 3,59 bilhões de usuários [3], conforme mostra Figura 1.

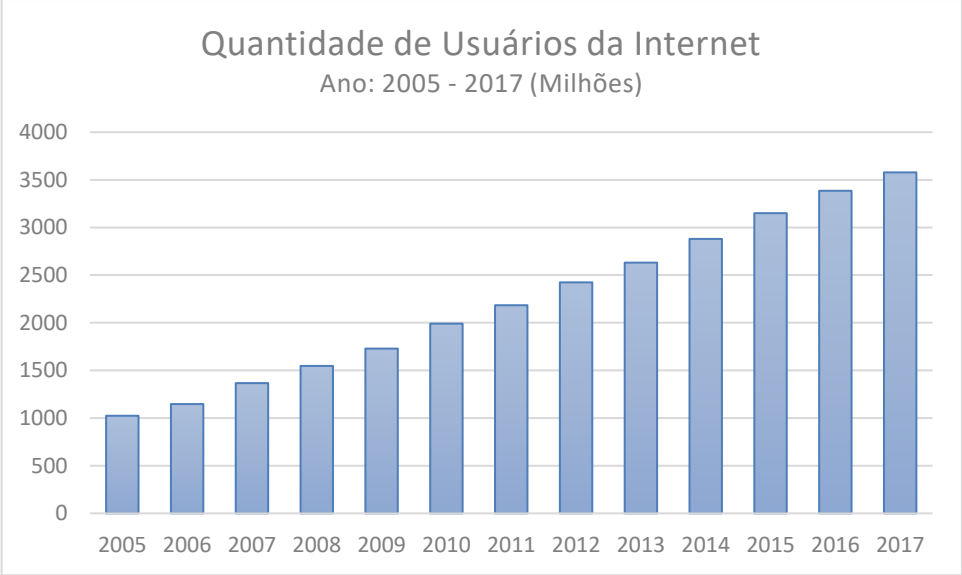


FIGURA 1: QUANTIDADE DE USUÁRIOS DA INTERNET. FONTE: STATISTA.

O acesso à Internet é quase universal quando analisamos países desenvolvidos, entretanto, o percentual de usuários cai consideravelmente quando falamos de países em desenvolvimento, principalmente devido à necessidade de investimento em infraestrutura. A Figura 2 ilustra a abrangência em percentual de indivíduos que possuem acesso à Internet por região, independente do meio de acesso [1].

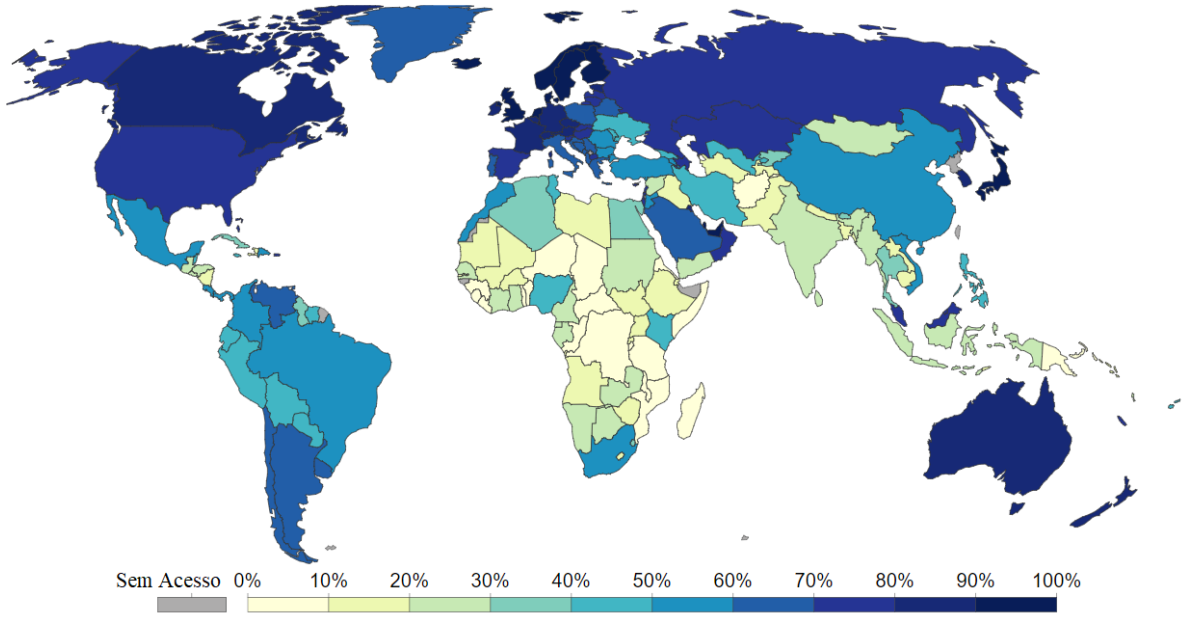


FIGURA 2: ABRANGÊNCIA MUNDIAL EM PERCENTUAL DE INDIVÍDUOS QUE POSSUEM ACESSO À INTERNET POR REGIÃO. FONTE: OUR WORLD IN DATA.

A expectativa é que esse crescimento não estagne nos próximos anos, estudos indicam um crescimento de 20% ao ano no que se refere a quantidade de dados trafegados em redes IP (*Internet Protocol*) pelo mundo. Este, em 2017, chegou a 122 EB (*Exabytes*) de conteúdo/mês e estimativas apontam o potencial de trafegar 186 EB por mês em 2021. A Tabela 1 ilustra os dados apresentados pela empresa Statista referindo-se à quantidade de dados trafegados no período de 2016 a 2021 [4].

TABELA 1 TRÁFEGO DE DADOS GLOBAL DE 2016 A 2021 (EM EB - EXABYTES). FONTE: STATISTA.

Tráfego de dados global de 2016 a 2021 (em EB - exabytes)		
Ano	**Dados/Mês**	**Dados/Ano**
2016	96	1153
2017*	122	1460
2018*	151	1811
2019*	186	2237
2020*	228	2741
2021*	278	3337

Tais predições otimistas se baseiam na expectativa de investimentos e avanços em infraestrutura que levarão o acesso a cada vez mais usuários com velocidades cada vez mais elevadas. Em 2021, a expectativa é de uma velocidade média global duas vezes maior que a de 2016, gerando assim um aumento no consumo de vídeo sobre IP que deve atingir 82% de todo o tráfego de dados mundial [5].

Apesar de toda expectativa otimista na elevação do tráfego de dados no mundo, os desafios em infraestrutura ainda são significativos, principalmente quando analisamos um país de proporções continentais como o Brasil, no qual existem abismos sociais e econômicos alarmantes. Exatamente onde a infraestrutura é precária, com baixa disponibilidade e capacidade de comunicação, estão as regiões com baixa densidade populacional onde, não raro, se concentram justamente a população com escassez de recursos e pouco acesso à serviços de qualidade, como a medicina especializada [31].

O Brasil como muitos outros países do mundo, chama atenção pelo seu lento desenvolvimento socioeconômico, crises constantes no sistema de saúde, agravados pelo alto índice de violência e acidentes. O país possui uma expectativa de vida de 73 anos que tende a aumentar para 82 anos em 2030. Doenças associadas ao envelhecimento da população são responsáveis por 72% das causas de mortes no país, com maior impacto na população mais vulnerável socioeconomicamente [31].

A prestação de serviços médicos é um problema no país, faltam profissionais especializados o que acarreta em ausência de tratamento adequado nessas regiões. Diante do cenário

explicitado, a Universidade Federal Fluminense, através do Núcleo de Estudos de Tecnologias Avançadas (Neta), em parceria com a Marinha do Brasil, originalmente financiado pela FAPERJ (Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro), desenvolveu um projeto com o objetivo de suplantar a necessidade de médicos especialistas através de um suporte remoto ao atendimento. O projeto consistiu em criar um consultório de saúde virtual (CSV) por meio de telepresença e holografia [31].. O sistema proposto garante ao médico remoto, a sensação de estar no mesmo local do paciente, sendo capaz de observar sua face, movimentos e reações.

A experiência deve ser fluida, garantir integridade do conteúdo transmitido e ter atraso desprezível, o que é um desafio dado que as regiões de interesse deste projeto são exatamente as que possuem também precário acesso a infraestrutura de comunicação de dados. A aplicação de vídeo conferência é sensível à perda de pacotes e a atrasos causados por mecanismos de retransmissão. Diante do cenário exposto, técnicas adicionais para acréscimo de redundância na transmissão se tornam atrativas e serão foco deste trabalho [31].

1.1. MOTIVAÇÃO

Existem vários mecanismos desenvolvidos para detecção e correção de erros. Estes mecanismos se propõem a corrigir erros que podem ser baseados na correção do nível dos bits de dados até a perda de pacotes completos.

O método usual de correção de erros através da retransmissão de pacote (ARQ - *Automatic repeat request*) é eficiente para aplicações que tenham alto compromisso com a integridade dos dados e ao mesmo tempo baixo compromisso com o atraso. No caso deste trabalho o ARQ não é aderente ao escopo da aplicação, por isso o este trabalho busca apresentar mecanismos que permitam garantir uma transmissão inteligível com mínimo atraso possível no meio de acesso disponibilizado para a região de destino.

A primeira fase do projeto consistiu em um piloto, a implantação de um Centro de Saúde Holográfico (CSH), local que é destinado aos médicos especialistas que darão o apoio remoto por meio da holografia. A infraestrutura do CSH deve ser confortável para atendimentos durante um dia inteiro, mitigando o desconforto causado por telas e excesso de luminosidade.

Após o sucesso do piloto realizado, a segunda fase do projeto contemplou a instalação de pilotos dos CSVs em diversos locais, buscando o aprimoramento tecnológico através dos seguintes requisitos:

- Definição dos recursos para o projeto de implementação dos CSVs (Consultório de Saúde Virtual) garantindo baixo custo e complexidade de instalação;
- Humanização do ambiente do CSV garantindo uma experiência agradável ao paciente;
- Automatização e simplicidade do controle de som, áudio e iluminação no CSV.
- Simplicidade e disponibilidade nas implementações através do uso de soluções baseadas em software;
- Uso de soluções baseadas em software livre, para redução de custos viabilizando a implantação do sistema por todo o país;
- Imagem com qualidade e baixa taxa de modo a mitigar os problemas de infraestrutura de rede nos locais mais remotos do Brasil;
- Uso de soluções complementares que garantam integridade e resiliência da comunicação.
- Criação de um ambiente de transmissão seguro e integrado com sistemas hospitalares, para que a troca de prontuários e exames seja transparente durante a consulta; [31]

1.2. OBJETIVO

O objetivo deste trabalho é estudar técnicas de correção de erro desenvolvidas na camada de aplicação e implementar o mecanismo de correção de erros (FEC – *Forward Error Correction*) em *streaming* de vídeo utilizando a técnica de ou exclusivo definida no padrão 2022-1 do SMPTE (*Society of Motion Picture & Television Engineers*) [31] e descrita pela recomendação Pro-MPEG CoP3 (*Pro-MPEG Code of Practice #3*) no fórum Pro-MPEG (*Professional MPEG Forum*) [23]. O objetivo é propor um mecanismo viável, que melhore a experiência das consultas remotas realizadas através dos sistemas de telepresença holográfica, em condições desfavoráveis em termos de qualidade dos canais de comunicação utilizados. Além de caracterizar os limites de correção desta técnica dentro de variações no canal emuladas em ambiente de laboratório.

1.3. ORGANIZAÇÃO DO TRABALHO

Esse trabalho está organizado da seguinte forma: o Capítulo 2 apresenta a arquitetura utilizada no projeto, O Capítulo 3 apresenta um resumo dos conceitos preliminares necessários

para esta dissertação, apresentando uma visão geral sobre holografia, áudio e vídeo, padrões de compressão de vídeo e teoria da informação. O Capítulo 4 descreve os diferentes tipos de mecanismos de correção de erros existentes e apresenta o protocolo que utilizaremos no ambiente de implementação. O Capítulo 5 apresenta os trabalhos relacionados ao tema desta dissertação. O Capítulo 6 apresenta a proposta de implementação e a análise dos dados. Por fim, no Capítulo 7, são apresentadas as conclusões da dissertação e os trabalhos futuros que cercam este tema.

2. PROJETO TELESAÚDE

A arquitetura da solução proposta é mostrada na Figura 3, composta pelo Centro de Saúde Virtual (CSV) e pelo Centro de Saúde Holográfico (CSH). O CSH é o local onde a junta médica especializada se reúne para prestar apoio ao profissional de saúde e ao paciente que estão em localização remota. No CSH a infraestrutura é mais robusta e exige investimento pois é onde a projeção holográfica é feita, permitindo através da tecnologia proporcionar aos médicos a sensação de estar no mesmo local que o paciente. O CSV é o consultório remoto e deve ter uma infraestrutura mais simples, com equipamentos de mercado e baixa complexidade de implementação. A comunicação entre os dois locais é realizada por meio de uma VPN (*Virtual Private Network*), garantindo assim segurança a transmissão. A conexão pode ser realizada por fibra ótica ou por conexão via satélite.

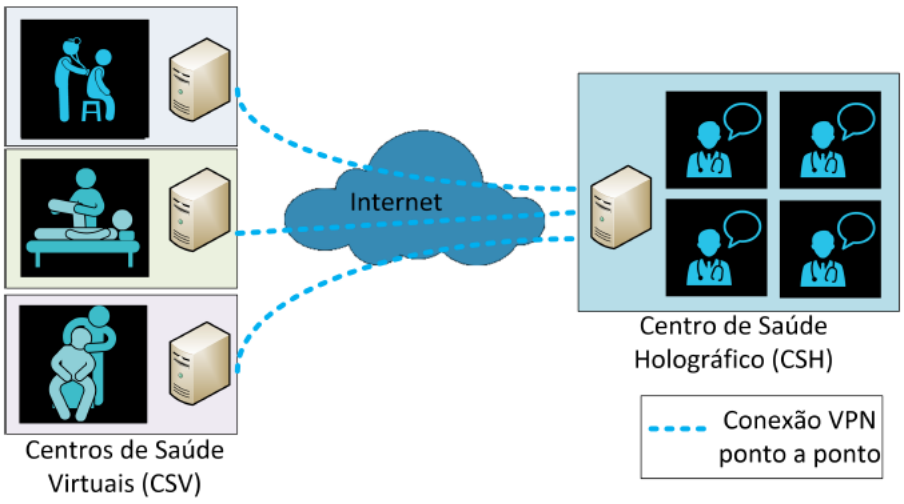


FIGURA 3: VISÃO GERAL DA SOLUÇÃO COM ARQUITETURA DE CONEXÃO ENTRE OS CSVS E O CSH [31].

O CSV é um consultório médico tradicional adequado a geração e captura da imagem holográfica e do som. Além disso, deve ser capaz de exibir imagem e som gerados pelo CSH com a equipe médica especializada. A Figura 4 mostra o CSV montado no CRASI (Centro de Recuperação e Assistência Social Integrada), Hospital Universitário Antônio Pedro.

FIGURA 4: CSV MONTADO NO CRASI/HOSPITAL UNIVERSITÁRIO ANTÔNIO PEDRO E OS RECURSOS DISPONÍVEIS.

O CSV contém uma maca, um computador com webcam HD, microfone, caixas de som e uma televisão. A sala deve ser revestida com tinta preta, nas paredes e no piso, garantindo assim a menor reflexão possível da luz garantindo assim maior qualidade na produção do efeito 3D da holografia. Conceitualmente o CSV deve ser uma instalação de baixo custo, fácil manutenção e operação.

O CSH é ilustrado na Figura 5, uma sala com paredes escuras e um palco para projeção holográfica. Nessa sala, o investimento é maior: microfones direcionais, mesa de som, mesa de iluminação, roteador, computador, data show atém da película para produção da holografia são necessários. A ideia é que uma sala dessa possa atender vários CSVs.

FIGURA 5: CSH INSTALADO NO CRASI/HOSPITAL UNIVERSITÁRIO ANTÔNIO PEDRO, NA UFF

3. ESTUDO BIBLIOGRAFICO

O conceito básico da comunicação se baseia em trafegar dados de um ponto emissor a um ou múltiplos pontos receptores através de mecanismos que permitam que a informação enviada possa ser recebida e restaurada em um ponto geograficamente distinto. Em um sistema de codificação digital a mensagem deve ser codificada em símbolos e preparada para ser transmitida em um canal. O transmissor, quando necessário, será responsável por agregar redundância à mensagem de modo a torná-la resiliente contra a incidência de ruídos e interferências. O receptor será responsável pelo processo de decodificação e reprodução do conteúdo buscando recriar a mensagem original. A Figura 6 ilustra o macroprocesso de envio e recebimento da informação [18].

FIGURA 6 SISTEMA DE COMUNICAÇÕES.

3.1. HOLOGRAFIA

O Sistema de Saúde Holográfico da UFF (Universidade Federal Fluminense) demonstra que é possível capturar, codificar e transmitir vídeo e áudio de um paciente em uma consulta médica, em localização remota. O sistema viabiliza que um hospital preparado com a tecnologia necessária para reprodução da imagem holográfica tenha um especialista apto a apoiar um médico remotamente durante a consulta e o diagnóstico [7].

O processo de geração de uma imagem holográfica é baseado nos conceitos básicos de incidência de raios luminosos em um dado sistema. Quando um objeto é iluminado, a onda se espalha atingindo os nossos olhos. Durante o processo de captura da imagem, o que é gravado é exatamente a variação da intensidade que a luz incide em um objeto, mas as informações referentes a fase são perdidas. Sendo assim, a percepção de dimensão em 3D não é capturada [7]. A Holografia é um método inventado por Gabor em 1948. É uma técnica que permite que se registre não só a amplitude, mas também a fase da onda luminosa.

A palavra holografia vem do grego e representa a junção da palavra *holos* significando completo e *graphein* significando gravação, ou seja, gravação de uma informação por completo. O resultado das variações de intensidade gravada foi denominado holograma, que com a

iluminação correta permite aos nossos olhos restaurar o objeto em amplitude e fase, possibilitando a percepção de profundidade que o nosso cérebro interpreta como um objeto em três dimensões [7]. Com o objetivo de conseguir uma imagem do paciente com a maior fidelidade possível, porém sem necessidade de uma banda muito elevada, a tecnologia escolhida para o Sistema de Saúde Holográfico da UFF foi aquela conhecida como *Pepper's Ghost*, que na prática não gera efetivamente hologramas, mas produz uma ilusão ótica gerando uma imagem tridimensional do paciente, do médico remoto e do ambiente da consulta [8]. Tal tecnologia foi desenvolvida em 1862 por John Henry Pepper, professor de química do Instituto Politécnico de Londres [9]. O efeito é criado através de uma película adicionada ao palco com um ângulo de 45º em relação ao plano do observador conforme ilustrado na Figura 7.

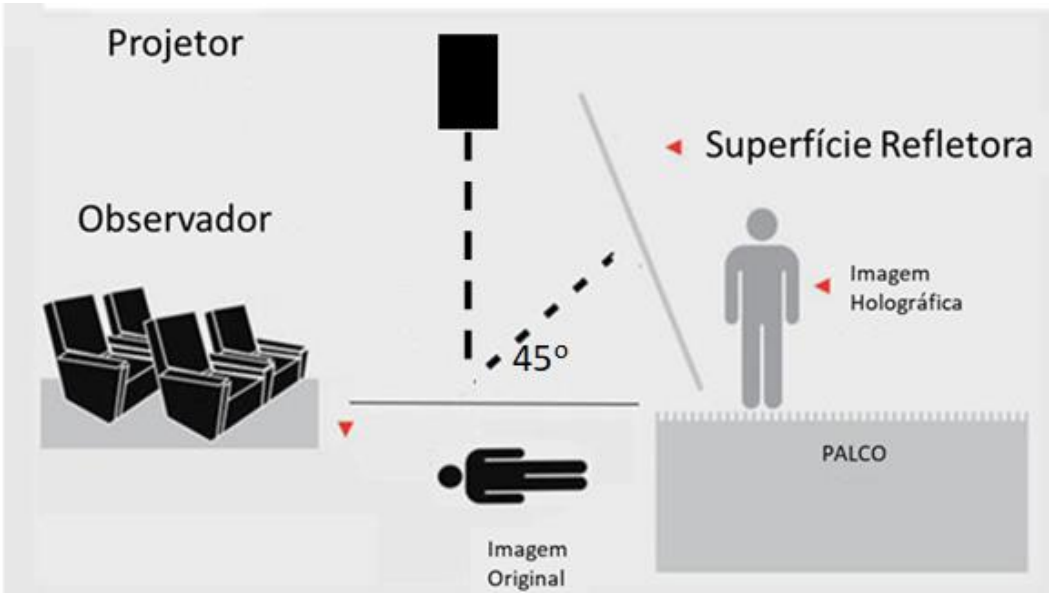


FIGURA 7: EFEITO DE PROJEÇÃO HOLOGRÁFICA CRIADO POR PEPPER'S GHOST.

O efeito de projeção holográfica criado por Peppers Ghost se baseia em técnicas de iluminação e reflexão e, por isso, o entendimento de propriedades óticas básicas se faz relevante. Para uma melhor compreensão do Sistema Holográfico em pauta, vamos analisar a interação das ondas luminosas em uma superfície refletiva transparente de vidro, o que nos permite considerar que a superfície se comportará como lente e também como espelho. Pela lei de Snell-Descartes, também conhecida como segunda lei da refração, podemos traçar o caminho de um raio luminoso quando incide em um dado sistema. A lei descreve a relação entre o índice de refração e o ângulo incidente, podemos definir o índice de refração (n) como o quociente

entre a velocidade de propagação da luz no vácuo (c) e sua velocidade de propagação no meio considerado (v).

$$Índice\ de\ Refração: n = \frac{c}{v}$$

A 2ª lei da refração descreve que o produto do índice de refração do primeiro meio, no qual se encontra o raio, pelo seno do ângulo que esse raio forma com a reta normal à interface no ponto de incidência com o segundo meio é a constante do produto do índice de refração do segundo meio pelo seno do ângulo de reflexão.

$$n_1 \sin\theta_1 = n_2 \sin\theta_2$$

Na Figura 8, podemos ver o comportamento do raio quando incide na superfície de vidro e ver que o mesmo sofrerá mudança de curso de propagação.

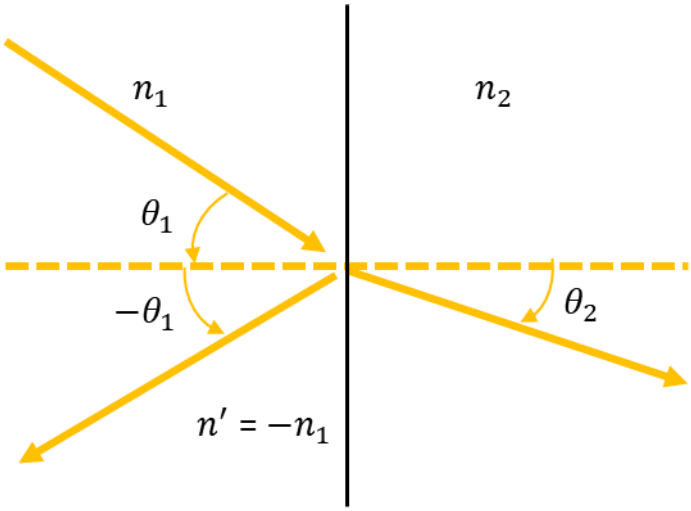


FIGURA 8: A FIGURA ILUSTRA A REFLEXÃO E REFRAÇÃO DE UM RAIO LUMINOSO QUANDO INCIDE EM UMA SUPERFÍCIE.

Na Figura 9 é exposto o modelo de reconstituição da imagem do paciente no sistema holográfico. Podemos ver que a imagem virtual é formada no lado oposto do espelho, essa propriedade explica porque a reflexão de objetos na ilusão de *Pepper's Ghost* aparece no palco

do lado oposto do vidro. A distância da imagem virtual ao espelho depende da distância da imagem original ao espelho.

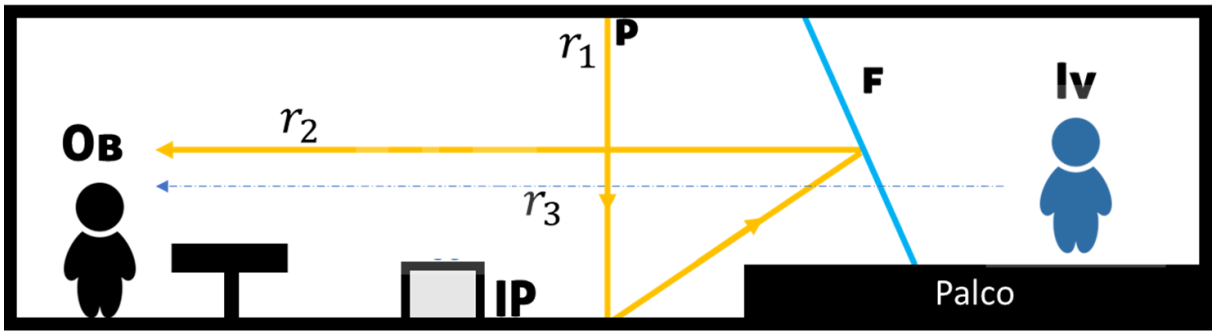


FIGURA 9: MODELO DE RECONSTITUIÇÃO DA IMAGEM DO PACIENTE NO SISTEMA HOLOGRÁFICO.

TABELA 2: DESCRIÇÃO DOS ELEMENTOS EXPOSTOS NA FIGURA 9.

ELEMENTO	ABREVIAÇÃO	DESCRIÇÃO
Observador	Ob	Caracteriza-se pelo especialista que observa a imagem virtual do paciente;
Imagem Virtual	Iv	É a imagem do paciente distante vista pelo Observador
Projetor	P	Projeta a imagem do paciente
Imagem do Paciente	IP	Local, denominado tecnicamente por "piscina", onde é projetada a imagem do paciente oriunda do projetor;
Foil	F	Película que reflete os raios provenientes da imagem do paciente e refrata os raios provenientes do Palco, dirigindo-os para o observador.

A ideia é simples, a imagem é projetada a imagem sobre a “piscina”, que reflete os raios, os quais atingem o *foil*. Por ser um material transparente, os raios refletidos do *foil* formam uma imagem virtual. Assim, através da reflexão e difração dos feixes r_2 e r_3, o observador verá uma imagem aparentemente tridimensional do paciente e do ambiente em que ele está inserido.

Contudo, essa sensação de 3D depende que o ambiente de projeção seja escuro. Outro fator que ajuda na sensação 3D é a aplicação de determinados feixes de luz que promovem a sensação de profundidade no palco, apesar do ambiente escuro. No ambiente do consultório de saúde virtual (CSV), algumas particularidades deverão ser atendidas para garantir uma boa imagem holográfica no destino. A principal delas, como já comentado, é a cor escura das paredes e a pintura aplicada deve ser capaz de absorver a luz e causar o mínimo de reflexão de luz nas paredes. Além disso, o CSV precisa ser muito iluminado, para diminuir o impacto das sombras na projeção em um local escuro como o consultório de saúde holográfico (CSH). Assim, iluminação é uma preocupação especial nesse tipo de ambiente. Outro ponto relevante é que a sensação de 3D está vinculada ao posicionamento da imagem projetada sobre o piso do palco. Uma imagem projetada que aproveita a largura do ambiente, mas não a profundidade, tende a ser mais facilmente projetada no local correto. Por exemplo, se a cena consiste tem duas pessoas em pé, uma ao lado da outra, é simples ajustar a imagem para que ambos fiquem com os pés no chão do palco. Contudo, se existem duas pessoas posicionadas uma atrás da outra, pode acontecer de os pés da pessoa de trás não tocarem o chão do palco. No caso do CSV, existem objetos que são projetados e que estão em profundidades diferentes, como mostrado na Figura 10: a maca, que fica no fundo da sala, e a mesa, que fica de frente para a câmera. Dependendo do tamanho do consultório e da distância entre a câmera e a mesa, pode não ser possível acertar a projeção da maca sobre o palco, criando a sensação de um objeto voando.

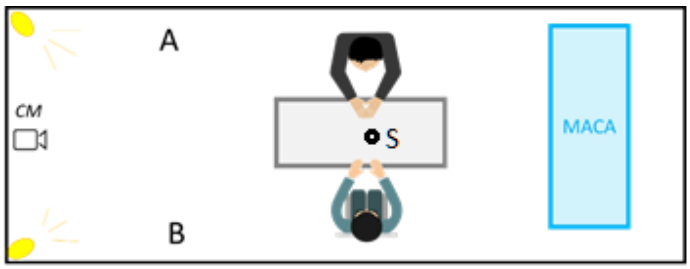


FIGURA 10: CONSULTÓRIO DE SAÚDE VIRTUAL.

TABELA 3: DESCRIÇÃO DOS ELEMENTOS EXPOSTOS NA FIGURA 10..

ELEMENTO	ABREVIAÇÃO	DESCRIÇÃO
Câmera	CM	Responsável pela captura das imagens do profissional de saúde e do paciente;
Caixas de som	A E B	emissor de áudio (*full duplex*) recebido pela sala de projeção;

Microfone	S	Equipamento responsável por captar o som ambiente;
Maca	MACA	A ser utilizada quando não se justifica o atendimento do paciente na mesa de consulta

Além desses elementos, existe ainda o plano de iluminação, que conta com focos de luz à meia altura em A e B, sobre a maca, embaixo da mesa e no teto. O ambiente deve possuir paredes escuras para intensificar o contraste das imagens transmitidas.

3.2. STREAMING DE VÍDEO

O melhor aproveitamento da infraestrutura desenvolvida exige um cuidadoso estudo das possíveis configurações do streaming de vídeo, buscando encontrar uma configuração que permita uma transmissão robusta e confiável mesmo com um canal com banda, perda e latência variáveis. Cada etapa do processo deve ser cuidadosamente estudada: captura, codificação/decodificação e compressão/descompressão do mesmo. A escolha atenta de cada parâmetro pode melhorar a qualidade do conteúdo transmitido, como qual codec de áudio e vídeo utilizar, taxa de bit, tamanho do *Group of Picture* (GOP), taxa de quadros, resolução e protocolo de transporte. Assume-se uma restrição na banda passante disponível, dado que, em geral, os CSVs ficarão em locais remotos, possivelmente conectados por provedores de acesso Internet de baixa capacidade. Na etapa da captura, a escolha da câmera é importante. A câmera deve atender os requisitos mínimos definidos para a comunicação e modelo de atendimento proposto. No caso de uma consulta médica remota, uma *webcam* pode ser utilizada, mas para termos uma codificação eficiente é importante observar as características do material capturado pela câmera escolhida. As configurações do codificador devem ser escolhidas baseadas nas configurações disponíveis na câmera: quanto maior a qualidade do material capturado, melhor será a qualidade resultante da codificação, mas esse custo benefício deve ser analisado para atender as premissas definidas para a especificação do CSV. A captura é a parte do processo responsável pela digitalização do material e duas etapas determinarão a qualidade do mesmo: amostragem e quantização [21]. A amostragem é a etapa que transformará o sinal capturado contínuo em uma sequência de amostras, tensões em diferentes intervalos de tempo, de modo que permita que o sinal seja totalmente recuperado. A quantização é a etapa que define a quantidade de níveis de tensão que será utilizado para reconstruir o sinal. Para o vídeo digital em alta definição, é recomendado o uso de 10 bits (1024 níveis) [21]. O processamento de uma

quantização inferior a 10 bits pode gerar artefatos e perda de resolução nos contornos da imagem. Após a fase de captura, o conteúdo de vídeo precisará, então, ser preparado para o canal de transmissão pretendido. Para isso, o vídeo será codificado e comprimido, de modo que a ser representado pela menor quantidade de símbolos, resultando assim em uma largura de banda inferior. Existem diferentes técnicas para codificar um vídeo e a escolha do codec deve ser feita cuidadosamente garantindo que se atinja o objetivo. A codificação busca representar em símbolos a informação [21]. A compressão busca eliminar as repetições e representar as palavras códigos de forma eficiente, por exemplo através da estatística. Além disso é comum para compressões de vídeo o uso de métodos que removam a redundância em cada quadro de vídeo (compressão entraram ou espacial) ou entre quadros de vídeo (compressão interframes ou temporal) [20].

O papel do codificador é buscar separar a informação única e eliminar a redundância do sinal transmitido [20], se após eliminar por completo a redundância não for possível atingir o resultado desejado, alguma informação deverá ser descartada.

Sistemas de compressão com perdas alcançam a redução de dados pela remoção de informação irrelevante ou de relevância menor. Para decidir o que pode ser considerado informação menos relevante, é importante que a análise seja feita no contexto da aplicação. No caso de vídeo e áudio, deve ser considerado o que os sistemas visual e auditivo dos seres humanos não são capazes de perceber [21]. Um sistema ideal conseguiria buscar toda a redundância espacial e temporal e eliminá-la, transmitindo apenas à taxa de entropia. Porém, na prática, isso demandaria um alto tempo de processamento, tornando o sistema complexo, caro e lento [20].

3.3. PADRÕES DE COMPRESSÃO DE VÍDEO

Durante anos, dois grupos foram os pioneiros e os principais responsáveis pelo desenvolvimento de padrões de compressão de vídeo: o MPEG (*Moving Picture Experts Group*) e a UIT (*União Internacional de Telecomunicações*) [20]. A Figura 11 ilustra a linha do tempo dos padrões de vídeo onde podemos verificar as contribuições dos grupos UIT-T VCEG (Telecommunication Standardization Sector Video Coding Experts Group) e MPEG WG 11 [20]. Os codecs resultantes são publicados por ambos os grupos, no grupo MPEG com o nome de MPEG-X e os publicados pelo grupo da UIT com o nome de H26X.

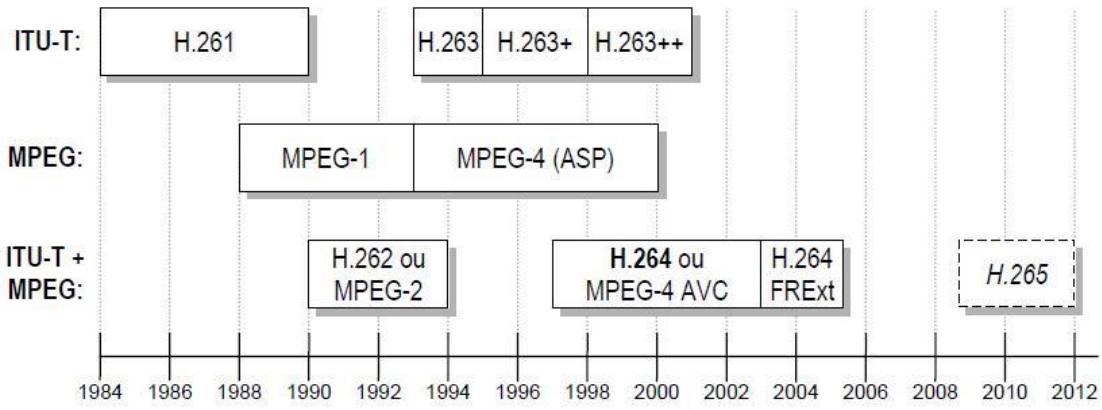


FIGURA 11: PADRÕES DE COMPRESSÃO DE VÍDEO NO TEMPO POR GRUPO DESENVOLVEDOR.

O padrão MPEG-4 é um algoritmo de referência e posteriormente foi remodelado pela UIT e chamado de H.264/AVC. O H.264/AVC possui uma compressão considerada de excelente qualidade, razão pela qual esse padrão foi adotado nas implementações do projeto. É comum chamá-lo de H.264/AVC ou H.264/MPEG- 4 AVC ou ainda MPEG-4 part 10. Esse padrão trata exclusivamente do vídeo [20]. A escolha do codec é de grande importância, pois interfere diretamente na qualidade do vídeo e na taxa de transmissão, os quais são dois fatores sensíveis na aplicação proposta. Seu desempenho é medido em função da quantidade de bits que são necessários para alcançar a qualidade visual pretendida para um dado vídeo. Essa qualidade, embora um tanto quanto subjetiva, depende da nitidez, sincronização entre áudio e vídeo, suavidade de contornos e movimentos e fidelidade das cores [20]. Na Figura 12, podemos observar os diferentes padrões de vídeo lançados comparando as taxas para transmitir a mesma informação, o que torna clara a nossa escolha pelo H.264. Esse padrão é capaz de comprimir em menor taxa de bits por segundo uma maior quantidade de informação.

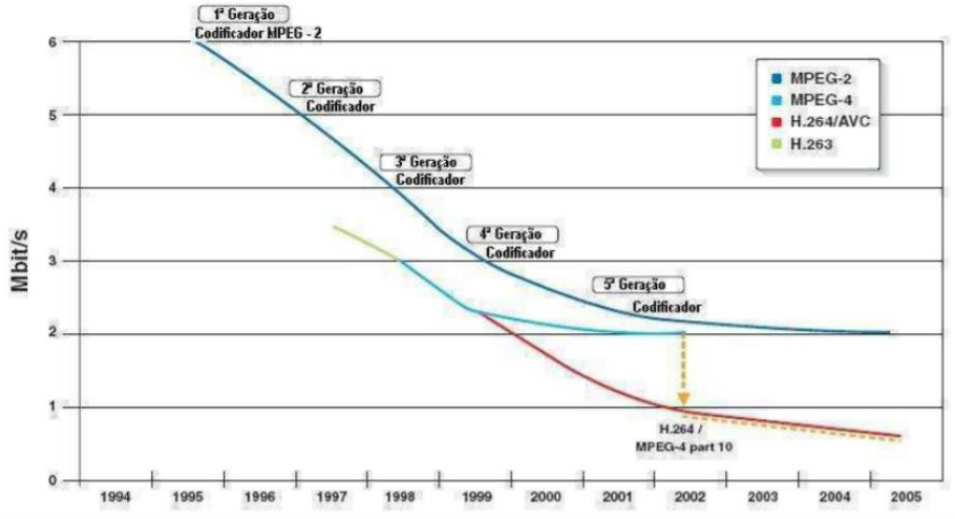


FIGURA 12 GERAÇÕES DE PADRÕES NO TEMPO PELA TAXA.

O algoritmo H.264 é responsável por realizar uma compressão com perdas, que busca eliminar informações do vídeo que não possam ser percebidas pelo olho humano. Dentre os mecanismos utilizados para eliminação de informação de menor relevância, está a sub amostragem de croma, devido a característica do olho humano ser mais sensível a informações de luminância que de crominância. É possível reduzir a amostragem de sinais de croma pela metade na horizontal quase sem prejuízo à imagem, gerando uma relevante economia na taxa de bits necessária para representar a informação.

O particionamento do quadro em macro blocos de informação de tamanhos variáveis é outro método de compressão relevante do H.264. Este método permite particionar a imagem de modo a transmitir apenas uma vez a informação relevante e, através desses macros blocos, recriá-las utilizando compressão intraframes. Na compressão interframe, utiliza-se os macro blocos como referência e adicionam-se informações de vetores de movimento para reposicionar a imagem nos frames subsequentes.

É possível subdividir um vídeo em Grupos de Imagens (GOP – *Group of Pictures*) e, através de um frame de referência, utilizar mecanismos de predição e vetores bidirecionais para reproduzir as imagens daquele grupo. Nesse caso, o frame I (Intra) contém a maior parte da informação e do peso e os demais B (bilateral) e P (preditivo) carregam pouquíssima informação.

3.4. SISTEMA DE COMUNICAÇÃO DIGITAL

O sistema de comunicação digital é responsável por converter a informação de uma fonte com sinal analógico (como áudio e vídeo) ou de um sinal digital (como dados de um computador), para um sinal digital, discreto no tempo e com número finito de símbolos. O sistema de comunicação digital é responsável por traduzir as mensagens de uma determinada fonte para uma sequência de dígitos binários. Idealmente, o sistema deve buscar representar a mensagem da forma mais eficiente possível e isso representa traduzir as mensagens no menor número de dígitos binários, removendo toda a redundância. Este processo é chamado de codificação da fonte. A sequência binária resultante é, então, entregue ao codificador do canal, que adicionará alguma redundância de modo que o receptor seja capaz de compreender a informação mesmo que a mensagem seja alterada pela incidência de interferência e ruído no canal. A redundância é um mecanismo que permite aumentar a resiliência do sinal em um canal ruidoso e o mecanismo escolhido deve ser cuidadosamente projetado para encontrar um ponto ótimo que garanta a eficiência da transmissão [13].

O projeto deve garantir um sistema que atenda as especificações dos principais elementos: fonte de informação, canal de comunicação e usuário receptor da informação, considerando também o tipo de aplicação e a experiência do usuário. O transmissor deve codificar e modular o sinal e transmitir no canal de modo que o receptor decodifique e demodule um sinal estimado próximo o suficiente do esperado para a aplicação, com custo adequado. O sistema de comunicação digital é ilustrado no diagrama da Figura 13 e explicita os elementos: Codificador/decodificador de fonte, codificador/decodificador de canal e modulador/demodulador [18].

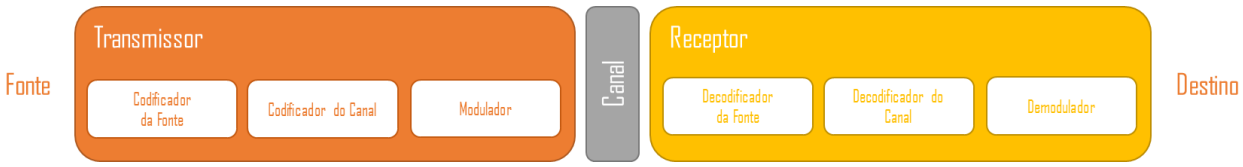


FIGURA 13 DIAGRAMA DE BLOCO DE UM SISTEMA DE COMUNICAÇÃO DIGITAL

O codificador de fonte é responsável por remover a redundância de modo a garantir o uso mais eficiente do canal. O resultado é uma sequência de símbolos chamada de palavra código da fonte. O fluxo de dados é processado em seguida pelo codificador do canal que será

responsável por produzir uma sequência de símbolos chamada de palavra código do canal. A palavra código do canal é mais longa que a palavra código da fonte pois adiciona redundância. O modulador é responsável por representar cada símbolo por um símbolo analógico produzindo um sinal resultante chamado de forma de onda que é adequado para ser transmitido pelo canal. No receptor, o sinal recebido é processado na ordem inversa que a explicitada acima para o transmissor, de modo a passar a forma de onda novamente para símbolos digitais, remover a redundância e decodificar a mensagem [18].

3.5. TEORIA DA INFORMAÇÃO E CODIFICAÇÃO

A teoria da informação descreve conceitos e teoremas relacionados a uma fonte binária e um canal simétrico, onde o canal não é confiável e pode ser descrito por uma probabilidade de erro p, onde p está compreendido entre 0 e $^1/_2$.

O teorema de Shannon descreve como tornar uma transmissão confiável relacionando três parâmetros: codificação, fonte de informação e capacidade do canal. Se a taxa de informação de uma dada fonte não excede a capacidade do canal, então é possível utilizar uma técnica de codificação que torne possível a transmissão através de um canal não confiável com taxa de erros tão baixa quanto desejável [19].

A codificação da fonte é o processo que produz uma representação para dados gerados por uma fonte discreta. Para que haja eficiência na escolha desta representação é necessário entender a estatística da fonte. Considerando a probabilidade de um símbolo acontecer, é possível explorar a geração de símbolos de uma fonte e escolher de forma eficiente as palavras código associadas, gerando códigos mais curtos para símbolos mais frequentes e códigos mais longos para símbolos menos frequentes, criando um código de comprimento variável [18].

O limite de Shannon prova que qualquer sistema de comunicações pode ser caracterizado pelo teorema de capacidade máxima do canal. Se a taxa da fonte de informação R é menor que a capacidade do canal C então existe um método de codificação do canal que garante que a informação seja transmitida de forma confiável. Esse teorema estabelece um limite teórico superior a taxa da fonte de informação na qual é possível codificar a informação de forma confiável através de um canal de comunicações [12].

A codificação do canal é responsável pelo mecanismo de correção de erros que permitirá tornar uma transmissão mais confiável em um canal ruídoso adicionando mais dados ao d. Por outro lado, o codificador de fonte é responsável por codificar a informação gerada pela fonte de modo que seja apropriada ao envio dentro da capacidade do canal [18].

3.6. MECANISMO DE CORREÇÃO DE ERROS

Diferentes técnicas podem ser utilizadas para adicionar robustez à transmissão. Essas técnicas são divididas em dois grandes grupos: Requisição automática de repetição (ARQ) que automaticamente requisita ao transmissor o reenvio de um pacote corrompido ou perdido e a técnica de correção antecipada de erros (FEC) que já carrega redundância na mensagem.

Esta pode ser subdividida em outras duas categorias que descrevem o método de inserção de redundância: Código de Bloco e Códigos Convolucionais. Os códigos de blocos não possuem memória nos codificadores e os códigos convolucionais possuem memória nos codificadores [13][14].

É possível utilizar esses mecanismos tanto para a detecção quanto para a correção de erros. As técnicas de codificação de erros permitem que dada uma taxa de erro R, menor ou igual a capacidade do canal, é possível encontrar um código de correção de erros que torne a probabilidade média de erros tão pequena quanto se queira002E Na Figura 14 podemos ver as diferentes técnicas que permitem tornar o sinal resiliente ao trafegar por canais ruidosos [13].

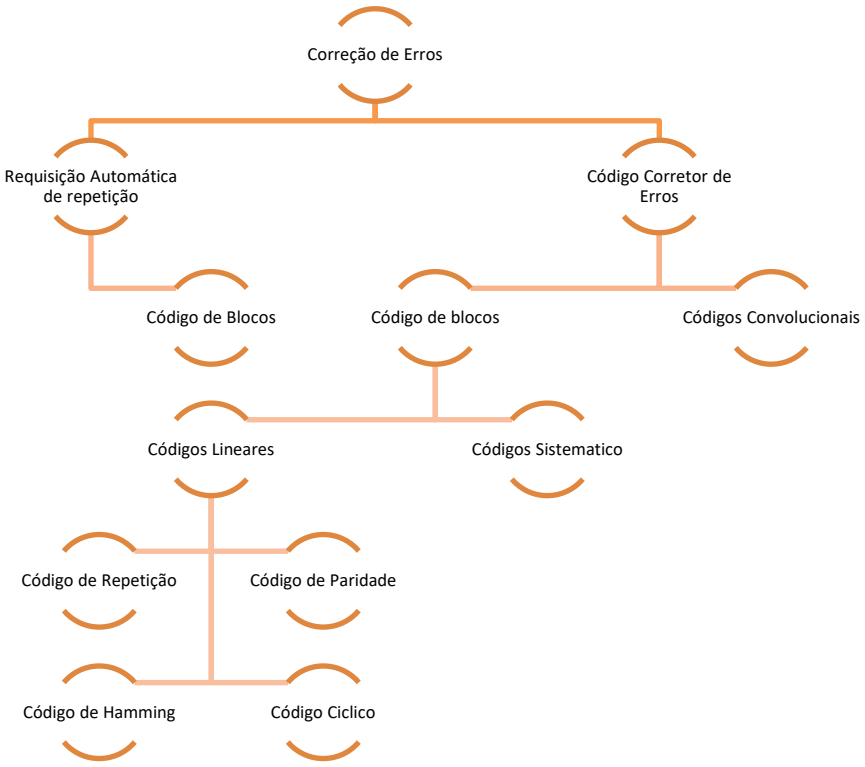


FIGURA 14: CLASSIFICAÇÃO DE MECANISMOS DE DETECÇÃO E CORREÇÃO DE ERROS

3.7. CODIFICAÇÃO DE BLOCOS

A definição de código de bloco linear é que dadas duas palavras código quaisquer, a operação ou exclusivo gera uma terceira palavra código representando a informação de redundância das duas primeiras [14][18]. As técnicas de codificação de blocos preveem adicionar uma quantidade fixa de informação como redundância de um bloco com tamanho também pré-definido e fixo. É um mecanismo sem memória que analisa e trata individualmente cada bloco [14].

O código de bloco prevê que um fluxo de informações será subdividido em blocos de tamanhos fixos. Esses blocos deverão ser processados e acrescidos de uma redundância dependendo do algoritmo desenvolvido [13].

A codificação de blocos codifica blocos de k bits de mensagem com n-k bits de redundância gerando um bloco codificado chamado "palavra código" de n bits, onde n é o tamanho do bloco do código e $n > k$ [18]. R_0 é a taxa de dados do canal, R_s é a taxa da fonte de informação, e k/n é a taxa do código [12]. A taxa de dados do canal é explicitada pela relação da taxa da fonte de informação pela taxa do código conforme equação:

$$R_0 = \frac{n}{k} R_s \text{ [bits/seg]}$$

Os mecanismos podem codificar bits, bytes, quadros e até pacotes, sendo o primeiro geralmente efetuado na camada física e o último na camada de aplicação. O codificador da fonte tem o objetivo de comprimir a informação extraindo ao máximo a redundância da mesma e o codificador do canal adiciona redundância ao sinal, de forma inteligente para aumentar a robustez da transmissão.

O incremento de robustez do código gera uma degradação na taxa de transmissão que pode ser definida como a relação entre a quantidade de informação n em bits e a quantidade de k bits codificados conforme exposto na equação [12].

$$Rc = \frac{k}{n}$$

4. A PERDA DE PACOTE E A CORREÇÃO DE ERROS

A perda de pacote é um comportamento comum em redes não gerenciadas, ocorre com frequência quando há congestionamento entre origem e destino. A recuperação de um pacote pode ser feita utilizando técnicas de *feedback* ou de correção de erros. A técnica de *feedback* costuma ser pouco eficiente em redes muito grandes devido ao atraso gerado pela solicitação de reenvio e pelo tempo de chegada do novo pacote ao receptor, o que pode gerar impacto na qualidade da experiência quando observamos aplicações em tempo real [20].

O protocolo TCP (*Transmission Control Protocol*) prevê a recuperação dos pacotes através do mecanismo ARQ (*Automatic Receive Request)*. O ARQ exige confirmação de chegada do pacote no destinatário, caso essa confirmação não ocorra em um tempo pré-determinado o nó de origem reenviará o pacote ao destino. Este tipo de protocolo baseado em mecanismo de confirmação de recebimento soluciona a perda de pacote ao custo de transmitir em menor taxa se comparado com o protocolo UDP (*User Data Protocol*) [22].

A qualidade do vídeo é diretamente proporcional a taxa do mesmo, por isso os protocolos UDP e RTP *(Real time Protocol)* são os mais utilizados para entrega de vídeo. Entretanto, UDP e RTP não utilizam mecanismos de retransmissão e a perda de um único pacote IP pode causar relevante impacto na qualidade, pois cada pacote IP carrega 7 pacotes MPEG. Mecanismos para recuperação de pacotes perdidos como o XOR (ou-exclusivo) e *Raptor Code* são utilizados com os protocolos UDP e RTP [20][22].

Para transmissões em tempo real é mais adequado implementar o protocolo UDP (*user data protocol*). Esse protocolo também se baseia no envio de pacotes de informação, mas remove toda a parte de verificação de erros, buscando acelerar o processo de envio de dados, além disso possui menor *overhead* [22].

O protocolo UDP simplesmente envia informações a um destinatário sem se preocupar se foram recebidas. Independente de ocorrem erros, o próximo pacote programado pelo sistema é enviado e os anteriores acabam não sendo recuperados. Para que seja possível a correção, um mecanismo de correção de erros pode ser utilizado enviando pacotes redundantes de modo que o receptor possa recuperar o pacote perdido ou corrompido.

Na Figura 15, é possível observar o resultado da perda de pacote em uma transmissão de fluxo de vídeo gerando impacto direto na qualidade e perda da informação para o usuário final. O mecanismo de correção de erros é mais indicado para aplicações que possuem alto compromisso com atraso. Este mecanismo introduz, dentro de cada pacote, algum nível de redundância permitindo identificar e corrigir erros mesmo que haja perda de parte dos dados.

Sendo possível remover a aleatoriedade da perda e caracterizar o canal é possível prever quando e quanto de redundância precisa ser adicionada para que a correção de erros seja eficiente [21].

A FEC a nível de pacote é útil quando pensamos em rede cabeadas onde é mais comum a perda de pacote que o erro dos dados, que se observa mais impactante em redes sem fio. Em uma rede híbrida, pode se obter boa robustez na transmissão de dados se utilizados ambos os mecanismos. Na rede cabeada, a perda de pacote é geralmente causada devido ao congestionamento nos roteadores e na rede sem fio a perda de pacote ocorre devido a erros causados pelo efeito de desvanecimento ou efeito de múltiplos caminhos [21].

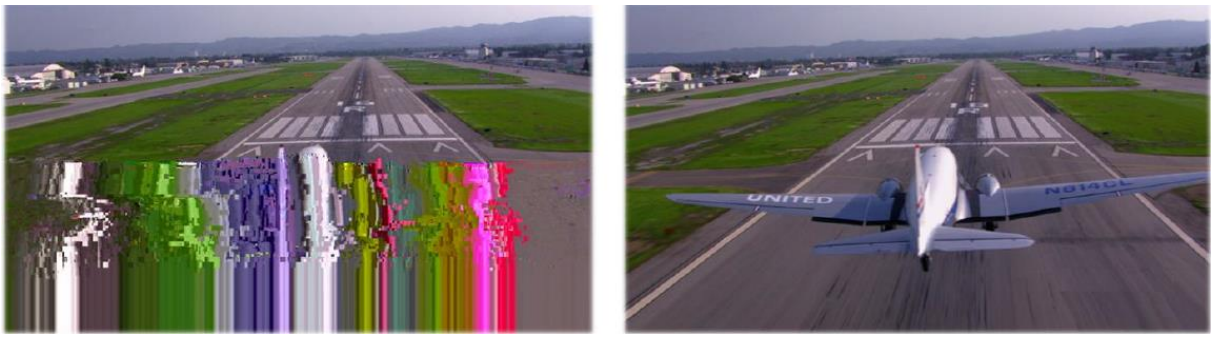


Figura 15: Comparação entre uma imagem A: com perda de pacotes durante a transmissão e a B: imagem original.

Existem três principais implementações na indústria para tecnologias de FEC a nível de pacote baseadas em códigos de bloco linear com ou exclusivo: PRO MPEG FEC (COP3R2)[23], SMPTE 2022 [31] e Digital Fountain's Raptor Codes [51]. O PRO MPEG FEC, que será descrito mais à frente em detalhes, descreve o mecanismo predecessor do uso do ou exclusivo para adicionar os pacotes de redundância a uma transmissão de vídeo sobre IP [22].

4.1. RFC 2733

A RFC 2733 foi proposta pelo IETF (*Internet Engineering Task Force*) e descreve um formato de carga útil para um pacote encapsulado em RTP carregando uma FEC genérica. A FEC descrita deve satisfazer as propriedades abaixo:

- Poder ser utilizada para qualquer tipo de mídia, seja áudio ou vídeo;
- Ser flexível para suportar qualquer mecanismo de FEC; e
- Ser desenvolvido para ser adaptável a diferentes necessidades.

Para geração dos pacotes de FEC, a RFC descreve os mecanismos de ou exclusivo e paridade. A RFC propõe diferentes esquemas para implementação e transmissão do mecanismo de FEC [11], que iremos detalhar na sequência.

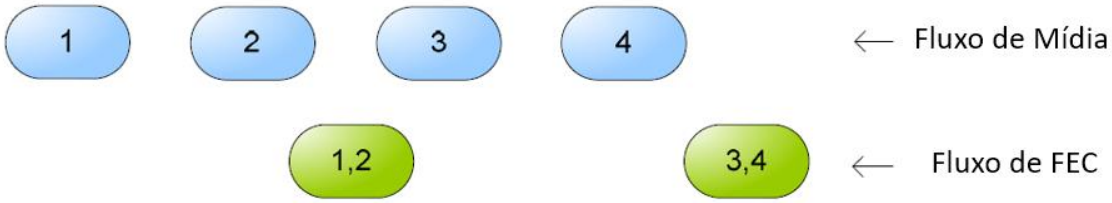


FIGURA 16: FLUXO DE FEC UNIDIMENSIONAL.

O esquema mostrado na Figura 16 é simples, eficiente e prevê trafegar em um fluxo de dados separado, enviado em outra porta lógica, todos os pacotes de FEC. A ilustração mostra que o pacote de redundância é resultante da combinação de dois pacotes de dados e será enviado em um segundo fluxo. Neste exemplo, cada dois pacotes de mídia gera um acréscimo de 50% de sobrecarga.

Outro esquema pode ser visto na Figura 17. Este modelo é mais complexo computacionalmente, mas carrega mais redundância, em cada pacote de redundância carregamos os dados de mídia do pacote anterior e do pacote posterior. Este esquema permite corrigir até dois pacotes perdidos em sequência porém dobra o *overhead* [11][17].

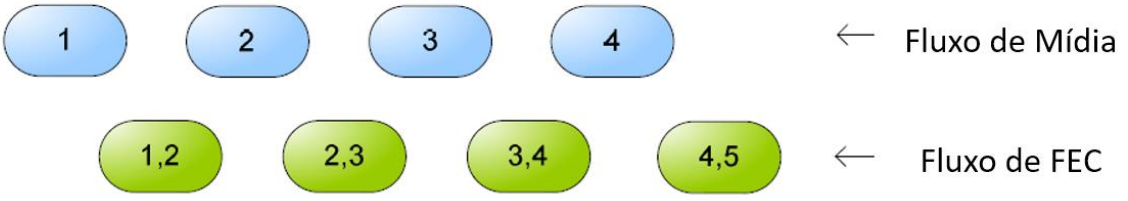


FIGURA 17: PROPOSTA DE FEC, ESQUEMA 1

A Figura 18, ilustra outra opção de modelo de inserção de FEC, onde apenas os pacotes resultantes de FEC serão transmitidos, esse modelo permite uma transmissão com menor

sobrecarga de dados e é capaz de corrigir erros em rajada de até três pacotes pois carrega a informações de cada pacote em três pacotes de redundância consecutivos. Entretanto, a recuperação pode ocasionar um atraso indesejado, pois se faz necessário receber vários pacotes antes de iniciar o processo de recuperação[11][17].

FIGURA 18: PROPOSTA DE FEC, ESQUEMA 2

O esquema ilustrado na Figura 19 permite recuperar até três pacotes consecutivos perdidos pelo custo computacional de armazenar e processar pelo menos quatro pacotes [11][17].

FIGURA 19: PROPOSTA DE FEC, ESQUEMA 3

O pacote de FEC é construído colocando na carga útil do RTP o cabeçalho e a carga útil do pacote de FEC, conforme ilustrado na Figura 20:

FIGURA 20: ESTRUTURA DE UM PACOTE DE FEC

A RFC propõe que o receptor saiba qual pacote de mídia gerou cada pacote de FEC, mesmo no caso de chegarem no receptor pacotes fora de ordem. O pacote de FEC é enviado em um fluxo separado e o pacote de mídia é enviado como se não houvesse mecanismo de FEC associado. A Figura 21 ilustra os parâmetros existentes no cabeçalho do RTP e veremos, na sequência, um exemplo de como este cabeçalho é processado para a geração do pacote de FEC resultante [17].

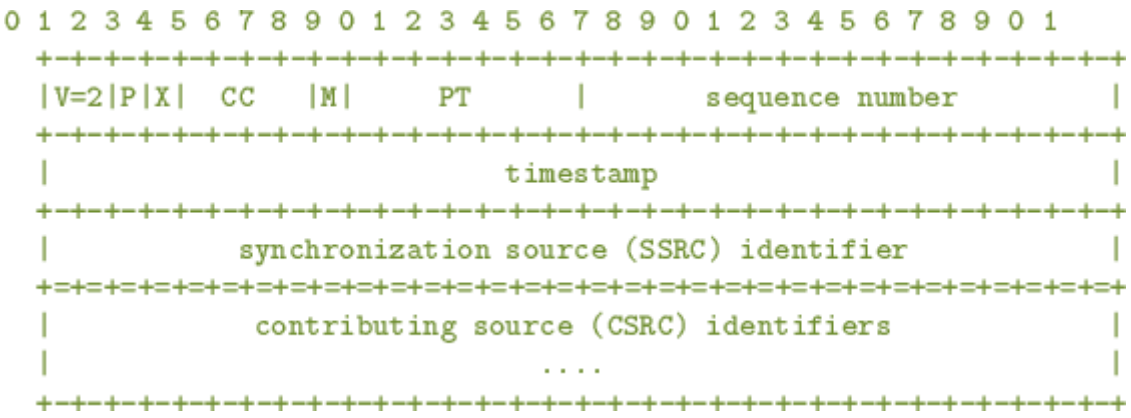


FIGURA 21: CABEÇALHO RTP

Nos pacotes de FEC, o campo versão é sempre definido como 2. O campo bit de preenchimento *(padding bit)*, bit de extensão *(extension bit)* e o campo bit de marcação *(marker)* são definidos através da combinação de paridade entre os pacotes de mídia que se intenciona adicionar a redundância. O campo SSRC possui o mesmo valor da mídia a ser protegida. Os campos CSRC e o campo extensão *(extension)* nunca estarão presentes. O campo do número de sequência *(sequence number)* será sempre o valor do pacote de FEC anterior acrescido de 1. O campo selo de tempo *(timestamp)* receberá o valor de tempo da mídia RTP no instante que o pacote de FEC é transmitido. O cabeçalho de FEC é definido conforme ilustrado na Figura 22.

FIGURA 22: CABEÇALHO DA FEC

4.2. OPERAÇÃO DE REDUNDÂNCIA

O mecanismo que gera o pacote de redundância concatena no campo carga útil do RTP os campos do cabeçalho e a carga útil do pacote de FEC. As diferenças de tamanho entre os campos comparados são preenchidas com zeros de modo a permitir que as sequências sejam percorridas e o ou exclusivo seja computado gerando o pacote resultante. Para cada pacote de mídia que será protegido, uma sequência de bits é gerada concatenando os campos do cabeçalho na sequência e operando o ou exclusivo entre a sequência de bits dos dois pacotes de mídia.

O procedimento para reconstrução do pacote perdido prevê a reconstrução não só da carga útil, mas também dos campos de cabeçalho de RTP e de FEC. Este procedimento é composto por duas etapas, a primeira que determina quais pacotes de mídia e de FEC devem ser combinados para que se obtenha a recuperação do pacote perdido e a segunda etapa que é responsável como reconstruir um dado perdido.

Para ilustrar o processo de inclusão da redundância consideraremos dois pacotes de mídia, X e Y, a serem enviados nas posições 8 e 9, com *timestamp* 3 e 5 respectivamente. O pacote X utiliza um *payload* tipo 1 e o pacote Y um *payload* tipo 19. O pacote X possui 10 bytes de *payload* e o pacote Y possui 11 bytes de *payload*. O pacote Y possui o bit *marker* definido como 1 e o cabeçalho RTP dos pacotes X e Y é ilustrado nas Figura 23 e Figura 24 respectivamente:

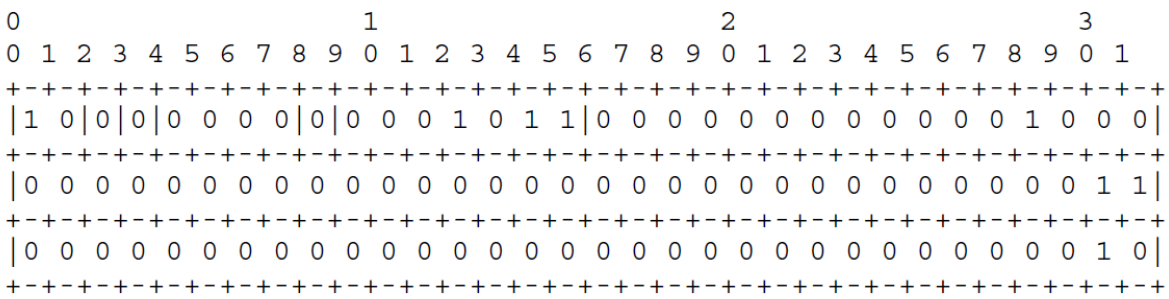


FIGURA 23: CABEÇALHO RTP PARA PACOTE DE MÍDIA X

Observando a Figura 23 podemos ver a configuração do cabeçalho RTP definido para o pacote X e conforme descrito abaixo:

Version: 2 (1 0)

Padding: 0 (0)

Extension: 0 (0)

CC: 0 (0000)

Marker: 0 (0)

PTI: 11 (0001011)

SN: 8 (0...1000)

TS: 3 (0...011)

SSRC: 2 (0..0100)

Observando a Figura 22, podemos ver a configuração resultante do cabeçalho de FEC definido para a combinação dos pacotes X e Y e ilustrado na Figura 24 e Figura 25 conforme abaixo:

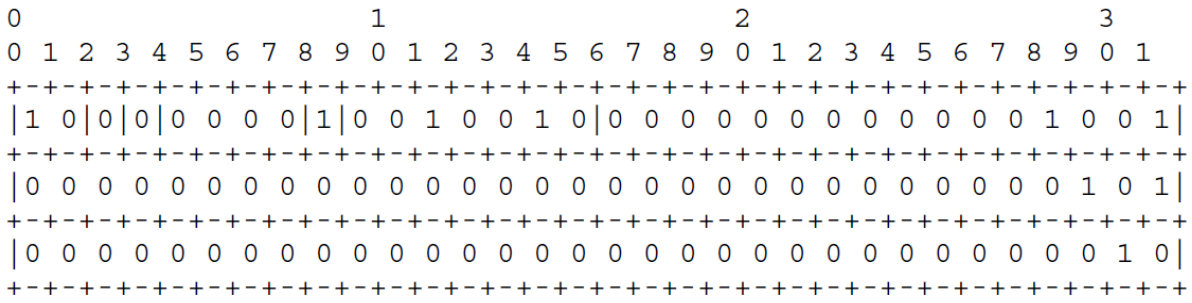


FIGURA 24: CABEÇALHO RTP PARA PACOTE DE MÍDIA Y

Observando a Figura 21 podemos ver a configuração do cabeçalho RTP definido para o pacote Y conforme descrito abaixo:

Version: 2 (1 0)

Padding: 0 (0)

Extension: 0 (0)

CC: 0 (0)

Marker: 1 (0001)

PTI: 18 (0010010)

SN: 9 (0...1001)

TS: 5 (0...101)

SSRC: 2 (0..010)

O cabeçalho de RTP resultante da combinação dos pacotes X e Y que formarão o cabeçalho de RTP do pacote de FEC resultante é exposto na Figura 25. O campo versão é sempre definido como “2” para pacotes de FEC. Os campos *padding, marker, SN e CC* são resultado do mecanismo de ou exclusivo definido pela Figura 27. Assumimos que o tipo de carga útil é parametrizado para 127 para indicar que é um pacote de FEC e o campo SSRC é definido de acordo com o parâmetro dos pacotes de mídia.

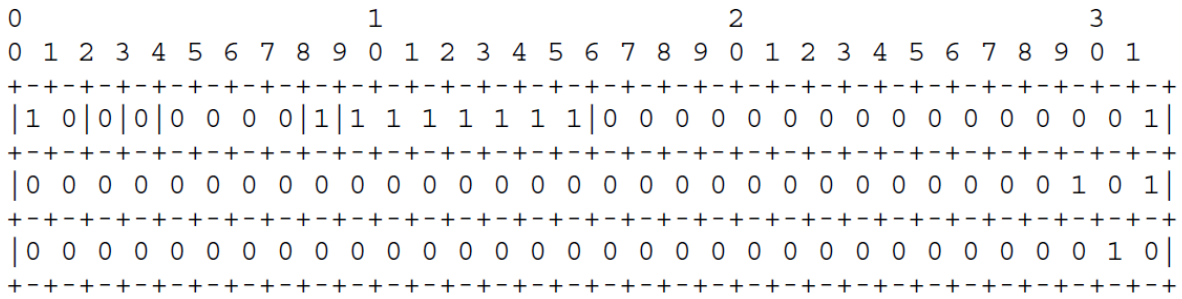


FIGURA 25: CABEÇALHO RTP DE FEC PARA OS PACOTES X E Y.

Observando a na Figura 25 podemos ver a configuração do cabeçalho RTP definido para o produto dos pacotes X e Y conforme abaixo:

Version: 2 (1 0)

Padding: 0 (0)

Extension: 0 (0)

Marker: 1 (0001)

PTI: 127 (1111111) -

SN: 1 (0...001)

TS: 5 (0...101)

SSRC: 2 (0...010)

O processo para criação do pacote de FEC aplica a lógica de ou exclusivo exposta na Figura 27. O mecanismo é realizado bit a bit percorrendo as *strings* de bits dos campos do cabeçalho de RTP do pacote X e do pacote Y. A Figura 26 ilustra o cabeçalho de FEC resultante formado conforme abaixo:

SN base: 8 [min(8,9)] = 1000

len. rec.: 1 [1000 xor 1001] = 0001

E: 0

PTI rec.: [01011 (11) xor 10010 18] = 11001 (25)

mask: 3 = 0011

TS rec.: [011 (3) xor 101 (5)] = 110 (6)

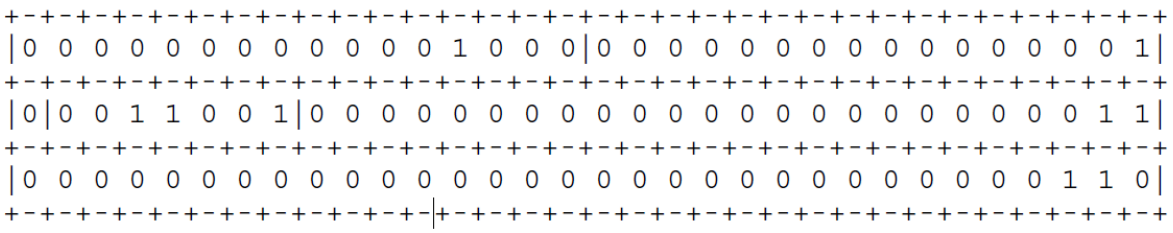


FIGURA 26: CABEÇALHO DE FEC RESULTANTE

Para a carga útil, o mesmo procedimento de ou exclusivo bit a bit deverá ser aplicado e cada bit de carga útil dos pacotes de mídia processado gerará um bit de carga útil do pacote resultante de FEC.

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

FIGURA 27: TABELA OU EXCLUSIVO

4.3. O PADRÃO PRO MPEG

O Pro-MPEG Fórum, formado em 1998, é uma associação de rádio difusores, produtoras, fornecedores de equipamento e componentes interessados em garantir a interoperabilidade de equipamentos profissionais de televisão, garantindo os requerimentos da radiodifusão e dos telespectadores. O fórum foi formado para suportar padrões abertos e aplicações emergentes. Atualmente, possui mais de 130 membros pelo mundo [11]. A recomendação descreve a transmissão de vídeo em MPEG-2 sobre redes IP e é chamada de MPEG Code of Practice #3 release 2 (CoP). Embora originalmente desenvolvida para *streamings* utilizando o codec de vídeo MPEG-2 o uso em outras codificações é transparente. O CoP foi desenvolvido baseado na RFC 2733 e propõe melhorias na capacidade de correção de erros descrita nesse documento [23].

O esquema de FEC proposto é um algoritmo bidimensional de ou exclusivo aplicado em uma matriz. A implementação mais simples possível é o uso de uma matriz unidimensional conforme exposto na RFC 2733. O padrão propõe a promoção da interoperabilidade e a simplificação da implementação, por isso definiu-se na recomendação um limite para o tamanho das linhas e colunas da matriz. Desta forma, garante-se que os fornecedores que implementem o padrão em suas soluções suportem minimamente qualquer combinação dentro destes limites, mas nada impede que extrapolem o mesmo.

O documento define que o número de colunas L deve ser de no mínimo 1 e no máximo 20. As linhas podem de no mínimo 4 até no máximo 20 linhas. O número total de pacotes que serão considerados no mecanismo de correção de erros não deve exceder 100 e deve atender as regras explicitadas nas equações subsequentes [11].

$$L \cdot D \leq 100$$
$$1 \leq L \leq 20$$
$$4 \leq D \leq 20$$

O mecanismo de correção de erros pode mitigar problemas de perda de pacote como reordenamento, erros de bits e perdas em rajada. A RFC 2733 define o formato de carga útil para permitir correção de erros de pacotes em RTP. Esse padrão descreve especificamente a transmissão de vídeo comprimido MPEG2-TS sobre IP encapsulado em pacotes RTP .

O mecanismo é bastante simples pois consiste apenas na operação de ou exclusivo, já descrita na sessão anterior, entre pacotes consecutivos conforme mostra a Figura 28:

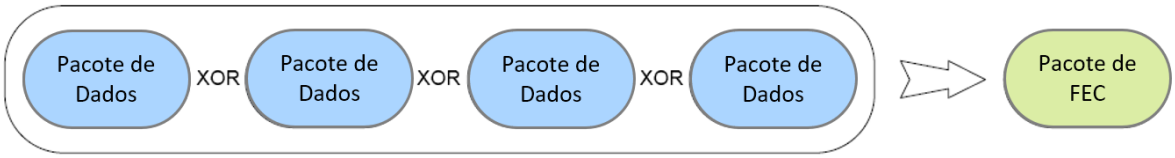


FIGURA 28: FEC EM UMA DIMENSÃO

A operação de ou exclusivo é simples, porém gera uma sobrecarga na transmissão dos pacotes extras. Na Figura 28, dos cinco pacotes transmitidos um é composto de informação redundante, ou seja 20% da transmissão neste caso não é informação exclusiva. Quanto maior o número de pacotes considerados no processo de criação do pacote de FEC, mais diluído se torna esse *overhead*, porém maior o atraso gerado para recuperar o pacote perdido. A Figura 29 ilustra o mesmo mecanismo sendo utilizado em uma matriz de duas dimensões, apesar do carga dos pacotes adicionais, este modelo permite aumentar a capacidade de reconstrução de múltiplos pacotes perdidos pois é capaz de reconstruir perdas em linhas e colunas [11].

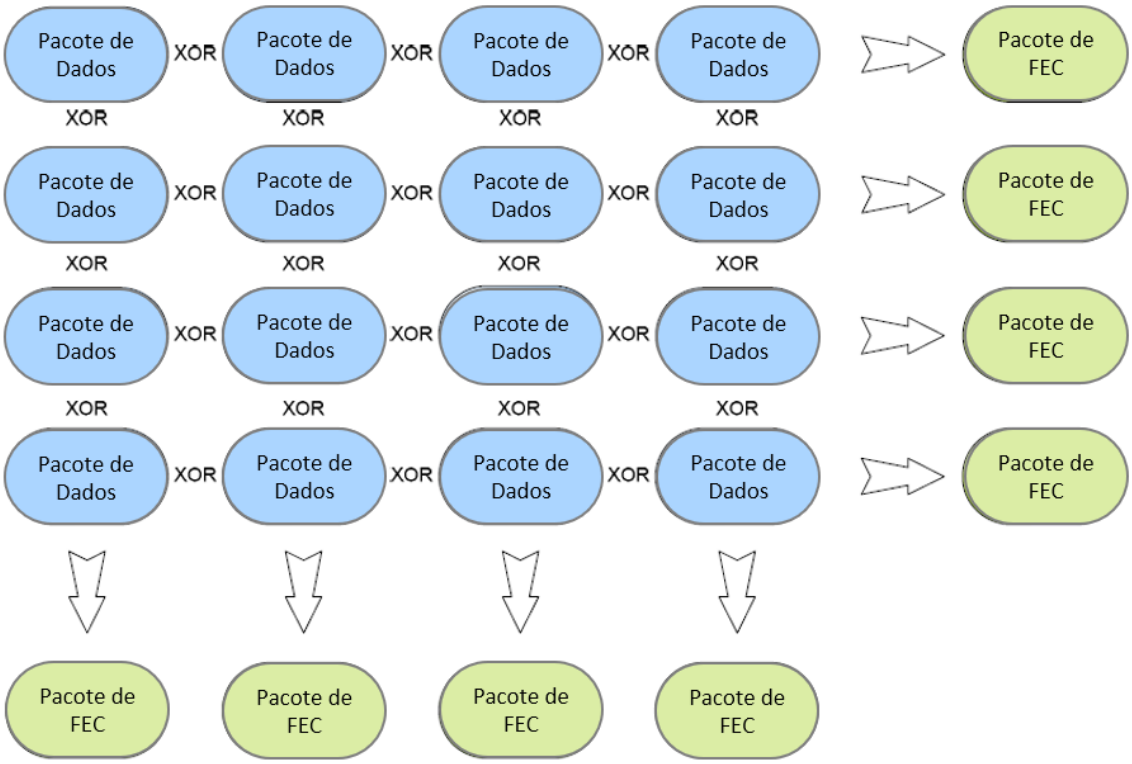


FIGURA 29: FEC EM DUAS DIMENSÕES

O envio constante de informações de redundância é custoso computacionalmente e gera um desperdício da capacidade do link, que poderia ser utilizado de forma mais eficiente aumentando a qualidade do vídeo, reduzindo custos ou empregando enlaces de menor capacidade. Para o Pro MPEG FEC com uma matriz de 25x4 é possível proteger uma perda em rajada de até 25 pacotes. Esse tamanho de matriz é capaz de proteger 100 pacotes, mas exige receber 129 pacotes para isso, 25 pacotes de FEC linha e 4 pacotes de FEC coluna. Ou seja, será acrescido 29% de gasto de banda, independente da presença ou ausência de erro. Esse processo gera um atraso na recepção e reprodução do vídeo no destino, pois depende que todos os pacotes da matriz, incluindo os pacotes de FEC, sejam recebidos, armazenados e processados para que seja possível corrigir algum erro. A partir de 5% de perda de pacote, espera-se impacto relevante em aplicações em tempo real como áudio e vídeo. Uma boa conexão de Internet possui perdas de aproximadamente 1% e esse valor pode chegar a 10% em uma conexão ruidosa. Para o mecanismo utilizado, é possível perder um pacote de linha e um pacote de coluna. Para ilustrar a Figura 30 explicita um exemplo de uma matriz 4x4.

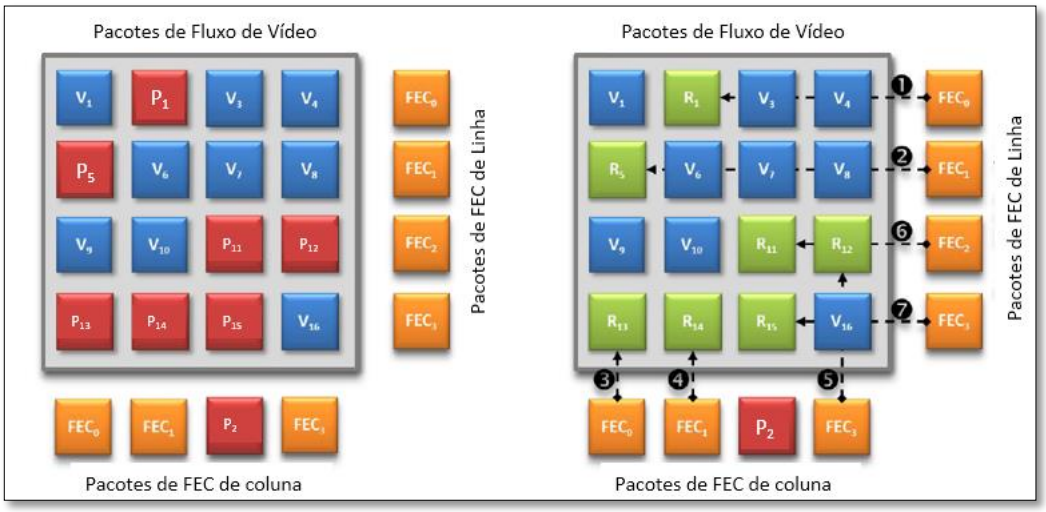


FIGURA 30: EXEMPLO DE PACOTES PERDIDOS NO CANAL E O RESULTADO DA RECUPERAÇÃO NO RECEPTOR.

O exemplo ilustra uma matriz com 4 linhas e 4 colunas com redundância de 8 pacotes de FEC. Para cada linha e coluna é aplicado o ou exclusivo entre todos os pacotes daquela linha ou coluna gerando um pacote resultante de redundância. Os pacotes azuis ilustram pacotes de

vídeo, os vermelhos os pacotes perdidos, os laranjas os pacotes de FEC e os verdes os pacotes recuperados.

A Figura 30 explicita na matriz da esquerda os pacotes perdidos pela interferência do canal e na matriz da direita o resultado da recuperação dos pacotes após processamento da redundância. O algoritmo inicia o escaneado primeiro das linhas, se existe apenas um pacote perdido na linha, o pacote é automaticamente recuperado. Se existe mais de um pacote perdido, então a linha é pulada até que se conclua o escaneamento. Após concluir-se o escaneamento de todas as linhas, o algoritmo continua com o escaneamento das colunas, note que neste caso R_1 e R_5 já foram recuperados no primeiro processo, ao escanear as colunas é possível recuperar também R_{12}, R_{13} e R_{14} pois são os únicos pacotes corrompidos em suas respectivas colunas neste momento. Como L_{11} e L_{15} ainda não foram recuperados é feito um novo escaneamento das linhas e na segunda rodada de escaneamento de linhas é possível recuperar R_{11} e R_{15}, totalizando 7 pacotes recuperados em sete interações, ou seja, neste exemplo foi possível reconstruir 43% de perda de pacotes. Na Figura 31, é possível observar um exemplo que com o mesmo número de pacotes perdidos não foi possível recuperar a informação total ao final de todas as tentativas.

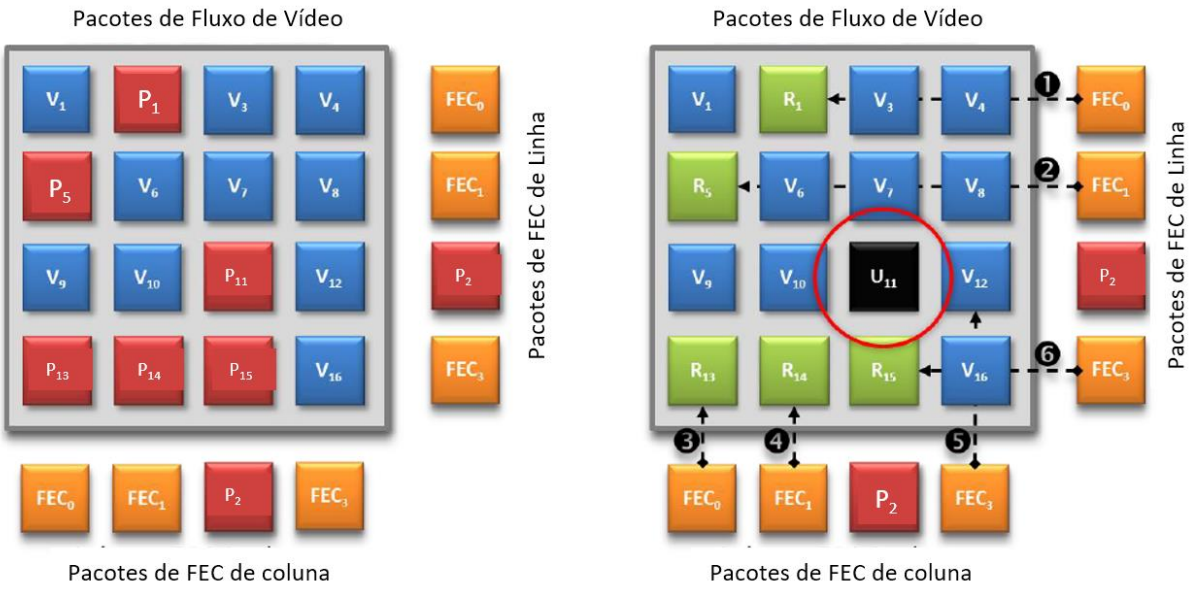


FIGURA 31 EXEMPLO DE PACOTES PERDIDOS E NÃO RECUPERADOS NO RECEPTOR.

O mesmo número de colunas/linhas e o mesmo algoritmo do exemplo anterior é utilizado, a diferença é que ao invés de perder dois pacotes de vídeo em uma linha, foi perdido um pacote

de vídeo e um pacote de redundância, impossibilitando desta forma a recuperação do dado de vídeo desta linha, sinalizado no exemplo como U_{11}.

A Figura 32 ilustra outro exemplo onde apenas quatro pacotes são perdidos e mesmo com todos os pacotes de FEC recebidos com integridade no destino não é possível recuperá-los pois em cada linha e em cada coluna houve perda de mais de um pacote [22]. Neste exemplo os pacotes 10, 11, 14 e 15 não são recuperados.

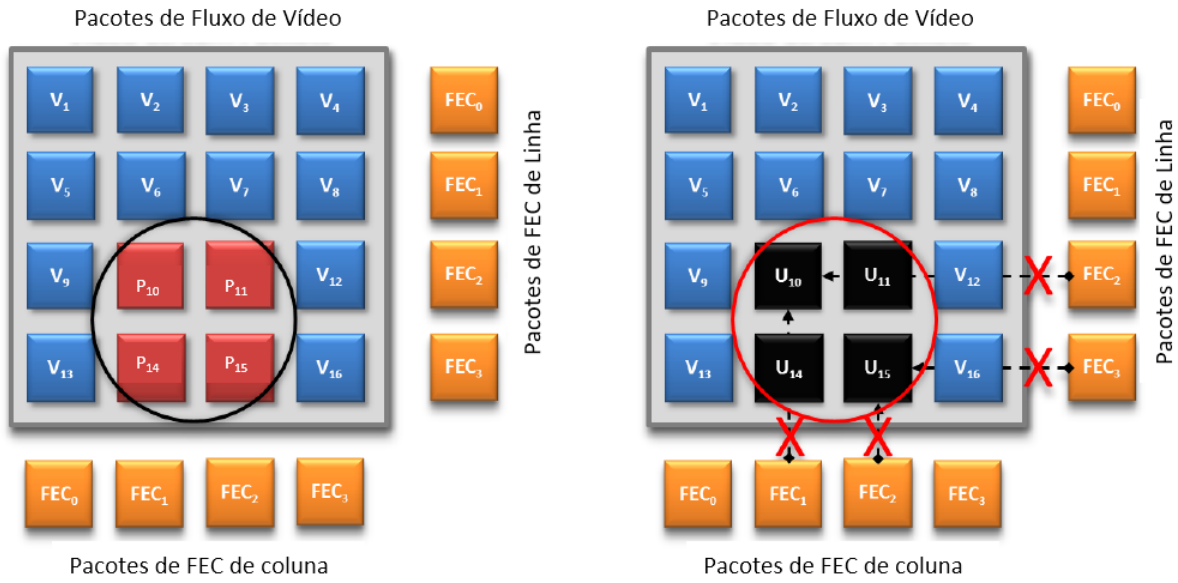


FIGURA 32 EXEMPLO DE PACOTES PERDIDOS E NÃO RECUPERADOS NO RECEPTOR.

Resultados indicam que uma matriz unidimensional permite uma boa correção de erros para um vídeo em MPEG-4 apenas quando a taxa de perda é muito pequena [11]. Para aumentar consideravelmente a capacidade de correção indica-se o uso de dois fluxos de FEC simultâneos [23]. Os streamings de FEC devem ser carregados em dois streamings distintos em portas de UDP separadas da porta que trafegará o streaming de mídia. Uma porta deve carregar o fluxo com a coluna de FEC e outra porta deve carregar o fluxo com a linha de FEC.

A Figura 33 resume para diferentes valores de L e D o impacto na sobrecarga, latência e capacidade de recuperação [22].

	Recovery (CoP)	Transmission Overhead	Buffer size, bytes. (CoP)	Buffer size in decoder, bytes.	Delay in decoder, ms.
XOR(5,10)	10 %	23 %	66400	195000	104
XOR(10,10)	10 %	16,7 %	132800	360000	192
XOR(20,5)	20 %	20 %	132800	375000	200
XOR(8,8)	12.5 %	20 %	84922	240000	128
XOR(10,5)	20 %	23 %	66400	195000	104
XOR(8,5)	20 %	24.5 %	53120	129000	68,8
XOR(5,5)	20 %	28.6 %	33200	120000	64
XOR(4,6)	16.5 %	29.4 %	31872	102000	54,4
XOR(6,4)	25 %	29.4 %	31872	102000	54,4

FIGURA 33: RELAÇÃO DE COMPROMISSO ENTRE PESO, LATÊNCIA E CAPACIDADE DE RECUPERAÇÃO

5. TRABALHOS RELACIONADOS

Códigos corretores de erros focados em implementações a nível de pacote são tema de estudo há algum tempo. O trabalho que é base para essa discussão foi desenvolvido em 1996 e é intitulado "*The case for packet level FEC*" [20]. Este trabalho discute em quais cenários é possível aplicar esse modelo de forma eficiente garantindo que o custo valha o investimento. O trabalho descreve pelo menos três aplicações em que esse modelo é efetivo. No caso de aplicações *multicast* para grupos grandes, mesmo no caso de taxas baixas de erro baixas por receptor resultaria em retransmissões para todo o grupo e o envio da redundância evitaria essas retransmissões. No caso de transmissões com alto atraso o uso da redundância permite controlar a taxa de erros dentro de um limite aceitável. Quando não é possível adicionar ao receptor memória suficiente para implementar técnicas sofisticadas de retransmissão [20].

O artigo discute as variações de redundância a nível de pacote considerando código de blocos, onde a transmissão de N pacotes de entrada é complementada por M pacotes de redundância. Se pelo menos N dos N + M pacotes for recebido corretamente então é possível recuperar os N pacotes originais [20].

A correção de erros a nível de pacote pode ser descrita pela equação abaixo, onde $X_{1..N}$ são pacotes de entrada e $Y_{1..M}$ são pacotes de redundância. É possível obter Y_j como umas função de $X_{1..N}$:

$$Y_j = \bigoplus \left(X_i \ll (i^{i-1} - 1) \right)$$

Para implementação deste recurso é necessário memória e capacidade de processamento computacional. O *host* de origem utiliza M adicionais *buffers* para computar redundância, o receptor deve dispor de N +M *buffers* para os casos de erros [20].

A redundância no formato de N + 1 pode dobrar a necessidade de processamento das melhores implementações TCP. O artigo considerou redundâncias no formato N+1, N+2 e N+3. Para simplificar assumiu-se que os erros são independentes, com redundâncias nos formatos: N+1, N+2 e N+3, a taxa de erros perseguida será uma função da taxa de erros de pacote como função da taxa de erros ϵ e do número N.

O resultado das taxas $\epsilon 1$, $\epsilon 2$ e $\epsilon 3$ podem ser calculadas conforme equações:

$$\epsilon_1 = \epsilon(1 - (1 - \epsilon)^N)$$

$$\epsilon_2 = \epsilon(1 - (1 - \epsilon)^{N+1} - (N + 1)\epsilon(1 - \epsilon)^N)$$

$$\epsilon_3 = \epsilon(1 - (1 - \epsilon)^{N+2} - (N + 2)\epsilon(1 - \epsilon)^{N+1} - \frac{(N + 2)(N + 1)}{2}\epsilon^2(1 - \epsilon)^N)$$

Em cada caso o *overhead* é M / (N+M) e é necessário avaliar os casos onde esse *overhead* vale a pena.

Uma fonte de transmissão envia pacotes para um determinado grupo usando serviço *multicast*. Assumimos que os erros ocorrerão sempre no último *path* de transmissão, que os erros são independentes e a probabilidade de um receptor não receber um dado pacote é ϵ [20].

A Figura 34 mostra a média de transmissões necessárias para um pacote ser recebido por um determinado grupo como função do tamanho do grupo, com interferência no canal gerando perda de 1%. A figura contém quatro curvas: uma curva sem redundância e as três demais com níveis de redundância: 7+1, 6+2 e 5+3. Quando a redundância é usada, ϵ é substituído por valores inferiores ao original 10^{-2}, como $6{,}8 \times 10^{-4}$, 2×10^{-5} ou $3{,}4 \times 10^{-7}$. Para poucos destinos o ganho da redundância não compensa o *overhead*. Pelas curvas podemos observar que a redundância mínima se torna mais eficiente que não ter redundância. Assim atinge-se o número de 18 destinos e o segundo nível de redundância se torna eficiente quando a curva corta 300 destinos. Um pacote será retransmitido sempre que for perdido por pelo menor um destino. A probabilidade desse evento ocorrer é:

$$1 - (1 - \epsilon)^G$$

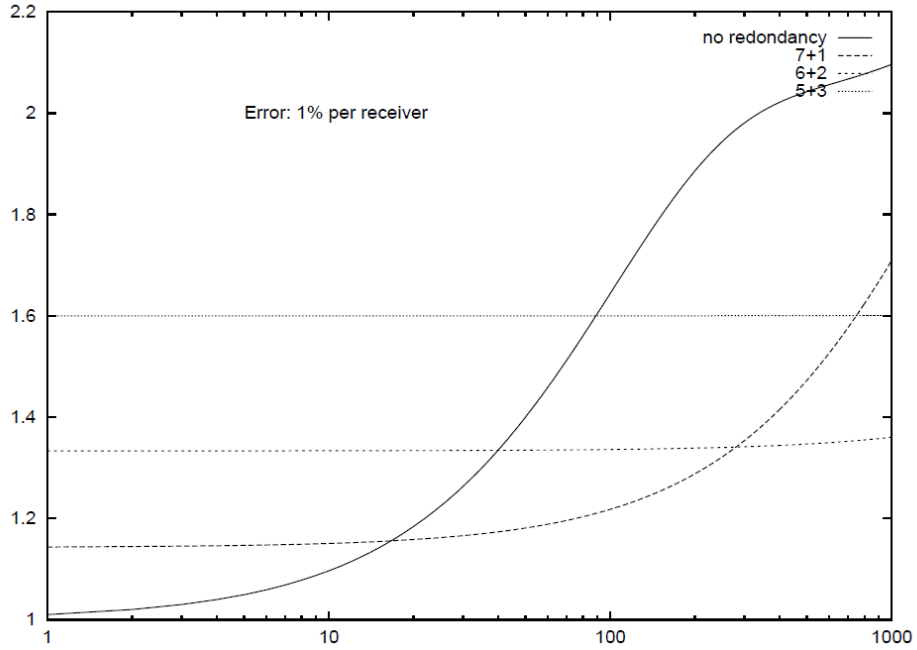


FIGURA 34: QUANTIDADE DE TRANSMISSÕES, COMO FUNÇÃO DO NÚMERO DE MEMBROS NO GRUPO COM TAXA DE ERRO DE PACOTE EM 1% [20].

Da mesma forma que no teste anterior, a Figura 35 plota o comportamento para quatro curvas sendo uma sem redundância e as demais com redundância (7+1, 6+2 e 5+3). Assumiu-se uma janela de transmissão de 200 pacotes correspondente a um salto de satélite ou redes de longa distância. O comportamento esperado é que a eficiência cresça como função da distância. Podemos observar que a taxa de redundância 5+3 resulta em maior estabilidade de latência mesmo com maiores taxas de erro [20].

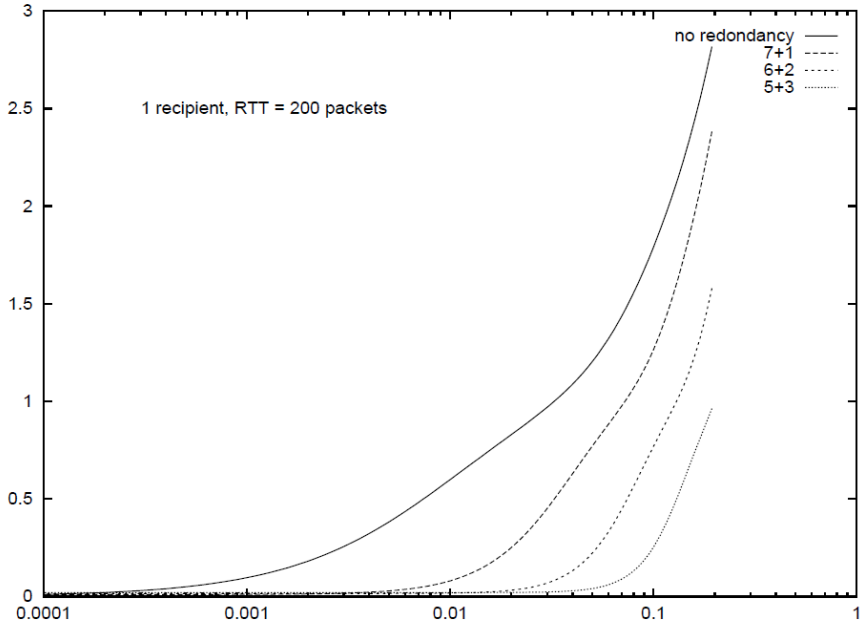


FIGURA 35: DELAY PELA TAXA DE ERRO EM UM RECEPTOR [20].

Um segundo trabalho que apoiou fortemente a metodologia e o desenvolvimento desta dissertação foi o "Projet d'approfondissement Master HES-SO Qualité de réception IPTV" desenvolvido por Rossier Jérémie [15]. Este trabalho buscou integrar o mecanismo de correção de erros descrito no SMPTE 2022 com o *player* de mercado VLC, este player também foi testado e utilizado no projeto telessaúde embora não tenha sido a solução final implementada devido ao *delay* que gerava [15].

O trabalho descreve os testes realizados com foco no padrão IPTV , como a correção de erros de transmissão baseadas em retransmissão não é útil possível para este tipo de aplicação, um dos métodos fornecidos para o aprimoramento de QoS é a correção de erros de transmissão SMPTE-2022 [15].

O foco do trabalho foi em descrever os passos necessários para o desenvolvimento do trabalho e serviu fortemente de base para o desenvolvimento do presente trabalho porém adaptado a linguagem python.

6. IMPLEMENTAÇÃO

Os mecanismos de correção de erros são amplamente implementados para todos os tipos de transmissão. É comum encontrarmos nos codificadores e decodificadores diferentes modelos de correção de erros implementados. Estes podem ser acrescidos nos pacotes ou no fluxo de dados e buscam transmitir a informação de um ponto a outro de forma robusta. Esse trabalho se propõe a implementar e analisar o mecanismo de correção de erros Pro Mpeg CoP#3, descrito nas seções anteriores, este mecanismo por sua simplicidade computacional é promissor em viabilizar a comunicação de áudio e vídeo no contexto de atendimento médico remoto proposto pelo projeto Sistema de Saúde Holográfica da Universidade Federal Fluminense.

As características aqui abordadas impactam diretamente no resultado, seja no custo, no desempenho, na qualidade e/ou na facilidade de implementação. Aplicações baseadas em vídeo são consideradas críticas quando pensamos em banda de transmissão, na qualidade e no atraso. Por isso este trabalho busca entender as características do canal e parametrizar o mecanismo de correção de erros escolhido adicionando robustez suficiente para garantir uma comunicação de qualidade no cenário de atendimento médico remoto especializado em áreas remotas com pouco recurso de infraestrutura.

A escolha de cada parâmetro deve ser analisada no caso a caso, e aqui emulamos em bancada um ambiente controlado, definindo as características do canal de transmissão e projetando o codificador para que garanta a entrega do conteúdo com qualidade no destino. Quanto maior a robustez imposta maior o uso da banda necessária, esse custo-benefício faz parte da avaliação, em cada projeto será importante definirmos quais as premissas da aplicação para configurar o codificador para atender a este escopo. Exemplo, para que não haja perda na qualidade podemos adicionar redundância aos dados transmitidos, buscando adicionar a robustez necessária para que não haja desperdício de banda, o que pode representar um superdimensionamento do link que pode não estar disponível da localização remota desejada. Neste capítulo é apresentada a implementação proposta e a parametrização estudada para alcançar as premissas definidas para levar a áreas carentes medicina especializada a áreas remotas através da transmissão de vídeo.

6.1. PROPOSTA

Os testes propostos neste trabalho foram realizados em um ambiente composto por dois *hosts* conectados por um cabo *ethernet* configurados para se comunicarem através de uma conexão local ponto a ponto. A primeira máquina com menor capacidade computacional foi

configurada para assumir o papel do transmissor. Esta possuía um software de codificação para áudio e vídeo com um algoritmo adicional para acréscimo dos fluxos de redundância.

A Figura 36 ilustra a topologia utilizada para realização dos testes com o codificador e decodificador implementados. Ambas as máquinas foram configuradas em uma rede 10.10.9.X/24 e todo o tráfego entre as máquinas era roteado para a porta ethernet “eth0”.

Utilizando o emulador de redes NetEM (*Network Emulator*), fomos capazes de emular o comportamento de perda desejado na portas de rede do *host* simulando um canal ruidoso. O objetivo foi avaliar a capacidade de correção do algoritmo quando ocorrem perdas de pacotes durante a transmissão no canal [26]. A perda de pacotes é especificada no comando ‘tc’ e o configurado deve é feito em porcentagem [27]. Utilizamos nos testes uma perda de 5%, nos testes observados percebemos que taxar maiores geravam perda de informação relevante ao vídeo. Esta taxa embora baixa é considerada uma perda comum em ambiente de redes, para adicionar essa perda à entrada do canal utilizamos o código abaixo:

```
sudo tc qdisc add/change dev eth0 root netem loss 5%
```

No *host* que foi configurado para ser o servidor do fluxo de dados, foi utilizada a biblioteca de mídias FFmpeg, que é uma poderosa ferramenta desenvolvida para ser utilizada em linha de comando e é totalmente *open source.* Essa ferramenta serve para converter, gravar e transmitir áudio e vídeo. Utilizando essa ferramenta, adicionamos os scripts nos dois *hosts* e configuramos para adicionar no fluxo de mídia codificado os pacotes de correção de erros.

Na nossa implementação utilizamos um vídeo de aproximadamente 30 segundos com tamanho de 5 MB (*Megabytes*). Todas as informações do vídeo original podem ser localizadas no Anexo II. Conforme código abaixo convertemos o vídeo original para uma taxa mais baixa, de modo a torna-lo computacionalmente mais leve devido a limitações de hardware na implementação que esta equiparado ao que se espera de uma implementação real em áreas remotas.

Desta forma conseguimos um cenário onde a transmissão seria suave para o transmissor e para o receptor evitando assim imputar erros aos testes causados pela falta de desempenho computacional dos *hosts*. O vídeo transmitido foi convertido para uma resolução de 800x600, com codec de H264, e taxa de vídeo máxima em 100 kbps, porém configurado como VBR (*Variable Bit Rate*), e o *frame rate* foi configurado para 5.

Foram testadas configurações com diferentes tamanhos de GOP (*Group of Pictures*) utilizando GOPs mais elevados com valores de 30, valores comuns de 15 até decidir-se pela configuração de trabalhar com *I-only*. Empiricamente o *I-only* foi o que apresentou melhor desempenho, embora a um custo de abrir mão de parte da qualidade. Desta forma independente da perda de pacote, isso não geraria uma perda de informação relevante no vídeo nos quadros seguintes e devido a necessidade de alta confiabilidade da transmissão da consulta médica a escolha foi por esta configuração. A escolha foi por um *frame rate* de 5, como o vídeo em uma consulta possui movimentos lentos e pouco variados, o arrasto entre quadros é pouco percebido e permite um relevante aumento na qualidade.

Essas alterações garantem que a transmissão ocorra com boa qualidade mesmo com recursos limitados. Os dados terão um pacote de correção de erros a cada N linhas e a cada M colunas e serão transmitidos através de um canal ruidoso por um fluxo de mídia RTP (*real time protocol*) via porta 8089. O primeiro fluxo de correção de erros é enviado totalmente apartado do fluxo original. Este é enviado 2 portas lógicas acima do fluxo de mídias, no exemplo dado será transmitido na porta 8091, esta porta lógica trafegará contendo os pacotes resultantes entre a combinação dos pacotes dispostas nas linhas. O segundo fluxo de correção de erros é resultante da combinação de pacotes das colunas e é enviado por uma terceira porta lógica, no nosso exemplo a porta 8098 [23]. O código abaixo determina as configurações utilizadas na nossa implementação onde basicamente variamos a configuração do mecanismo de correção de erros escolhido conforme sintaxe: “-fec prompeg=l=M:d=N*”*.

ffmpeg -re -i SampleVideo_1280x720_5mb.mp4 -intra -s 800x600 -pix_fmt yuv444p -vcodec copy -q 5 -b:a 50k -maxrate 100k -bufsize 500 -preset slow -benchmark -psnr -copyinkf -f rtp_mpegts -fec prompeg=l=6:d=4 rtp://10.10.9.2:8089

O fluxo configurado representa um *unicast*, ponto a ponto, ao atingir o receptor é decodificado através de um algoritmo que corrige os erros causados pelo canal ruidoso gerando um o fluxo resultante da combinação do tráfego dos três fluxos enviados nas portas: 8089, 8091 e 8093. Este fluxo é redirecionado em *loopback* no receptor e foi o fluxo de dados que gravamos para analisar e discutir os resultados objetivos na qualidade dos vídeos.

FIGURA 36: TOPOLOGIA UTILIZADA NA IMPLEMENTAÇÃO;

Buscando estabelecer conclusões referentes aos melhores *setups* para solucionar problemas de interferência no canal e entender qual o limiar de correção de erros que podemos utilizar, realizamos diferentes avaliações neste trabalho: avaliações de desempenho, latência, custo para a rede e qualidade do vídeo foram considerados. A ideia foi comparar o material original (antes de sofrer qualquer processamento), o material codificado, porém ainda não enviado e o material resultante após sofrer a incidência dos ruídos adicionados ao canal, considerando sua chegada no receptor e a etapa de decodificação e correção de erros. Para garantir um ambiente controlado e capaz de remover ao máximo, a aleatoriedade do processo utilizamos um cenário pequeno e ferramentas de auxílio a repetição dos testes.

Utilizamos para análise do tráfego de dados as ferramentas TCPDump e Wireshark e para garantir que cada fluxo de dados gerado pudesse ser cuidadosamente avaliado, foi utilizado a aplicação TCP Replay. Os dados capturados pelo wireshark foram gravados em um arquivo de extensão .pcap. Este arquivo pôde ser reproduzido posteriormente utilizando a ferramenta TCP Replay. A TABELA 4 mostra o descritivo dos testes realizados e analisados em detalhes. A perda configurada foi a de 5% e configuramos a correção de erros com diferentes níveis de redundância, abaixo para ilustrar vamos analisar 4 configurações para os parâmetros linha e coluna da FEC.

FEC	Linha	Coluna	Arquivo Resultante
N	0	0	5LOSS

S	4	4	44_5LOSS
S	6	4	64_5LOSS
S	8	5	85_5LOSS
S	10	5	105_5LOSS

TABELA 4: CENÁRIOS CONFIGURADOS PARA OS TESTES.

As configurações escolhidas foram baseadas nos maiores percentuais de recuperação esperados. No caso do FEC(4, 4) adiciona-se a maior redundância possível pela menor quantidade de dados. No caso da configuração com FEC (6,4) a taxa de recuperação esperada é de 25%. A configuração FEC (8,5) e FEC(10,5) prevê uma taxa de recuperação de 20%. A Figura 37 explicita as diferentes taxas de recuperação descritas na recomendação PRO-MPEG CoP3 [23].

	Recovery (CoP)
XOR(5,10)	10 %
XOR(10,10)	10 %
XOR(20,5)	20 %
XOR(8,8)	12.5 %
XOR(10,5)	20 %
XOR(8,5)	20 %
XOR(5,5)	20 %
XOR(4,6)	16.5 %
XOR(6,4)	25 %

FIGURA 37: RELAÇÃO DO TIPO DE REDUNDÂNCIA PELA CAPACIDADE DE RECUPERAÇÃO PREVISTA [23].

6.2. ANÁLISE DOS RESULTADOS

A configuração utilizada contemplou uma perda de 5% dos pacotes transmitidos no canal, esta perda foi escolhida por ser comumente encontrada em redes IP e adicionalmente foi

a que apresentou os melhores resultados durante os testes em que a taxa sofreu variação entre 0 e 10% de perda.

Os dados transmitidos no canal e os resultados estão ilustrados na TABELA 1Tabela 5**Erro! Fonte de referência não encontrada.** e na Figura 38. A quantidade de informação redundante enviada é maior com linha e coluna configurados para o valor de 4. Esta quantidade de dados cai proporcionalmente quando a relação entre os pacotes de redundância enviados reduz em relação aos pacotes de mídia enviados.

Nas diferentes configurações de correção de erros podemos observar que uma quantidade relevante de dados é transmitida para garantir a correção esperada. No caso da configuração de FEC(4,4) o *overhead* é de 49,9%. Na configuração de FEC (6,4) enviamos 41,6% de overhead. Na opção em que FEC (8,5) enviamos 31,2% a mais de informação e no caso da configuração de FEC (10,5) enviamos 29,8% de informação adicional. É possível calcular o percentual matemático de recuperação esperado em cada configuração através da relação quantidade de erros máxima que pode ser perdida, considerando que haverá apenas uma etapa de correção, sobre a quantidade total de arquivos de mídia transmitido. Na configuração de FEC (6,4) e FEC (4,4) espera-se corrigir no máximo 25% dos erros, este caso é atingido quando há uma perda em rajada de todos os pacotes da linha. Na opção em que FEC (8,5) e FEC (10,5) espera-se corrigir no máximo 20% dos erros também considerando a situação de perda em rajada.

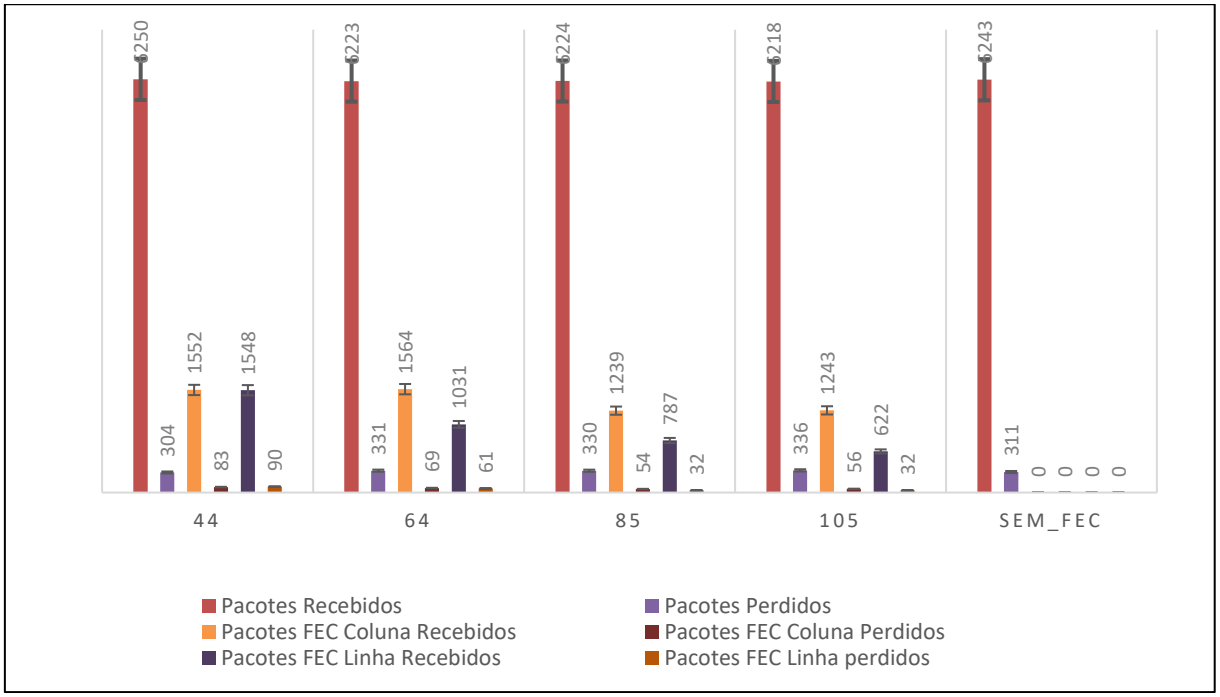


FIGURA 38: COMPARAÇÃO DA QUANTIDADE DE PACOTES ENVIADOS E RECEBIDOS POR TIPO

TABELA 5: ANALISE DAS INFORMAÇÕES DE REDUNDÂNCIA INSERIDAS E CORRIGIDAS. (PARTE 1)

Parâmetros				Total		Mídia				
FEC	Linha	Coluna	Perda	Total de Pacotes	Overhead	Recebidos	Perdido	Taxa de Perda	Recuperado/Perdido	% Recuperado/Perdido
N	0	0	5%	6554	0,0%	6243	311	4,75%	0/311	0,00%
S	4	4	5%	9827	49,9%	6250	304	4,64%	304/304	100,00%
S	6	4	5%	9279	41,6%	6223	331	5,05%	330/331	99,70%
S	8	5	5%	8666	32,2%	6224	330	5,04%	328/330	99,39%
S	10	5	5%	8507	29,8%	6218	336	5,13%	331/336	98,51%

As figuras 38 e 39 ilustram os resultados obtidos nos testes. Para obtenção desses números foram realizadas cinco simulações para cada configuração definida e a tabela expõe a média entre os resultados obtidos. Podemos observar em quantidade e percentual os pacotes recebidos e perdidos de mídia, de redundância aplicadas às linhas e de redundância aplicada às colunas. Notamos que nos quatro cenários obtivemos uma recuperação superior a 97%. Em uma escolha qualitativa optamos por considerar ideal recuperações superiores a 99% de modo a garantir que não haja nenhum defeito no vídeo.

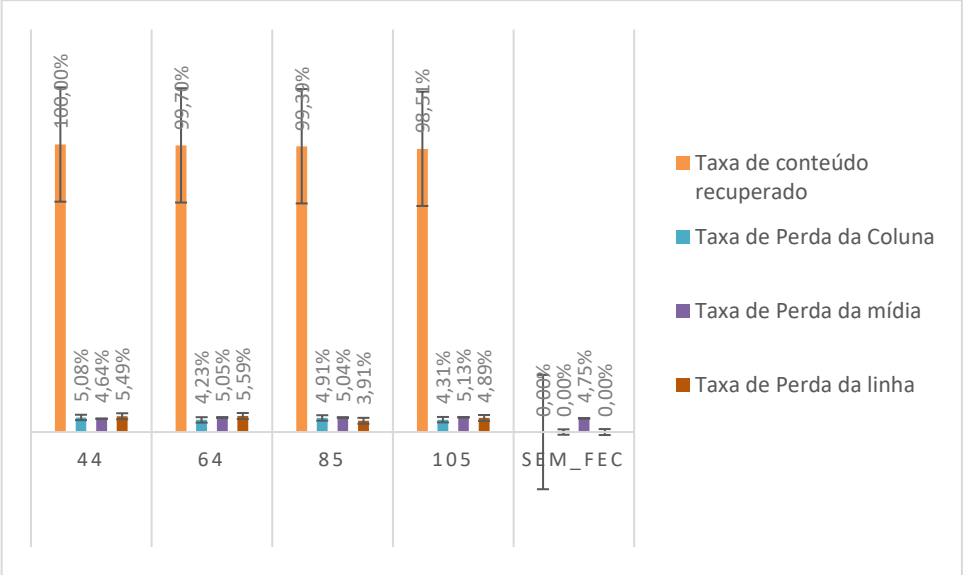


FIGURA 39: PERCENTUAL DE PACOTES ENVIADOS E RECEBIDOS DE MÍDIA, L E D.

Utilizamos player de vídeo VLC para apoiar no processo de gravação e avaliação qualitativa do conteúdo. Através da avaliação subjetiva foi possível confirmar que a correção foi realmente efetiva e resultou em melhoria significante na qualidade do vídeo percebida pelo usuário final.

Para reforçar tal análise utilizamos também duas métricas objetivas de avaliação o SSIM (*Structural Similarity Index*) e o PSNR (*Peak-Signal-to-Noise-Ratio*). O PSNR é uma métrica muito utilizada na indústria para avaliação objetiva da qualidade, esta métrica é derivada do MSE (*Mean Squared Error*) e está descrita conforme equações expostas a seguir.

$$PSNR_{dB} = 10\log\frac{10(2^n - 1)^2}{MSE}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}[f(i) - g(i)]^2$$

Em que f(i) é o sinal original e g(i) é o sinal degradado, n é o número de bits/amostra e N o número de amostras. O resultado é um número em decibéis dB, geralmente variando entre 30 e 40 sendo o máximo igual a 50dB, onde quanto maior o valor, melhor a qualidade do vídeo [28].

A métrica PSNR não busca avaliar a qualidade do vídeo no receptor e sim a integridade dos dados enviados e recebidos, comparando a diferença entre os pixels da imagem original e da imagem resultante. Por isso o PSNR pode fazer uma avaliação indesejada quando comparamos um vídeo com variação de luminância e um vídeo com ruído por exemplo. O vídeo com ruído será considerado melhor do que o com alteração na luminância pois o ruído pode ser encontrado em um percentual de pixels do vídeo e a luminância altera todos os pixels do mesmo. A Figura 40 ilustra a comparação entre três imagens, no qual o PSNR da imagem ruidosa é

superior ao da imagem com variação de luminância e por isso para essa métrica é considerado melhor.

FIGURA 40: COMPARAÇÃO ENTRE O PSNR DA IMAGEM ORIGINAL COM UMA RUIDOSA E UMA COM VARIAÇÃO DE LUMINÂNCIA [28].

A avaliação PSNR aplicada na nossa implementação mostra que o vídeo com 5% de erro resultou em um PSNR médio de 15,47 e com a inclusão do mecanismo de correção de erros FEC(4,4) o PSNR passa a ser 34,25 dB, FEC(6,4) o PSNR passa a ser 33,66 dB, FEC(8,5) a PSNR passa a ser 23,83 dB e FEC(10,5) a PSNR passa a ser 23,13 dB. Reforçando a conclusão exposta na figura 37 de que a melhor correção é encontrada na implementação com FEC(4,4) e a pior é a FEC(10,5) [29].

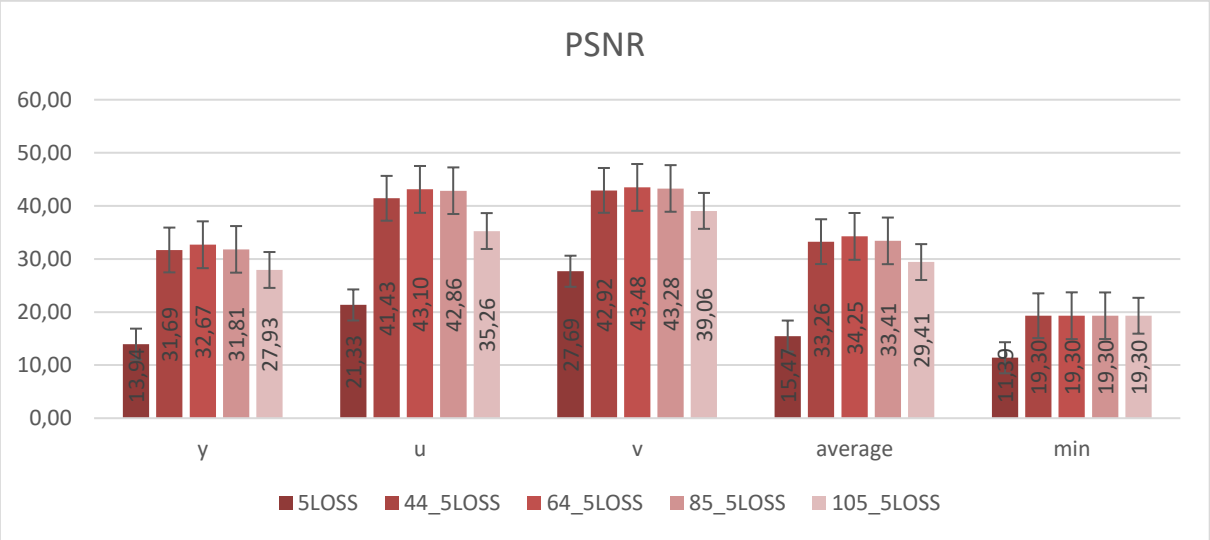


FIGURA 41: AVALIAÇÃO COM MÉTRICA PSNR.

O segundo método de avaliação objetiva da qualidade do vídeo utilizada é o SSIM (*Structural Similarity Index*). Esta análise traz uma abordagem diferente para a qualidade de

vídeo pois se propõe a fazer uma avaliação baseada da medição estrutural da distorção. O objetivo desse método é avaliar algo mais próximo do que o olho humano percebe, ou seja, esse modelo de avaliação busca fazer uma melhor correlação com a subjetividade de uma avaliação de qualidade considerando contraste, luminância e textura. Sendo x= $\{x_i | i = 1,2,...,N\}$ o sinal original e y = $\{y_i | i = 1,2,...,N\}$ o sinal distorcido, calculamos o SSIM conforme exposto abaixo:

$$SSIM = \frac{(2\overline{xy} + C_1)(2\sigma_{xy} + C_2)}{[(\bar{x})^2 + (\bar{y})^2 + C_1](\sigma_x^2 + \sigma_y^2 + C_2)}$$

Nesta equação $\bar{x}, \bar{y}, \sigma_x, \sigma_y, \sigma_{xy}$ são os valores estimados respectivamente para a média do sinal original x, média do sinal distorcido y, a variância de x, a variância de y e a covariância entre x e y. C1 e C2 são constantes. O valor de SSIM varia entre -1 e 1 e tem seu melhor valor em 1 se x_i igual a y_i em todos os valores de i. Na Figura 42 podemos observar a imagem qualitativamente e os respectivos valores de SSIM calculados [28].

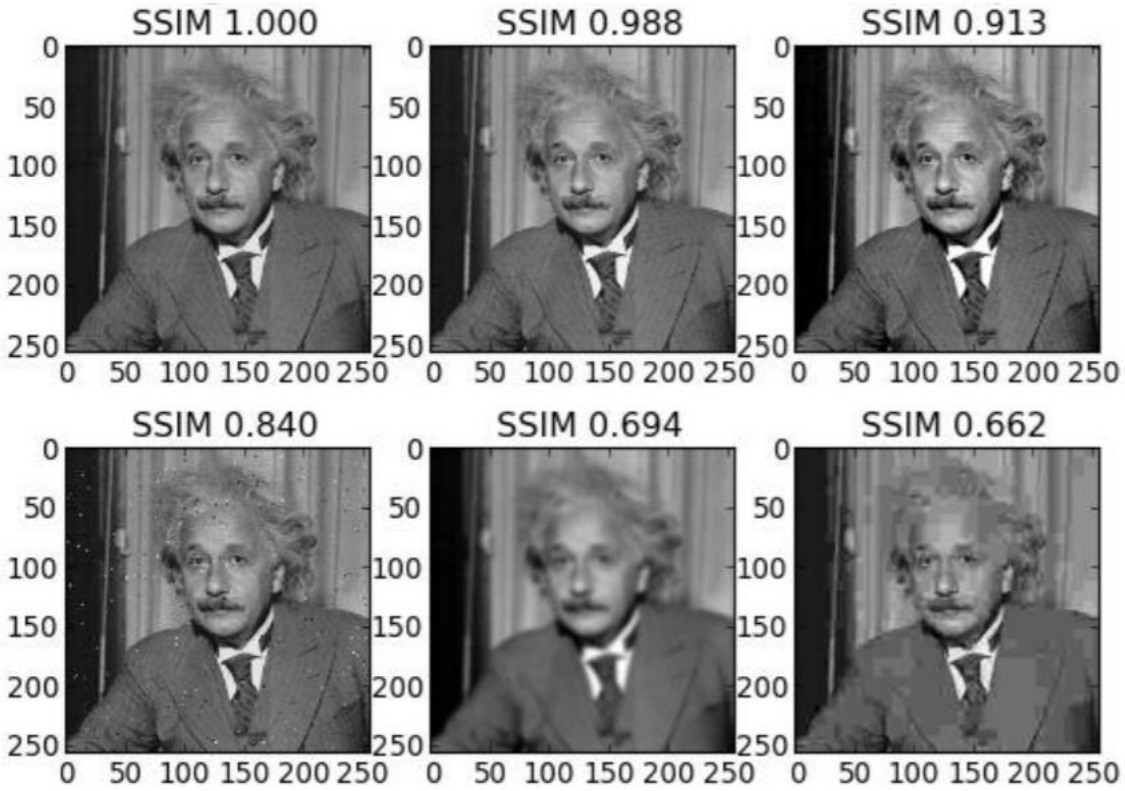


FIGURA 42: COMPARAÇÃO ENTRE DIFERENTES AVALIAÇÕES DE SSIM [28].

A avaliação SSIM aplicada a nossa implementação mostra que o vídeo com 5% de erro resultou em um SSIM de 0,756 e com a inclusão do mecanismo de correção de erros FEC(4,4) a SSIM passa a ser 0,984. Para a FEC(6,4) a SSIM passa a ser 0,982. Para FEC(8,5) a SSIM passa a ser 0,956. Finalmente, para FEC(10,5) a SSIM passa a ser 0, 929. Estes dados estão sumarizados na FIGURA 43.

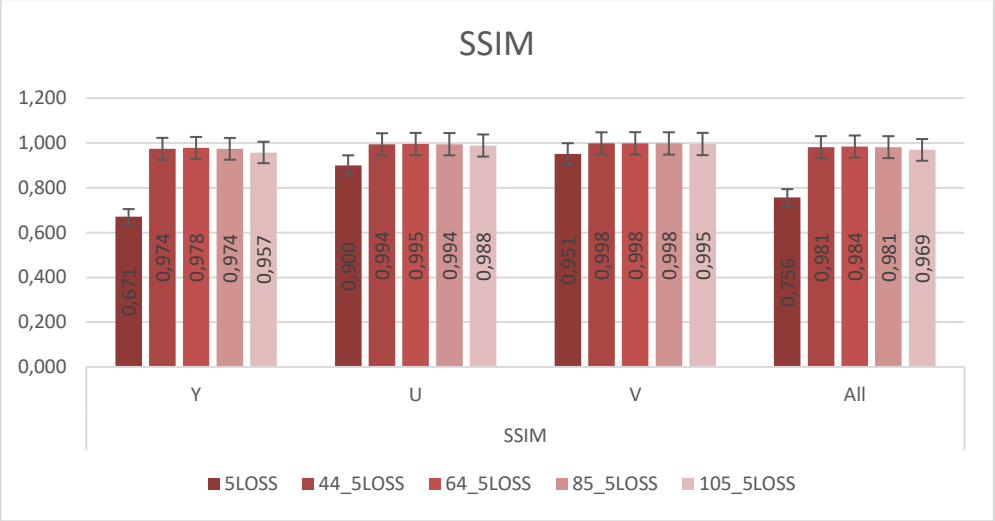


FIGURA 43:AVALIAÇÃO COM MÉTRICA SSIM

A mesma avaliação pode ser feita qualitativamente no conjunto de figuras exposto a seguir. Na Figura 44, é possível observar o impacto nas imagens quando o vídeo sofre uma perda de pacotes igual a 5%. Essa avaliação pode ser feita mesmo sem o uso de ferramentas objetivas de qualidade pois a perda é facilmente percebida em uma avaliação subjetiva. Neste caso o resultado do PSNR entre o vídeo original e o vídeo resultante foi de 15,37 dB para um vídeo com 5% de perda de pacote sem correção de erros e no caso do SSIM de 0,76. O valor mínimo do PSNR para manter o vídeo inteligível e não causar impactos ao telespectador é de 30 dB.

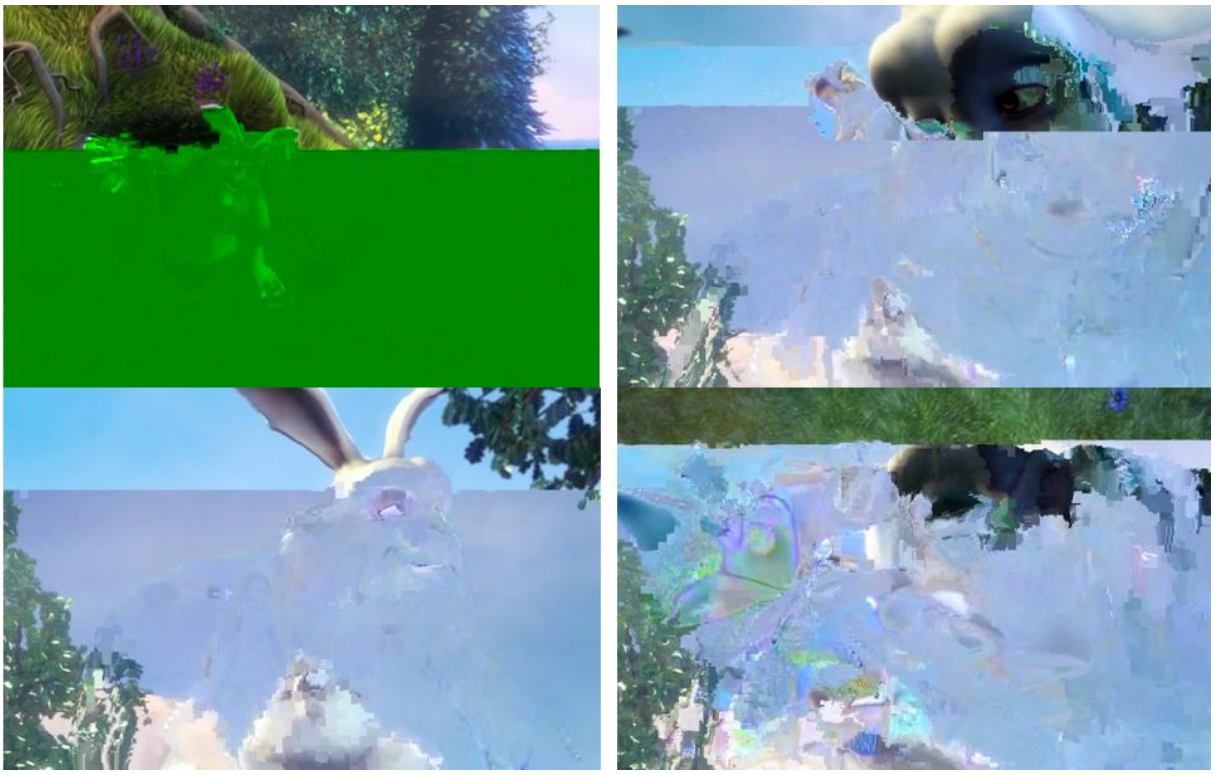


FIGURA 44: VÍDEO COM 5% DE ERRO SEM CORREÇÃO DE PACOTE (PSNR = 15,37; SSIM= 0,75592)

Ao acrescentarmos o mecanismo de correção de erros de pacote nesta transmissão com 5% de perda de pacote no canal, configuramos o codificador para incrementar um pacote de redundância a cada 4 linhas e a cada 4 colunas (L= 4 e D = 4). É possível notar na avaliação subjetiva exposta na Figura 45 que a informação é totalmente recuperada. Com essa configuração, é possível atingir uma taxa de correção de erros em mídia útil igual 100%.

Objetivamente o valor de PSNR e SSIM também confirmam que a taxa de erros e a experiência de usuário é restaurada. O valor do PSNR ultrapassa o corte de 30 dB (PSNR = 32,7 dB) e o valor do SSIM se aproxima de 1 (SSIM= 0,984). Notamos que as imagens possuem boa qualidade e conseguem transmitir toda informação relevante para compreensão do conteúdo.

FIGURA 45: VÍDEO COM 5% DE ERRO E FEC COM L=D=4. (PSNR = 34,250133 , SSIM= 0,984083 E TAXA DE CORREÇÃO = 100%)

Em um segundo teste modificamos os parâmetros configurados para incrementar pacotes de redundância a cada 6 linhas e a cada 4 colunas (L= 6 e D = 4). Na avaliação subjetiva exposta na Figura 46 podemos observação que essa correção é muito eficiente embora ainda seja possível observar *artefatos* e macro blocos no vídeo mas com baixo impacto na compressão do conteúdo. Com essa configuração foi observada uma taxa de correção de erros igual 99,7%, objetivamente o valor de PSNR e SSIM também confirmam que a taxa de erros e a experiência dos usuários são praticamente restauradas. O valor do PSNR atinge 33,66 dB e o valor do SSIM atinge 0,982.

FIGURA 46 VÍDEO COM 5% DE ERRO E FEC COM L=6 E D=4. (PSNR = 33,660425, SSIM= 0,982667 E TAXA DE CORREÇÃO = 99,7%)

Em um terceiro teste modificamos os parâmetros configurados para incrementar um pacote de redundância a cada 8 linhas e a cada 5 colunas (L= 8 e D = 5). Na avaliação subjetiva exposta na Figura 47 validamos a redução na capacidade de recuperação e o impacto subjetivo gerada. Com essa configuração foi observada uma taxa de correção de erros igual 99,4%, objetivamente o valor de PSNR é de 23,83dB e o valor de SSIM atingiu 0,956, o que ilustra uma melhora significativa na transmissão do vídeo, mas ainda gera perda de qualidade e conteúdo causando impacto na experiência do usuário. Notamos que as imagens podem ser compreendidas, porém a presença de artefatos e a perda de qualidade gera desconforto ao telespectador.

FIGURA 47: VÍDEO COM 5% DE ERRO E FEC COM L=8 E D=5. (PSNR = 23,827446 , SSIM= 0,956 E TAXA DE CORREÇÃO = 99,39%)

Em um último teste modificamos os parâmetros configurados para incrementar um pacote de redundância a cada 10 linhas e a cada 5 colunas (L= 10 e D = 5). Na avaliação subjetiva exposta na Figura 48 validamos que esta configuração não atinge uma qualidade ótima, a taxa de correção de erros medida é igual 98,51%, objetivamente o valor de PSNR é 20,13 dB e o valor de SSIM atingiu 0,93. Há uma melhora relevante na imagem porém ainda há muita perda de informação e qualidade. Essa configuração gera grande impacto na experiência do usuário final por isso essa configuração não atingiu as expectativas da aplicação para garantir qualidade e integridade da informação.

FIGURA 48: VÍDEO COM 5% DE ERRO E FEC COM L=10 E D=5. (PSNR = 20,133941, SSIM= 0,929065 E TAXA DE CORREÇÃO = 98,51%)

A tabela 7 explicita os atrasos medidos em cada caso, entre o transmissor e o receptor, esse atraso é na ordem de 1 segundo e o uso do FEC adiciona alguns milissegundos a este atraso não sendo considerado impactante.

TABELA 6: ATRASO NA RECEPÇÃO

FEC	Linha	Coluna	Atraso (seg)
N	0	0	00:00:01;14
S	4	4	00:00:01;17
S	6	4	00:00:01;18
S	8	5	00:00:01;18
S	10	5	00:00:01;24

7. CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho a parametrização do codificador, do mecanismo de correção de erros e das possíveis interferências ao canal foram foco de estudo. As experimentações realizadas possibilitaram definir uma parametrização para o codificador que garantisse uma comunicação íntegra mesmo em meios ruidosos. Avaliamos a qualidade do vídeo, o atraso, o custo computacional e o *overhead* gerado pelo mecanismo de correção de erros implementado Pro-MPEG Code of Practice #3 (CoP3).

O codificador foi parametrizado para atingir uma boa comunicação em um canal com perdas randômicas na ordem de 5%. A parametrização permitiu eliminar 100% dos erros do canal. Foi necessário fazer adequações no codificador de vídeo e no mecanismo de correção de erros. O codificador foi reconfigurado e a taxa de frame foi reduzida de 30 frames por segundo para 5 frames por segundo. Embora essa alteração tenha um impacto na suavidade das transições, este impacto não é tão relevante para a aplicação de vídeo conferência onde não se espera transições de cenário e movimentos bruscos e rápidos. A resolução e a taxa de bits também foram consideravelmente reduzidas, como a imagem de um consultório possui um fundo estático e homogêneo, é possível abrir mão de algum nível de nitidez nos contornos e nas cores.

A configuração do tamanho da matriz de correção de erros permitiu reconstruir completamente o vídeo inicialmente deteriorado pelo ruído do canal. Realizamos testes a uma taxa de erro de aproximadamente 5%, e foi possível reconstruir totalmente o vídeo porém com o impacto de adicionar um *overhead* de aproximadamente 50%. Essa sobrecarga de dados adicionada para inserir a redundância poderia ser convertida em aumento da qualidade em um ambiente de menor ruído no canal. Entretanto, como o foco do nosso trabalho é garantir uma comunicação de qualidade para regiões remotas e com recursos limitados de infraestrutura esse mecanismo poderá viabilizar o acesso. Em canais com perdas superiores a 5% o mecanismo também se provou eficiente e apresentou boa recuperação, superiores a 95% de correção dos erros, porém tal perda para aplicações de vídeo já gera impacto visual e resulta em um vídeo com incidência de macro blocos e artefatos. Além do acréscimo de *overhead* um pequeno acréscimo de latência foi medido. A diferença de latência gerada pelo processamento de codificar e decodificar o mecanismo de correção de erros é na ordem de 10 frames, ou seja, inferior a 1 segundo. Empiricamente, observamos que o custo de *overhead* nos testes realizados varia de 20% a 50% e resultam em perdas inferiores a 1%. Ou seja, na pior configuração de

FEC avaliada, um canal com 5% de ruído, o que gera uma perda de 336 pacotes, é capaz de corrigir 331 pacotes, resultando em uma perda final de 0,08%.

Os resultados dos testes foram satisfatórios e avaliando a qualidade dos vídeos recebidos no receptor e a reconstrução obtida validamos a importância e a relevância do uso dos mecanismos de correção de erros para transmissão de vídeo em meios com muita perda. Esta implementação pode ser uma ferramenta viabilizadora das implementações do projeto Sistema de Saúde Holográfica da Universidade Federal Fluminense.

Em trabalhos futuros seria interessante testar para atendimentos médicos especializados em áreas remotas, o uso dos mecanismos de correção de erros implementando a aplicação em dispositivos móveis, com o advento das redes de comunicação sem fio, implementações como o 5G podem alavancar o acesso a conectividade a áreas remotas.

Outra opção de trabalho futuro é avaliar e registrar as estatísticas da rede sem fio que é ainda mais suscetível a ruído e implementar um algoritmo de aprendizado que consiga decidir estatisticamente qual a melhor taxa de redundância de pacotes que deve ser utilizada para garantir a completa recuperação dos pacotes perdidos. Utilizando um algoritmo que insira sob demanda os pacotes de redundância seria possível utilizar de forma eficiente a capacidade do canal.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MURPHY, Julia; ROSER, Max. Internet. 2018. Disponível em: <https://ourworldindata.org/Internet>. Acesso em: 15 fev. 2018.

[2] STATISTA. 2018. Disponível em: <https://www.statista.com/statistics/273018/number-of-Internet-users-worldwide/>. Acesso em: 15 fev. 2018.

[3] INTERNET World Stats. 2018. Disponível em: <http://www.Internetworldstats.com/emarketing.htm>. Acesso em: 15 fev. 2018.

[4] STATISTA. 2017. Disponível em: <https://www.statista.com/statistics/499431/global-ip-data-traffic-forecast/>. Acesso em: 15 fev. 2018.

[5] THE ZETTABYTE Era: Trends and Analysis. 2017. Disponível em: <http://ttps://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.htm>. Acesso em: 15 fev. 2018.

[6] DIAZ, César; CABRERA, Julián; JAUREGUIZAR, Fernando. Enhancement of pro-mpeg cop3 codes and application to Layer-aware fec protection of two-layered video transmission". 2013 IEEE International Conference on Image Processing, ICIP 2013 Proceedings, pp. 1598 – 1602

[7] POON, Ting-Chung; LIU, Jung-Ping. Introduction to Modern Digital Holography with Matlab. United Kindom: Cambridge University Press, 2014. 248 p.

[8] SHELLINGTON, Kylie. Creating the blind pepper's ghost: analyzing sound spectra using fourier transforms to predict dimensional variation for image reconstruction. 2015. 102 p. Monografia (Bacharelado em Ciência da Computação)- Stetson University, Flórida, EUA, 2015.

[9] BINDER, Oliver. Interactive Rendering For Projection-Based Augmented Reality Displays. 2002. 214 p. Tese (Doutorado em Ciência da Computação)- Stetson University, Technische Universitä Darmstadt, Daaden, Alemanha, 2002.

[10] BRANDERUD, Anders. Adaptive FEC-Encoding for Video Streaming over WiFi. 2013. 34 p. Dissertação (Mestrado em Ciência da Computação)- Stockholm, Suíça, 2013. **[11]** Maria Baltzer Pedersen, "FEC on IP-output for video encoder", 2006.

[12] AWWAL, Ibrahim Sariyya. Simulation Of Error Correction Codes On Wireless Communication Systems. 2013. 63 p. Dissertação (Mestrado em Engenharia Elética)- Ahmadu Bello University, Zaria, Nigéria, 2013.

[13] IBRAHIM, Mohamed Elsir. Implementation And Analysis Of Forward Error Correction Techniques. 2013. 99 p. Dissertação (Bacharelado em Engenharia Elética)- University of Khartoum, Cartum, Sudão, 2013

[14] KUMAWAT, Himmat Lal; SHARMA, Sandhya. Implementation of a forward error correction technique using convolution encoding with viterbi decoding. International Journal of Soft Computing and Engineering, [S.l.], v. 2, n. 5, p. 95-99, nov. 2012. Disponível em: <https://www.ijsce.org/>. Acesso em: 01 mar. 2018.

[15] ROSSIER, Jérémie. Projet d'approfondissement Master HES-SO. 2011. 43 p. Dissertação (Mestrado em Engenharia Elética)- University of Applied Sciences Western Switzerland, [S.l.], 2011.

[16] STOCKHAMMER, Thomas et al. Application Layer Forward Error Correction for Mobile Multimedia Broadcasting Case Study. 2009. Disponível em: <https://www.qualcomm.com/media/documents/files/raptor-codes-for-mobile-multimedia-broadcasting-case-study.pdf>. Acesso em: 01 mar. 2018.

[17] ROSENBERG, Julius; SCHULZRINNE, Henning. RFC 2733: An RTP Payload Format for Generic Forward Error Correction. [S.l.: s.n.], 1999. 26 p. Disponível em: <https://tools.ietf.org/html/rfc2733>. Acesso em: 01 mar. 2018.

[18] HAYKIN, Simon. Communication System. 4. ed. [S.l.]: John Wiley And Sons Inc, 2001. 838 p.

[19] PROAKIS, John G. Digital Communication. 4. ed. [S.l.]: McGrawHill, 2000. 938 p.

[20] HUITEMA, Christian. The case for packet level FEC. In: International Workshop on Protocols for High Speed Networks, 2002, Berlin, Germany. th IFIP/IEEE International Workshop... Berlin, Germany: Springer, 2002. p. 109-120.

[21] SOHN, Yejin; HWANG, Jun; KANG, Seung-Seok. Adaptive Packet-Level FEC Algorithm for Improving the Video Quality over IEEE 802.11 Networks. International Journal of Software Engineering and Its Applications, Canadá, v. 6, n. 3, p. 27-34, jan. 2012.

[22] ROZENBERG, Adi. The Hidden Limitations of FEC.

[23] FORUM, Pro Mpeg. Transmission of professional MPEG-2 Transporte Streams Over IP Networks. [S.l.: s.n.], 2004. 15 p. **[24]** Tektronix (2009a). Um Guia para Fundamentos de MPEG e Análise de Protocolo. Tektronix.

[25] TEKTRONIX. Video Group. Um Guia para Fundamentos de MPEG e Análise de Protocolo.

[26] BROWN, Martin A. Linux Traffic Control. [S.l.: s.n.], 2016. Disponível em: <http://www.tldp.org/HOWTO/html_single/Traffic-Control-HOWTO/>. Acesso em: 01 mar. 2018.

[27] CRANDALL, Jedidiah. Traffic Control Manual For Lab1 [S.l.: s.n.], 2013. Disponível em: <http://www.cs.um.edu/>. Acesso em: 01 mar. 2018.

[28] WANG, Yubing. Survey of Objective Video Quality Measurements. 2006.

[29] UNG, Agata. Comparison of Video Quality Assessment Methods. 2017. 49 p. Comparison of Video Quality Assessment Methods (Mestrado em Engenharia Elética)- Blekinge Institute of Technology, [S.l.], 2017.

[30] SHIH, Chi-Huang. Enhancing Packet-level Forward Error Correction for Streaming Video in Wireless Networks. IJCSI International Journal of Computer Science Issues, Taiwan, v. 9, n. 5, p. 146-155, set. 2012. Disponível em: <http://www.IJCSI.org>. Acesso em: 01 mar. 2018.

[31] SMPTE. Video. ST 2022-1:2007 - Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks.. [S.l.: s.n.], 2007. 22 p.

APÊNDICE A: RESULTADOS CONCATENADOS:

Parâmetros						PSNR				
FEC	Linha	Coluna	Netem	Original	Destino	Y	U	V	Média	min
N	0	0	5%	Original	5LOSS	13,9424	21,334	27,6813	15,4661	11,38801
S	4	4	5%	Original	44_5LOSS	32,673798	43,104197	43,47772	34,250133	19,3038
S	6	4	5%	Original	64_5LOSS	32,070496	42,764765	43,296982	33,6604	19,3038
S	8	5	5%	Original	85_5LOSS	22,158305	34,600947	37,618639	23,8274	12,9382
S	10	5	5%	Original	105_5LOSS	18,44781	32,219174	34,052842	20,1339	11,4753

TABELA 7: AVALIAÇÃO DOS VÍDEOS TESTADOS PELA MÉTRICA OBJETIVA DE QUALIDADE PSNR

Parâmetros					Perda	Sobrecarga	Delay	Encoder	Decoder	Quantidade de Pacotes Bufferizados
FEC	Linha	Coluna	Perda	Original						
N	0	0	5%	Original	0,00%	0,00%	44	1097	1053	24
N	0	0	5%	5LOSS	5,00%	0,00%	44	1006	962	34
S	4	4	5%	44_5LOSS	5,50%	41,00%	47	596	549	0
S	6	4	5%	64_5LOSS	4,60%	34,30%	48	581	533	0
S	8	5	5%	85_5LOSS	5,80%	24,80%	54	685	631	53
S	10	5	5%	105_5LOSS	4,80%	23,00%	54	869	815	65

TABELA 8 AVALIAÇÃO DA LATÊNCIA, SOBRECARGA DE INFORMAÇÃO E TAMANHO DE BUFFER.

APÊNDICE A.1: CARACTERÍSTICAS DO ARQUIVO DE TESTES:

Media Info:

General

Complete name : SampleVideo_1280x720_5mb.mp4
Format : MPEG-4
Format profile : Base Media
Codec ID : isom (isom/iso2/avc1/mp41)
File size : 5.01 MiB
Duration : 29s 568ms
Overall bit rate mode : Variable
Overall bit rate : 1 422 Kbps
Encoded date : UTC 1970-01-01 00:00:00
Tagged date : UTC 2014-07-19 17:22:11
Writing application : Lavf53.24.2

Video

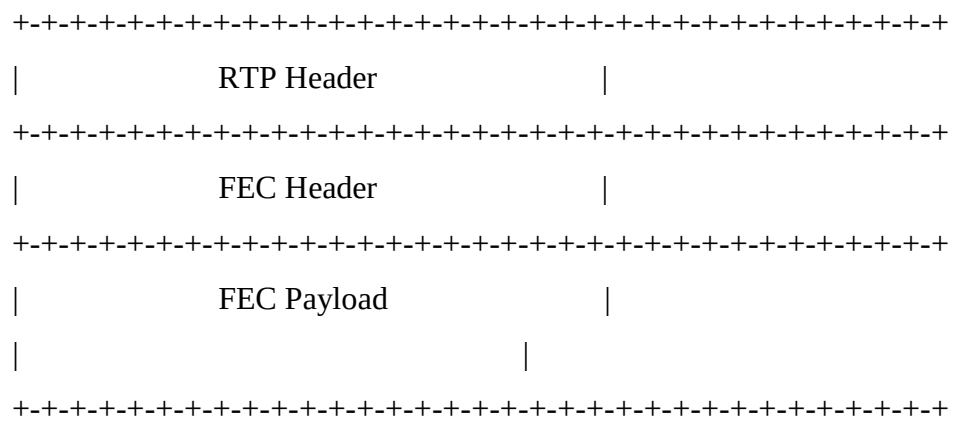
ID : 1
Format : AVC
Format/Info : Advanced Video Codec
Format profile : Main@L3.1
Format settings, CABAC : Yes
Format settings, ReFrames : 1 frame
Codec ID : avc1
Codec ID/Info : Advanced Video Coding
Duration : 29s 560ms
Bit rate : 1 033 Kbps
Width : 1 280 pixels
Height : 720 pixels
Display aspect ratio : 16:9
Frame rate mode : Constant
Frame rate : 25.000 fps
Color space : YUV

Chroma subsampling : 4:2:0
Bit depth : 8 bits
Scan type : Progressive
Bits/(Pixel*Frame) : 0.045
Stream size : 3.64 MiB (73%)
Encoded date : UTC 1970-01-01 00:00:00
Tagged date : UTC 1970-01-01 00:00:00

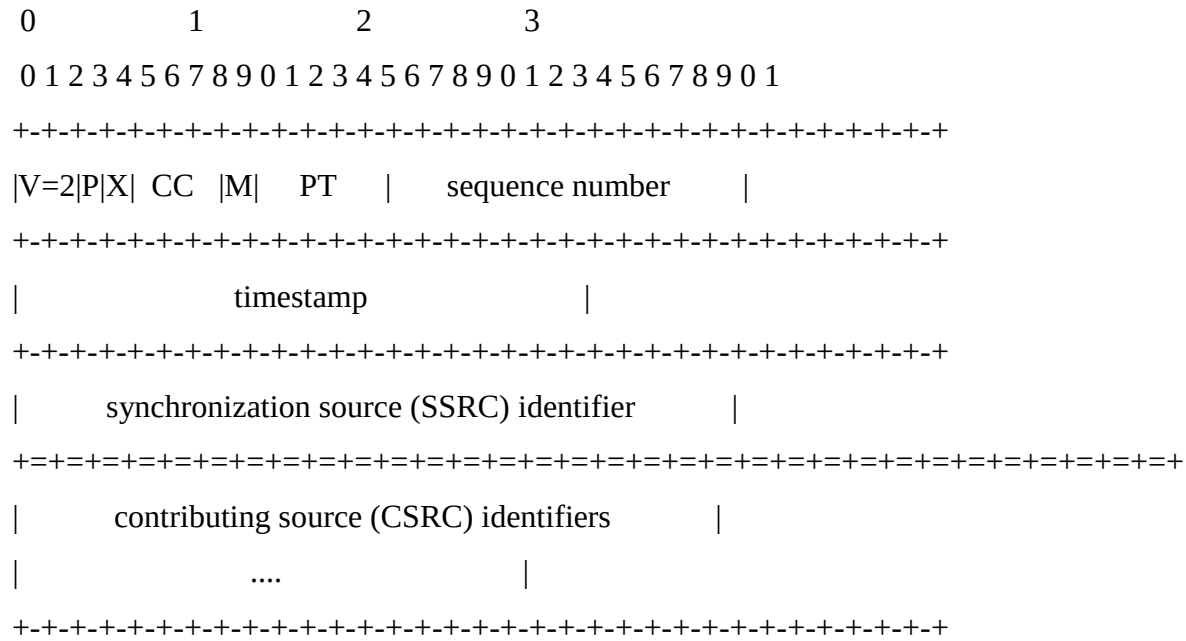
Audio
ID : 2
Format : AAC
Format/Info : Advanced Audio Codec
Format profile : LC
Codec ID : 40
Duration : 29s 568ms
Bit rate mode : Variable
Bit rate : 384 Kbps
Maximum bit rate : 417 Kbps
Channel(s) : 2 channels
Channel(s)_Original : 6 channels
Channel positions : Front: L C R, Side: L R, LFE
Sampling rate : 48.0 KHz
Frame rate : 46.875 fps (1024 spf)
Compression mode : Lossy
Stream size : 1.35 MiB (27%)
Default : Yes
Alternate group : 1
Encoded date : UTC 1970-01-01 00:00:00
Tagged date : UTC 1970-01-01 00:00:00

APÊNDICE A.2: ALGORITMO DE CORREÇÃO DE ERROS PROMPEG

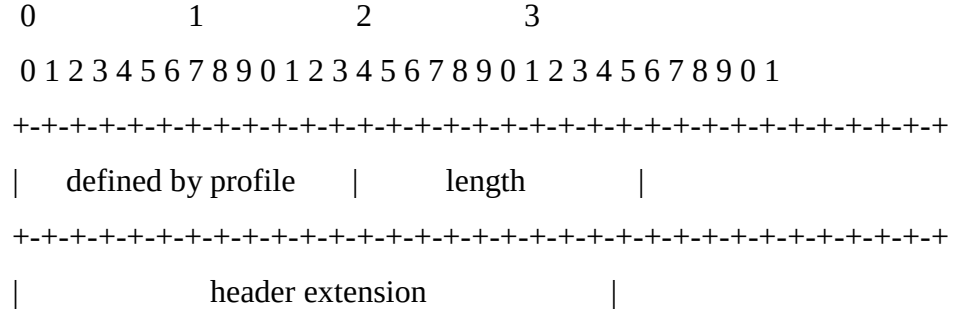
```
/*
 * Pro-MPEG Code of Practice #3 Release 2 FEC
 * Copyright (c) 2016 Mobibase, France (http://www.mobibase.com)
 *
 * This file is part of FFmpeg.
 *
 * FFmpeg is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * FFmpeg is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with FFmpeg; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

/**
 * @file
 * Pro-MPEG Code of Practice #3 Release 2 FEC protocol
 * @author Vlad Tarca <vlad.tarca@gmail.com>
 */

/*
 * Reminder:

 [RFC 2733] FEC Packet Structure
```



[RFC 3550] RTP header



[RFC 3550] RTP header extension (after CSRC)



```
  |                        ....                           |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[Pro-MPEG COP3] FEC Header

```
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |      SNBase low bits          |       length recovery         |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |E| PT recovery |                mask                           |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                         TS recovery                           |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |X|D|type |index|    offset     |      NA       |SNBase ext bits|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

*/

```
#include "libavutil/avstring.h"
#include "libavutil/intreadwrite.h"
#include "libavutil/opt.h"
#include "libavutil/parseutils.h"
#include "libavutil/random_seed.h"
#include "avformat.h"
#include "config.h"
#include "url.h"

#define PROMPEG_RTP_PT 0x60
#define PROMPEG_FEC_COL 0x0
#define PROMPEG_FEC_ROW 0x1

typedef struct PrompegFec {
    uint16_t sn;
    uint32_t ts;
```

```
    uint8_t *bitstring;
} PrompegFec;

typedef struct PrompegContext {
    const AVClass *class;
    URLContext *fec_col_hd, *fec_row_hd;
    PrompegFec **fec_arr, **fec_col_tmp, **fec_col, *fec_row;
    int ttl;
    uint8_t l, d;
    uint8_t *rtp_buf;
    uint16_t rtp_col_sn, rtp_row_sn;
    uint16_t length_recovery;
    int packet_size;
    int packet_idx, packet_idx_max;
    int fec_arr_len;
    int bitstring_size;
    int rtp_buf_size;
    int init;
    int first;
} PrompegContext;

#define OFFSET(x) offsetof(PrompegContext, x)
#define E AV_OPT_FLAG_ENCODING_PARAM

static const AVOption options[] = {
    { "ttl",        "Time to live (in milliseconds, multicast only)", OFFSET(ttl),
AV_OPT_TYPE_INT, { .i64 = -1 }, -1, INT_MAX, .flags = E },
    { "l", "FEC L", OFFSET(l), AV_OPT_TYPE_INT, { .i64 =  5 }, 4, 20, .flags = E },
    { "d", "FEC D", OFFSET(d), AV_OPT_TYPE_INT, { .i64 =  5 }, 4, 20, .flags = E },
    { NULL }
};

static const AVClass prompeg_class = {
```

```
    .class_name = "prompeg",
    .item_name  = av_default_item_name,
    .option     = options,
    .version    = LIBAVUTIL_VERSION_INT,
};

static void xor_fast(const uint8_t *in1, const uint8_t *in2, uint8_t *out, int size) {
    int i, n, s;

#if HAVE_FAST_64BIT
    uint64_t v1, v2;

    n = size / sizeof (uint64_t);
    s = n * sizeof (uint64_t);

    for (i = 0; i < n; i++) {
        v1 = AV_RN64A(in1);
        v2 = AV_RN64A(in2);
        AV_WN64A(out, v1 ^ v2);
        in1 += 8;
        in2 += 8;
        out += 8;
    }
#else
    uint32_t v1, v2;

    n = size / sizeof (uint32_t);
    s = n * sizeof (uint32_t);

    for (i = 0; i < n; i++) {
        v1 = AV_RN32A(in1);
        v2 = AV_RN32A(in2);
        AV_WN32A(out, v1 ^ v2);
```

```
        in1 += 4;
        in2 += 4;
        out += 4;
    }
#endif

    n = size - s;

    for (i = 0; i < n; i++) {
        out[i] = in1[i] ^ in2[i];
    }
}

static int prompeg_create_bitstring(URLContext *h, const uint8_t *buf, int size,
        uint8_t **bitstring) {
    PrompegContext *s = h->priv_data;
    uint8_t *b;

    if (size < 12 || (buf[0] & 0xc0) != 0x80 || (buf[1] & 0x7f) != 0x21) {
        av_log(h, AV_LOG_ERROR, "Unsupported stream format (expected MPEG-TS over
RTP)\n");
        return AVERROR(EINVAL);
    }
    if (size != s->packet_size) {
        av_log(h, AV_LOG_ERROR, "The RTP packet size must be constant (set pkt_size)\n");
        return AVERROR(EINVAL);
    }

    *bitstring = av_malloc(s->bitstring_size);
    if (!*bitstring) {
        av_log(h, AV_LOG_ERROR, "Failed to allocate the bitstring buffer\n");
        return AVERROR(ENOMEM);
    }
```

```
    b = *bitstring;

    // P, X, CC
    b[0] = buf[0] & 0x3f;
    // M, PT
    b[1] = buf[1];
    // Timestamp
    b[2] = buf[4];
    b[3] = buf[5];
    b[4] = buf[6];
    b[5] = buf[7];
    /*
     * length_recovery: the unsigned network-ordered sum of lengths of CSRC,
     * padding, extension and media payload
     */
    AV_WB16(b + 6, s->length_recovery);
    // Payload
    memcpy(b + 8, buf + 12, s->length_recovery);

    return 0;
}

static int prompeg_write_fec(URLContext *h, PrompegFec *fec, uint8_t type) {
    PrompegContext *s = h->priv_data;
    URLContext *hd;
    uint8_t *buf = s->rtp_buf; // zero-filled
    uint8_t *b = fec->bitstring;
    uint16_t sn;
    int ret;

    sn = type == PROMPEG_FEC_COL ? ++s->rtp_col_sn : ++s->rtp_row_sn;

    // V, P, X, CC
```

```
buf[0] = 0x80 | (b[0] & 0x3f);
// M, PT
buf[1] = (b[1] & 0x80) | PROMPEG_RTP_PT;
// SN
AV_WB16(buf + 2, sn);
// TS
AV_WB32(buf + 4, fec->ts);
// CSRC=0
//AV_WB32(buf + 8, 0);
// SNBase low bits
AV_WB16(buf + 12, fec->sn);
// Length recovery
buf[14] = b[6];
buf[15] = b[7];
// E=1, PT recovery
buf[16] = 0x80 | b[1];
// Mask=0
//buf[17] = 0x0;
//buf[18] = 0x0;
//buf[19] = 0x0;
// TS recovery
buf[20] = b[2];
buf[21] = b[3];
buf[22] = b[4];
buf[22] = b[5];
// X=0, D, type=0, index=0
buf[24] = type == PROMPEG_FEC_COL ? 0x0 : 0x40;
// offset
buf[25] = type == PROMPEG_FEC_COL ? s->l : 0x1;
// NA
buf[26] = type == PROMPEG_FEC_COL ? s->d : s->l;
// SNBase ext bits=0
//buf[26] = 0x0;
```

```
    // Payload
    memcpy(buf + 28, b + 8, s->length_recovery);

    hd = type == PROMPEG_FEC_COL ? s->fec_col_hd : s->fec_row_hd;
    ret = ffurl_write(hd, buf, s->rtp_buf_size);
    return ret;
}

static int prompeg_open(URLContext *h, const char *uri, int flags) {
    PrompegContext *s = h->priv_data;
    AVDictionary *udp_opts = NULL;
    int rtp_port;
    char hostname[256];
    char buf[1024];

    s->fec_col_hd = NULL;
    s->fec_row_hd = NULL;

    if (s->l * s->d > 100) {
        av_log(h, AV_LOG_ERROR, "L * D must be <= 100\n");
        return AVERROR(EINVAL);
    }

    av_url_split(NULL, 0, NULL, 0, hostname, sizeof (hostname), &rtp_port,
            NULL, 0, uri);

    if (rtp_port < 1 || rtp_port > UINT16_MAX - 4) {
        av_log(h, AV_LOG_ERROR, "Invalid RTP base port %d\n", rtp_port);
        return AVERROR(EINVAL);
    }

    if (s->ttl > 0) {
        snprintf(buf, sizeof (buf), "%d", s->ttl);
```

```
        av_dict_set(&udp_opts, "ttl", buf, 0);
    }

    ff_url_join(buf, sizeof (buf), "udp", NULL, hostname, rtp_port + 2, NULL);
    if (ffurl_open_whitelist(&s->fec_col_hd, buf, flags, &h->interrupt_callback,
            &udp_opts, h->protocol_whitelist, h->protocol_blacklist, h) < 0)
        goto fail;
    ff_url_join(buf, sizeof (buf), "udp", NULL, hostname, rtp_port + 4, NULL);
    if (ffurl_open_whitelist(&s->fec_row_hd, buf, flags, &h->interrupt_callback,
            &udp_opts, h->protocol_whitelist, h->protocol_blacklist, h) < 0)
        goto fail;

    h->max_packet_size = s->fec_col_hd->max_packet_size;
    s->init = 1;

    av_dict_free(&udp_opts);
    av_log(h, AV_LOG_INFO, "ProMPEG CoP#3-R2 FEC L=%d D=%d\n", s->l, s->d);
    return 0;

fail:
    ffurl_closep(&s->fec_col_hd);
    ffurl_closep(&s->fec_row_hd);
    av_dict_free(&udp_opts);
    return AVERROR(EIO);
}

static int prompeg_init(URLContext *h, const uint8_t *buf, int size) {
    PrompegContext *s = h->priv_data;
    uint32_t seed;
    int i;

    s->fec_arr = NULL;
    s->rtp_buf = NULL;
```

```
if (size < 12 || size > UINT16_MAX + 12) {
    av_log(h, AV_LOG_ERROR, "Invalid RTP packet size\n");
    return AVERROR_INVALIDDATA;
}

s->packet_idx = 0;
s->packet_idx_max = s->l * s->d;
s->packet_size = size;
s->length_recovery = size - 12;
s->rtp_buf_size = 28 + s->length_recovery; // 12 + 16: RTP + FEC headers
s->bitstring_size = 8 + s->length_recovery; // 8: P, X, CC, M, PT, SN, TS
s->fec_arr_len = 1 + 2 * s->l; // row + column tmp + column out

if (h->flags & AVFMT_FLAG_BITEXACT) {
    s->rtp_col_sn = 0;
    s->rtp_row_sn = 0;
} else {
    seed = av_get_random_seed();
    s->rtp_col_sn = seed & 0x0fff;
    s->rtp_row_sn = (seed >> 16) & 0x0fff;
}

s->fec_arr = av_malloc_array(s->fec_arr_len, sizeof (PrompegFec*));
if (!s->fec_arr) {
    goto fail;
}
for (i = 0; i < s->fec_arr_len; i++) {
    s->fec_arr[i] = av_malloc(sizeof (PrompegFec));
    if (!s->fec_arr[i]) {
        goto fail;
    }
    s->fec_arr[i]->bitstring = av_malloc_array(s->bitstring_size, sizeof (uint8_t));
```

```
        if (!s->fec_arr[i]->bitstring) {
            av_freep(&s->fec_arr[i]);
            goto fail;
        }
    }
    s->fec_row = *s->fec_arr;
    s->fec_col = s->fec_arr + 1;
    s->fec_col_tmp = s->fec_arr + 1 + s->l;

    s->rtp_buf = av_malloc_array(s->rtp_buf_size, sizeof (uint8_t));
    if (!s->rtp_buf) {
        goto fail;
    }
    memset(s->rtp_buf, 0, s->rtp_buf_size);

    s->init = 0;
    s->first = 1;

    return 0;

fail:
    av_log(h, AV_LOG_ERROR, "Failed to allocate the FEC buffer\n");
    return AVERROR(ENOMEM);
}

static int prompeg_write(URLContext *h, const uint8_t *buf, int size) {
    PrompegContext *s = h->priv_data;
    PrompegFec *fec_tmp;
    uint8_t *bitstring = NULL;
    int col_idx, col_out_idx, row_idx;
    int ret, written = 0;

    if (s->init && ((ret = prompeg_init(h, buf, size)) < 0))
```

```
    goto end;

if ((ret = prompeg_create_bitstring(h, buf, size, &bitstring)) < 0)
    goto end;

col_idx = s->packet_idx % s->l;
row_idx = s->packet_idx / s->l % s->d;

// FEC' (row) send block-aligned, xor
if (col_idx == 0) {
    if (!s->first || s->packet_idx > 0) {
        if ((ret = prompeg_write_fec(h, s->fec_row, PROMPEG_FEC_ROW)) < 0)
            goto end;
        written += ret;
    }
    memcpy(s->fec_row->bitstring, bitstring, s->bitstring_size);
    s->fec_row->sn = AV_RB16(buf + 2);
    s->fec_row->ts = AV_RB32(buf + 4);
} else {
    xor_fast(s->fec_row->bitstring, bitstring, s->fec_row->bitstring,
            s->bitstring_size);
}

// FEC (column) xor
if (row_idx == 0) {
    if (!s->first) {
        // swap fec_col and fec_col_tmp
        fec_tmp = s->fec_col[col_idx];
        s->fec_col[col_idx] = s->fec_col_tmp[col_idx];
        s->fec_col_tmp[col_idx] = fec_tmp;
    }
    memcpy(s->fec_col_tmp[col_idx]->bitstring, bitstring, s->bitstring_size);
    s->fec_col_tmp[col_idx]->sn = AV_RB16(buf + 2);
```

```
            s->fec_col_tmp[col_idx]->ts = AV_RB32(buf + 4);
        } else {
            xor_fast(s->fec_col_tmp[col_idx]->bitstring, bitstring,
                    s->fec_col_tmp[col_idx]->bitstring, s->bitstring_size);
        }

        // FEC (column) send block-aligned
        if (!s->first && s->packet_idx % s->d == 0) {
            col_out_idx = s->packet_idx / s->d;
            if ((ret = prompeg_write_fec(h, s->fec_col[col_out_idx], PROMPEG_FEC_COL)) < 0)
                goto end;
            written += ret;
        }

        if (++s->packet_idx >= s->packet_idx_max) {
            s->packet_idx = 0;
            if (s->first)
                s->first = 0;
        }

        ret = written;

    end:
        av_free(bitstring);
        return ret;
    }

    static int prompeg_close(URLContext *h) {
        PrompegContext *s = h->priv_data;
        int i;

        ffurl_closep(&s->fec_col_hd);
        ffurl_closep(&s->fec_row_hd);
```

```
    if (s->fec_arr) {
        for (i = 0; i < s->fec_arr_len; i++) {
            av_free(s->fec_arr[i]->bitstring);
            av_freep(&s->fec_arr[i]);
        }
        av_freep(&s->fec_arr);
    }
    av_freep(&s->rtp_buf);

    return 0;
}

const URLProtocol ff_prompeg_protocol = {
    .name                = "prompeg",
    .url_open            = prompeg_open,
    .url_write           = prompeg_write,
    .url_close           = prompeg_close,
    .priv_data_size      = sizeof(PrompegContext),
    .flags               = URL_PROTOCOL_FLAG_NETWORK,
    .priv_data_class     = &prompeg_class,
};
```

APÊNDICE A.3: ALGORITMO DE DECODIFIÇÃO DE ERROS PROMPEG

```
#include <iostream>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <pthread.h>
#include "socketConnection.h"
#include "packetQueue.h"
```

```
#include "packetBuffer.h"
#include "monitor.h"
#define RECVBUFLEN 1500

using namespace std;

void *threadproc(void *arg);
packetBuffer *myPacketBuffer = new packetBuffer(2048);
monitor *myMonitor = new monitor();

int main(int argc, char *argv[]) {
   string mediaIP;
   string mediaPort;
   string fecTimes;
   string maxDelay;
   unsigned char *sockRecvBuf = (unsigned char*)malloc(RECVBUFLEN * sizeof(unsigned
char));

   if(argc == 2) {
      mediaIP = "239.0.0.1";
      mediaPort = "20000";
      maxDelay = argv[1];
   }
   else if(argc == 4) {
      mediaIP = argv[1];
      mediaPort = argv[2];
      maxDelay = argv[3];
   }
   else {
      mediaIP = "239.0.0.1";
      mediaPort = "20000";
      maxDelay = "500";
   }
```

```
    socketUtility *mySocketUtility = new socketUtility(mediaIP.c_str() , mediaPort.c_str());

    fd_set master;
    fd_set read_fds;
    FD_ZERO(&master);
    FD_ZERO(&read_fds);
    FD_SET(mySocketUtility -> media_Sockfd , &master);
    FD_SET(mySocketUtility -> fecRow_Sockfd , &master);
    FD_SET(mySocketUtility -> fecCol_Sockfd , &master);

    read_fds = master;

    pthread_t tid;
    pthread_create(&tid , NULL , &threadproc , NULL);

    for(;;) {
        myPacketBuffer -> updateFecQueue();
        myPacketBuffer -> fecRecovery();
        myPacketBuffer -> updateMediaQueue( mySocketUtility -> output_Sockfd ,
stoi(maxDelay) * 100 );
        myPacketBuffer -> updateMinSN();

        myMonitor -> updateRecovered(myPacketBuffer -> recovered);

        read_fds = master;
        struct timeval tv = {0 , 50};
        select( (mySocketUtility -> fdmax)+1 , &read_fds , NULL , NULL , &tv);

        if(FD_ISSET(mySocketUtility -> media_Sockfd , &read_fds)) {
            int bytes = 0;
            if((bytes = recv(mySocketUtility -> media_Sockfd , sockRecvBuf , RECVBUFLEN ,
0)) < 0)
```

```
            mySocketUtility -> ~socketUtility();

         myMonitor -> media -> updateStatus(sockRecvBuf);
         myPacketBuffer -> newMediaPacket(sockRecvBuf , bytes);
      }

      if(FD_ISSET(mySocketUtility -> fecRow_Sockfd , &read_fds)) {
         int bytes = 0;
         if((bytes = recv(mySocketUtility -> fecRow_Sockfd , sockRecvBuf , RECVBUFLEN ,
0)) < 0)
            mySocketUtility -> ~socketUtility();

         myMonitor -> fecRow -> updateStatus(sockRecvBuf);
         myPacketBuffer -> newFecPacket(sockRecvBuf , bytes);
      }

      if(FD_ISSET(mySocketUtility -> fecCol_Sockfd , &read_fds)) {
         int bytes = 0;
         if((bytes = recv(mySocketUtility -> fecCol_Sockfd , sockRecvBuf , RECVBUFLEN ,
0)) < 0)
            mySocketUtility -> ~socketUtility();

         myMonitor -> fecCol -> updateStatus(sockRecvBuf);
         myPacketBuffer -> newFecPacket(sockRecvBuf , bytes);
      }
   }

   return 0; //never reached
}

void *threadproc(void *arg) {
   while(1) {
      sleep(5);
```

```
        myPacketBuffer -> bufferMonitor();
        myMonitor -> printMonitor();
    }
    return 0;
}
```

APÊNDICE A.4: SCRIPT DE TESTE SERVIDOR

 # Este Código garante uma transferência ponto a ponto sem perda no canal e sem correção de erros.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os
import sys
os.system('sudo ifconfig eth0 10.10.9.2 netmask 255.255.255.0 broadcast 10.10.9.255')

## DEFINIR VIDEO DO TESTE:
#Video_In =  "SampleVideo_2mb.mp4"
Video_In =  "SampleVideo_1280x720_5mb.mp4"
#Video_In =  "SampleVideo_30mb"
Path = "/home/carol/Downloads/Samples_Videos/" + Video_In
code = "ffmpeg -re -i " + Path + " -intra -s 800x600 -pix_fmt yuv444p -vcodec copy -q 5 -b:a 50k -maxrate 100k -bufsize 500 -preset slow -benchmark -psnr -copyinkf -f rtp_mpegts -fec prompeg=l=6:d=4 rtp://10.10.9.2:8089 “
os.system(code)
```

 # Este código garante uma transferência ponto a ponto com 1% de perda no canal e sem correção de erros.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

```
import os
import sys
os.system('sudo ifconfig eth0 10.10.9.2 netmask 255.255.255.0 broadcast 10.10.9.255')
os.system("sudo tc qdisc del dev eth0 root netem loss 1%")
## DEFINIR VIDEO DO TESTE:
#Video_In =  "SampleVideo_2mb.mp4"
Video_In =  "SampleVideo_1280x720_5mb.mp4"
#Video_In =  "SampleVideo_30mb"
os.system("sudo tc qdisc add dev eth0 root netem loss 1%")
#os.system("sudo tc qdisc change dev eth0 root netem loss 1%")

Path = "/home/carol/Downloads/Samples_Videos/" + Video_In

code = " ffmpeg -re -i " + Path + " -intra -s 800x600 -pix_fmt yuv444p -vcodec copy -q 5 -b:a
50k -maxrate 100k -bufsize 500 -preset slow -benchmark -psnr -copyinkf -f rtp_mpegts -fec
prompeg=l=6:d=4 rtp://10.10.9.2:8089 "

os.system(code)
```

 # Este código garante uma transferência ponto a ponto com 1% de perda no canal e com correção de erros definida por L e D.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os
import sys
#Define variaveis
L = 6
D= 4
os.system('sudo ifconfig eth0 10.10.9.3 netmask 255.255.255.0 broadcast 10.10.9.255')
os.system("sudo tc qdisc del dev eth0 root netem loss 1%")
## DEFINIR VIDEO DO TESTE:
#Video_In =  "SampleVideo_2mb.mp4"
```

```
Video_In =  "SampleVideo_1280x720_5mb.mp4"
#Video_In =  "SampleVideo_30mb"
os.system("sudo tc qdisc add dev eth0 root netem loss 1%")
#os.system("sudo tc qdisc change dev eth0 root netem loss 1%")

Path = "/home/carol/Downloads/Samples_Videos/" + Video_In

code = " ffmpeg -re -i " + Path + " -intra -s 800x600 -pix_fmt yuv444p -vcodec copy -q 5 -b:a 50k -maxrate 100k -bufsize 500 -preset slow -benchmark -psnr -copyinkf -f rtp_mpegts -fec prompeg=l=6:d=4 rtp://10.10.9.2:8089 "

os.system(code)
```

APÊNDICE A.5: SCRIPT DE TESTE CLIENTE

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import os
import sys
#### Define as variaveis do teste ####
L = 6
D = 4
os.system('sudo ifconfig eth0 10.10.9.3 netmask 255.255.255.0 broadcast 10.10.9.255')

receptor1 = "ffmpeg -i rtp://10.10.9.3:8089  -c copy ORIGINALFILE.mov"
receptor2 = "ffmpeg -i rtp://10.10.9.3:8089 -c copy FILE_1_LOSS.mov"
receptor3 = "ffmpeg -i rtp://10.10.9.3:8089 -f rtp_mpegts -fec prompeg=l=6:d=4 -c copy 6_4_FEC_FILE_1_LOSS .mov"
```

APÊNDICE A.6: RESULTADO DO DECODIFICADOR

Correção de erros L = 4 e D = 4

media stream:
recvd: 6195 , lost: 358 , loss rate: 5.463147
recovered/lost: 356/358 = 99.441341
loss rate after recovery: 2/6553 = 0.030520
fecRow stream:
recvd: 1550 , lost: 88 , loss rate: 5.372405
fecCol stream:
recvd: 1555 , lost: 80 , loss rate: 4.892966

Correção de erros L = 6 e D = 4
media stream:
recvd: 6211 , lost: 343 , loss rate: 5.233445
recovered/lost: 343/343 = 100.000000
loss rate after recovery: 0/6554 = 0.000000
fecRow stream:
recvd: 1029 , lost: 63 , loss rate: 5.769231
fecCol stream:
recvd: 1546 , lost: 87 , loss rate: 5.327618
Correção de erros L = 8 e D = 5
media stream:
recvd: 6211 , lost: 343 , loss rate: 5.233445
recovered/lost: 338/343 = 98.542274
loss rate after recovery: 5/6554 = 0.076289
fecRow stream:
recvd: 781 , lost: 37 , loss rate: 4.523227
fecCol stream:
recvd: 1234 , lost: 69 , loss rate: 5.295472

Correção de erros L = 10 e D = 5
media stream:
recvd: 6234 , lost: 320 , loss rate: 4.882514
recovered/lost: 311/320 = 97.187500
loss rate after recovery: 9/6554 = 0.137321

fecRow stream:

recvd: 617 , lost: 38 , loss rate: 5.801527

fecCol stream:

recvd: 1242 , lost: 59 , loss rate: 4.534973