Universidade de São Paulo

Escola de Engenharia de São Carlos

Departamento de Engenharia Elétrica

Trabalho de Conclusão de Curso

Protocolos de Comunicação para Automação de Sistemas de Energia – Análise Teórica e Aplicação Prática

Autor:

Vinicius Orlando Grigoletto

Número USP: 6447531

Orientador:

Prof. Dr. Dennis Brandão

São Carlos, Novembro de 2012.

Orlando Grigoletto, Vinicius
Protocolos de Comunicação para Automação de Sistemas de Energia - Análise Teórica e Aplicação Prática / Vinicius Orlando Grigoletto; orientador Dennis Brandão. São Carlos, 2012.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2012. 1. Modbus. 2. DNP3. 3.IEC61850. 4. Protocolos.

Vinicius Orlando Grigoletto

Protocolos de Comunicação para Automação de Sistemas de Energia – Análise Teórica e Aplicação Prática

Trabalho de Conclusão de Curso apresentado à
Escola de Engenharia de São Carlos,
da Universidade de São Paulo

Curso de Engenharia Elétrica com
ênfase em Eletrônica

ORIENTADOR: Prof. Dr. Dennis Brandão

São Carlos

2012

FOLHA DE APROVAÇÃO

Nome: Vinicius Orlando Grigoletto

Título: "Protocolos de Comunicação para Automação de Sistemas de Energia - Análise Teórica e Aplicação Prática"

Trabalho de Conclusão de Curso defendido e aprovado em 26 / 11 / 2012,

com NOTA 10,0 (dez, zero), *pela Comissão Julgadora:*

Prof. Dr. Dennis Brandão (Orientador)
SEL/EESC/USP

Prof. Associado Ivan Nunes da Silva
SEL/EESC/USP

Prof. Assistente Edson Gesualdo
SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

Dedicatória

À minha família, agradeço por todo suporte,
atenção e paciência durante todo este trajeto.
A pessoa que sou hoje é apenas o resultado
de todos estes anos de dedicação e carinho.
Meus mais sinceros agradecimentos.

Agradecimentos

Este trabalho começou a ser idealizado após a disciplina *SEL0378 – Redes de Computadores* – ministrada pelo Prof. Dr. Ivan Nunes da Silva, ter sido cursada durante o penúltimo ano da graduação. Após visão geral dos aspectos práticos e teóricos de protocolos genéricos ser apresentada na disciplina, a sugestão de uma comparação teórica e prática entre exemplos utilizados no mercado foi bem aceita pelo orientador desta monografia, motivando o trabalho aqui desenvolvido. Por este motivo, seguem meus mais sinceros agradecimentos:

- ao professor Dr. Dennis Brandão, pelas orientações, dicas e conselhos ao me guiar neste trabalho;

- ao colega Eng. Guilherme Sestito, por toda atenção, paciência e auxílio prestados durante o desenvolvimento desta monografia;

- ao Eng. Paulo Roberto A. de Souza, por toda a ajuda, atenção e tempo cedidos que contribuíram fundamentalmente na escrita deste trabalho;

- aos engenheiros e amigos de trabalho Leandro Pereira, Guilherme Lourenço e Thiago Pedro, por todo o aprendizado passado dia-a-dia, desde o primeiro dia de estágio;

- aos meus gestores Rafael Ozaki e Mauro Ablas, agradeço por toda a atenção, preocupação, pela experiência passada e por sempre disponibilizarem tempo para orientar e discutir sobre qualquer assunto, apesar do ritmo caótico do dia-a-dia;

- à minha família: Eva, Olinda, Lúcio, Orlando e Layza: obrigado por todo suporte, carinho e por enfrentarem comigo todas as barreiras encontradas durante os últimos anos;

- a todos os meus amigos, por tornarem todos estes últimos anos os mais cansativos, mas também os mais divertidos. Foram marcantes, sem dúvida nenhuma.

Muito obrigado a todos!

Sumário

Índice de Figuras

Índice de Tabelas

Definições e Abreviações

- IED: *Intelligent Electronic Device*
- DNP: *Distributed Network Protocol*
- CSMA-CD: *Carrier Sense Multiple Access with Collision Detection*
- GOOSE: *Generic Object Oriented Substation Events*
- OSI - *Open Systems Interconnection*
- IEC - *International Electrotechnical Commission*
- IP – *Internet Protocol*
- TCP – *Transmission Control Protocol*
- MAC – *Media Acess Control*
- *GSE - Generic Substation Events*
- IHM – Interface Homem – Máquina
- SCADA – *Supervisory Control and Data Acquisition*

Resumo

GRIGOLETTO V. O. *Protocolos de Comunicação para Automação de Sistemas de Energia – Análise Teórica e Aplicação Prática.* 2012. Trabalho de Conclusão de Curso – Escola de Engenharia de São Carlos, Universidade de São Paulo.

O trabalho aqui desenvolvido propõe uma abordagem teórica e prática de três protocolos de comunicação bastante utilizados para automação de sistemas de energia, Modbus, DNP3 e IEC61850. Inicia-se a monografia com uma pequena revisão teórica de conceitos básicos de redes de comunicação, buscando clarear conceitos básicos de organização lógica de protocolos, topologias físicas e equipamentos comuns do ramo. Realizada esta revisão, parte-se para uma análise detalhada das características lógicas de cada protocolo, buscando mostrar ao leitor como os *frames* de cada um são formados, como se dá a comunicação entre dispositivos que se conectam através do padrão em estudo e quais são os pontos mais importantes a serem observados na escolha da utilização dentre estes citados. Para tornar a abordagem teórica mais próxima da realidade de mercado, realizaram-se testes de bancada para relacionar aspectos abordados na teoria ao que, de fato, são fatores importantes em suas utilizações práticas.

Palavras-chaves: Modbus, IEC61850, DNP3, *Ethernet*, Protocolo.

Abstract

GRIGOLETTO V. O. *Communication Protocols for Energy Systems Automation – Theoretical Analysis and Practical Application.* 2012. Course Conclusion Work – Engineering School of São Carlos, University of São Paulo.

The monograph here developed proposes a theoretical and practical analysis of three main protocols usually found on automation of energy systems: Modbus, DNP3 and IEC61850. This monograph starts with a brief review of basic concepts of communication networks, focusing on clarify the principles of protocols logic organization, physics topology and most frequently found equipment. After this review, it follows to a detailed analysis of the logical characteristics of each protocol, aiming to show the reader how the message *frames* are built, how the communication between devices using the same standard works and which are the most important characteristics that must be studied when choosing one of the protocols in study. To make the theoretical approach closer to the practical use in the field, some workbench tests were done to relate aspects seen in theory that are, in fact, important for practical use.

Key-words: Modbus, IEC61850, DNP3, *Ethernet*, Protocol.

Capítulo 1 – Introdução

A crescente evolução de funcionalidades dos dispositivos eletrônicos inteligentes integrantes de sistemas de energia, conhecidos como IED's – *Intelligent Electronic Devices* – teve como consequência o aumento da complexidade lógica e de processamento de cada aparelho, exigindo cada vez maior desempenho das redes de comunicação que interligam estes dispositivos. A necessidade de uma comunicação rápida e confiável entre estas entidades eletrônicas, a grande variedade de possíveis soluções para uma mesma situação e o dilema de encontrar um ponto de equilíbrio entre desempenho e custo de um sistema tornam o planejamento e manutenção destas redes assuntos bastante delicados no setor de energia.

A criação de protocolos de comunicação surgiu da necessidade de se padronizar a maneira com que entidades se comunicam; a simples utilização de IED's com alto desempenho individual não garante um bom desempenho de um sistema se a maneira com que estes dispositivos trocam informações for muito simples (e com baixa capacidade de transmissão de dados) ou muito complexa (aumentando demasiadamente a probabilidade de eventuais erros).

Além da estrutura lógica do protocolo, outro ponto bastante importante na automação de subestações de energia é a interoperabilidade entre dispositivos de diferentes marcas: seria bastante importante criar protocolos que fossem mundialmente aceitos e que a capacidade de transferência de dados fosse independente dos fabricantes dos dispositivos participantes da rede, simplificando a instalação de sistemas de energia e aumentando as possíveis combinações de aparelhos de diferentes marcas trabalhando em conjunto, visando maximizar o desempenho do sistema e uma concorrência justa no mercado.

O intuito deste trabalho de conclusão de curso é apresentar ao leitor uma abordagem teórica e uma amostra prática de utilização de três protocolos de comunicação que buscaram satisfazer estes quesitos e que são amplamente utilizados no mercado de sistemas de energia mundial: **Modbus**, **DNP3** e **IEC61850**. Tais protocolos passam por diferentes momentos no mercado, possuem diferentes estruturas físicas, de dados e de focos de aplicação; é uma tarefa bastante árdua decidir qual protocolo melhor atende determinada solução devido à grande quantidade de características que devem ser levadas em conta.

Iniciando este trabalho com uma breve revisão de conceitos básicos de topologia e estruturas de dados de redes, o leitor poderá conhecer no detalhe aspectos lógicos de cada um dos protocolos, a forma de organização dos *frames* de dados e os prós e contras de suas utilizações. Para tornar este estudo mais próximo da realidade, apresentam-se, ao final do trabalho, exemplos práticos da utilização destes protocolos em bancada, resgatando as características abordadas na teoria.

Espera-se que com o estudo desta monografia o leitor consiga refletir com maior clareza qual protocolo atende, de forma mais plausível, a suas necessidades práticas, além de entender quais as tendências que impulsionam as atualizações desta área de

comunicação de dispositivos de sistemas de energia. O estudo aqui abordado será apresentado da seguinte forma:

- Capítulo 2: é apresentada uma breve revisão de topologias físicas de redes, de elementos básicos encontrados em sistemas e uma breve descrição do modelo OSI, permitindo ao leitor se adaptar e entender noções básicas que serão recuperadas nos capítulos subsequentes.

- Capítulo 3: apresentação teórica do protocolo Modbus, contendo informações acerca de sua estrutura lógica, fluxo de comunicação entre dispositivos que compartilham o protocolo, breve história de sua utilização e características das áreas de aplicação.

- Capítulo 4: com estrutura semelhante ao capítulo anterior, apresentam-se ao leitor as características lógicas e funcionais do protocolo DNP3, realizando também uma abordagem histórica de sua utilização para situar os dados apresentados dentro do contexto prático.

- Capítulo 5: análise teórica do protocolo IEC61850, abordando as características do mercado que impulsionaram seu desenvolvimento, características lógicas, estruturais e funcionais e também situações práticas de sua utilização.

- Capítulo 6: testes em bancada da utilização dos protocolos anteriormente apresentados, buscando realizar a conexão entre toda a abordagem teórica realizada anteriormente com a utilização, na prática, dos padrões de comunicação, focando especialmente as características mais destacadas no mercado.

- Capítulo 7: considerações finais.

Capítulo 2 – Revisão de Conceitos de Redes

Este capítulo tem como objetivo esclarecer alguns conceitos básicos de redes de comunicação de dispositivos eletrônicos, visando facilitar o entendimento dos próximos capítulos em que serão detalhadas características básicas de cada protocolo.

Entre os pontos que serão aqui abordados, pode-se citar definições de termos como protocolo, topologias físicas e lógicas, recomendações de arquiteturas de acordo com a aplicação, formas de representação de um protocolo, etc.

2.1 – Topologias Físicas

A primeira questão que deve ser levantada ao se iniciar o projeto de uma rede de comunicação é qual será a disposição física que será implementada entre os dispositivos. Para respondê-la, deve-se levar em consideração fatores como o espaço disponível, a velocidade que irá satisfazer com qualidade a aplicação, a necessidade (ou não) de redundância.

As principais topologias físicas utilizadas em sistemas de energia são as do tipo **estrela** e **anel.** Daremos ênfase a estes dois tipos, detalhando como os dispositivos são dispostos na rede e quais as vantagens e desvantagens de se optar por uma ou outra configuração.

2.1.1 - Topologia Estrela

A topologia estrela é caracterizada pela presença de um ponto central de distribuição em que todos os dispositivos pertencentes à rede se conectam, normalmente um dispositivo *switch.* Dessa forma, não existe conexão direta entre dispositivos, que podem ser PC's, relés digitais de proteção, roteadores, etc.

Esta configuração possibilita uma alta taxa de transmissão de dados, fácil entendimento da arquitetura e facilidade de expansão (dependendo apenas da capacidade do dispositivo central). É normalmente utilizada em redes pequenas, mas existe a possibilidade de utilizar mais de um *switch* caso o número de portas de um único dispositivo seja insuficiente, através de cascateameanto de *switches* sem a utilização de cabo *crossover* (cabo de par trançado que permite a ligação de dois dispositivos pelas respectivas placas de rede sem a necessidade de utilização de um dispositivo *switch)*. Por outro lado, o cabeamento desta arquitetura é mais custoso, e um defeito no dispositivo central compromete todo o sistema. Por outro lado, um problema em um dispositivo isolado não compromete o restante da rede. A figura 1 mostra a disposição básica desta topologia.

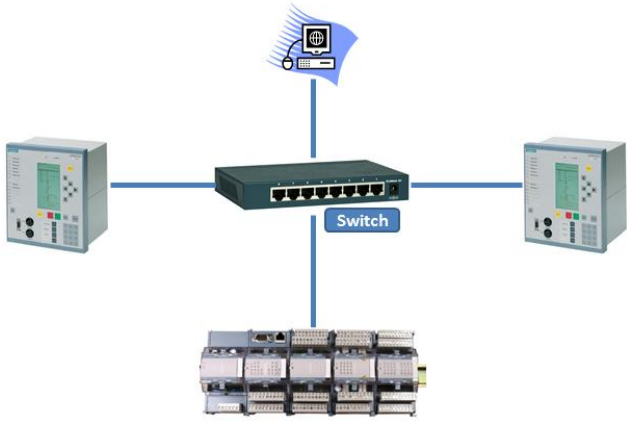

Figura 1 - Disposição de equipamentos na topologia estrela - fonte: [12]

Switches são elementos chaveadores: eles possibilitam simular um segmento da rede para cada dispositivo a ele conectado, chaveando em alta velocidade e com a possibilidade de organizar prioridade de atendimento das portas, fator muito importante em sistemas de energia, em que elementos de proteção devem possuir preferência de tráfego em relação a outros dispositivos. Tais elementos possuem preços muito acessíveis no mercado, e sua eficiência e rapidez no tráfego dos dados garante sua utilização na maioria dos sistemas de comunicação.

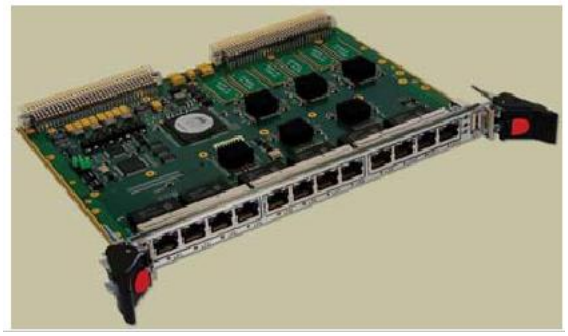

Figura 2 - Exemplo de um dispositivo switch - fonte: [1]

Pelo fato destes dispositivos estarem cada vez mais acessíveis, sempre se opta por não utilizar estruturas muito básicas que não garantem nenhum tipo de redundância de envio em caso de falhas. A figura 1 mostra uma disposição muito básica dos equipamentos, e, como citado, a falha no *switch* central impediria toda a comunicação da rede. Um exemplo de aperfeiçoamento da rede contra falhas está exemplificado na figura 5: apenas pela adição de um *switch* extra, cada dispositivo integrante da rede possui dois caminhos possíveis para se comunicar com os outros dispositivos (desde que cada dispositivo possua duas portas disponíveis trabalhando em paralelo, fato bastante comum nos dispositivos atuais – exemplo mostrado na figura 3). Em caso de falha de um dos *switches,* o módulo de

comunicação altera a porta de comunicação para a que está conectada ao *switch* “reserva”, sem interromper o tráfego de dados.

Figura 3 - Módulo de comunicação Ethernet com 2 portas - redundância - fonte: [11]

2.1.2 - Topologia Anel

A topologia anel é caracterizada, como o próprio nome diz, pela disposição dos componentes em um círculo fechado – cada dispositivo é conectado a outros dois, conforme mostrado na Figura 4. A informação é transmitida de dispositivo em dispositivo até atingir o alvo. Pode-se imaginar que, caso exista um problema em um dos dispositivos pertencentes ao anel, quebra-se o círculo e a comunicação é interrompida, mas tal fato não é mais uma preocupação, pois a rede consegue se rearranjar para uma configuração em linha caso um dos dispositivos perca comunicação, comportando-se como um barramento.

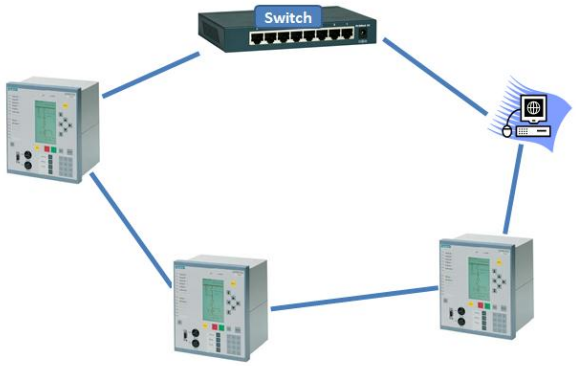

Figura 4 - Disposição de equipamentos na topologia anel - fonte: [12]

Novamente, procura-se melhorar as alternativas de rearranjo da rede em caso de falhas: a utilização de *switches* trabalhando em paralelo “fechando” um anel de dispositivos de segurança é uma boa opção no quesito custo x desempenho, conforme mostra a figura 5. A adição de um *switch* adicional possibilita que o sistema continue a transmitir dados em caso de um dos *switches* apresentarem problemas, não interrompendo a comunicação com o sistema supervisório (representado por PC na figura 5).

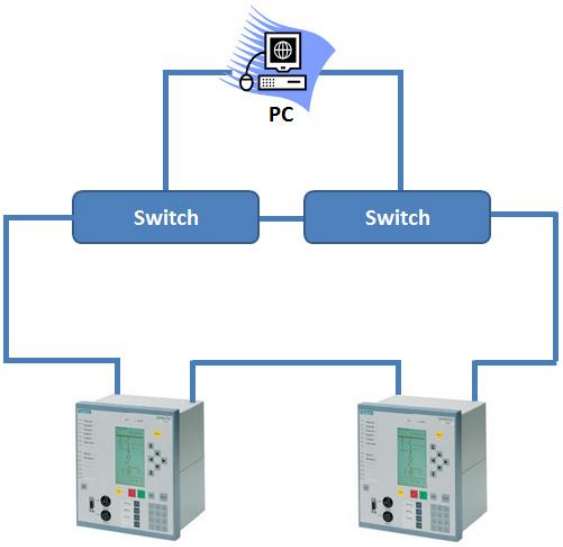

Figura 5 - Topologia anel com switches - fonte: [12]

2.2 – Métodos de acesso ao meio físico

Existem três métodos mais utilizados de acesso ao meio físico de transmissão de dados utilizados por dispositivos pertencentes a uma rede, que dependem da topologia física instalada: acesso mestre-escravo, acesso via *token* e acesso CSMA.

Na primeira situação, que pode ser implementada em qualquer topologia física, os dispositivos “mestre” (*master*) possuem acesso liberado à rede de transmissão; os dispositivos “escravos” (*slaves)* são questionados, em intervalos de tempo pré-definidos, pelos mestres e enviam mensagens de resposta quando aplicável. Não existe comunicação direta entre dispositivos escravos, portanto é um método de acesso bastante centralizado e limitado. Um exemplo deste tipo de acesso pode ser visto na segunda imagem da Figura 6.

Já o acesso via *token,* ilustrado na primeira imagem da Figura 6, mais comum em redes de topologia anel, possui uma sequência de *bits* (o *token)* controlada pelo *hardware* do meio de comunicação que trafega em alta velocidade entre os dispositivos: quando algum integrante da rede precisa enviar uma mensagem, ele retira o *token* da rede, garantindo acesso exclusivo a ela momentaneamente; feito isso, transmite sua mensagem para o dispositivo alvo, que quando a recebe, copia para si e devolve a mensagem na rede; por fim, a mensagem retorna ao dispositivo remetente, que entende que sua mensagem foi transmitida, devolvendo o *token* à rede e liberando acesso ao meio para os outros dispositivos.

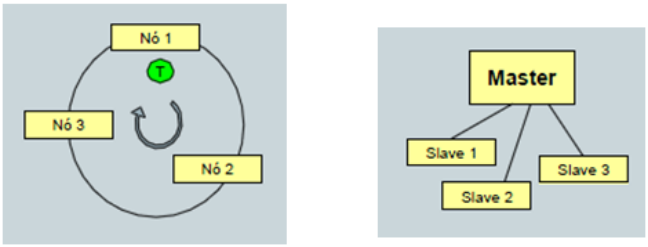

Figura 6 - Métodos de acesso ao meio físico - token e mestre/escravo – fonte: [10]

Por último, o acesso via tecnologia *Ethernet* com mecanismo CSMA – *Carrier Sense Multiple Acess* – necessita da utilização de topologia barramento (utilização mais comum do cabo *Ethernet*). Quando um dispositivo conectado ao barramento deseja transmitir sua mensagem, ele ganha acesso exclusivo ao cabo e transmite sua informação em ambas as direções do barramento, obrigando todos os outros dispositivos a aguardar a finalização de sua transmissão.

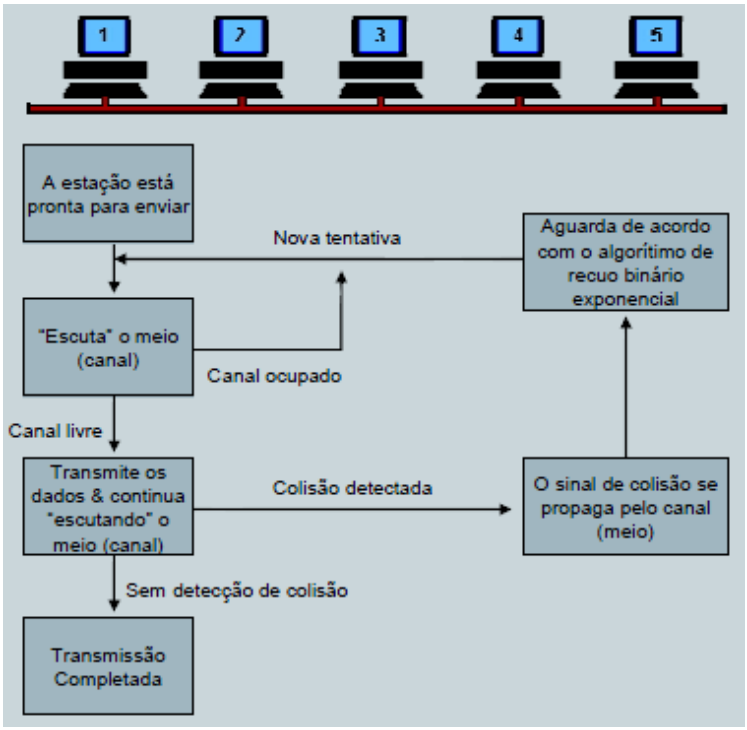

Figura 7 - Mecanismo CSMA-CD – fluxograma – fonte: [10]

O mecanismo CSMA-CD consiste do monitoramento constante da atividade elétrica presente no barramento (*carrier sense).* Quando não há presença de portadora, é sinal de que nenhum dispositivo está transmitindo, situação em que o dispositivo prossegue para o envio de sua mensagem. Entretanto, podem ocorrer casos de dois dispositivos não detectarem a portadora, entenderem que estão liberados para transmitir e tentarem acessar o barramento simultaneamente, gerando uma tensão anormal no cabo: nestas situações, ambos os dispositivos recuam e fazem nova tentativa de acesso ao meio de transmissão em

um intervalo aleatório entre 0 e T segundos. Se ocorrer novamente o caso de ambos os dispositivos escolherem o mesmo momento para retransmitir a mensagem, ambos os dispositivos escolhem novo momento de retransmissão entre 0 e 2T segundos, e assim por diante. Este método de expansão de intervalos de retransmissão é denominado *Binary Exponential Backoff,* e garante que os dispositivos conseguirão acessar o meio de transmissão em um número reduzido de tentativas. Um fluxograma representando estas ações de tentativas de transmissão pode ser visto na Figura 7 acima.

2.3 - Protocolos e Modelo de Dados

Para que seja possível a troca de informação entre dois dispositivos, ambos precisam seguir as mesmas regras de comunicação para que entendam o papel de cada conjunto de *bits* dentro da mensagem. Denomina-se protocolo este conjunto de regras que governa a comunicação entre dois dispositivos, alinhando características de sintaxe (formato dos dados e codificação), semântica (informações de controle para coordenação de erros) e sincronização (definição da velocidade de envio das mensagens). Sem este conjunto de regras estabelecendo as diretrizes de comunicação entre os dispositivos, pode-se comparar tal situação a dois dispositivos falando idiomas diferentes, impossibilitando o recebimento/envio da mensagem.

Em suma, os protocolos de comunicação determinam todas as ações desde a geração de uma mensagem até seu recebimento no destino correto. Ações de segmentação e reagrupamento de mensagens muito grandes, encapsulamento, controle da conexão e determinação de prioridades são características que também devem ser especificadas pelo protocolo.

A criação de protocolos só foi possível a partir da determinação de um modelo de arquitetura que possibilitasse a comunicação entre máquinas heterogêneas, denominado modelo OSI – *Open Systems Interconnection* - Interconexão de Sistemas Abertos, lançado em 1984. Este modelo divide as redes de computadores em sete camadas, cabendo a cada protocolo atribuir regras particulares a cada uma destas camadas, possibilitando que diferentes tecnologias sejam utilizadas em conjunto em um mesmo sistema. A figura 8 mostra as camadas que compõem o modelo OSI.

Figura 8 - Estruturação das camadas no modelo OSI - fonte: [1]

O princípio de organização do modelo OSI é a definição exata das funções de cada camada, possibilitando a criação de protocolos direcionados a cada uma delas; é amplamente aceito no mundo, e se aplica à maioria das redes existentes. Vale ressaltar que as redes não precisam seguir fielmente o modelo OSI: é bastante comum omitir algumas camadas buscando melhor desempenho ou simplicidade da lógica da rede, mas toda rede pode ser mapeada no modelo.

Como visto na última figura, toda a comunicação entre dispositivos é dividia em sete camadas, que são separadas em dois grupos; as camadas superiores fazem requisições às camadas inferiores, que devem fornecer os dados requisitados; são numeradas em ordem crescente de baixo para cima: da camada um até a camada quatro têm-se as camadas do **conjunto de transporte**; da camada cinco até a camada sete têm-se as camadas **do grupo de aplicação**. Para que exista a comunicação entre dispositivos, é estritamente necessário que ambos se comuniquem baseando-se no mesmo protocolo (camadas equivalentes de dispositivos diferentes devem utilizar o mesmo protocolo de comunicação).

A denominação genérica de uma camada do modelo OSI é "camada K". Camadas de um mesmo nível de dispositivos diferentes comunicam-se entre si através de protocolos de tipo K transmitindo dados denominados unidade de dados de protocolo – *Protocol Data Unit,* ou simplesmente PDU. Por outro lado, os dados que trafegam de uma camada (K) para uma camada inferior (K-1) são denominados unidades de dados de serviço – *Service Data Unit,* ou simplesmente SDU. Dados deste tipo ainda não foram encapsulados pela camada inferior: somente após o processo de encapsulamento elas se tornam PDU's e podem ser transmitidos via protocolo. A figura 9 exemplifica esta denominação:

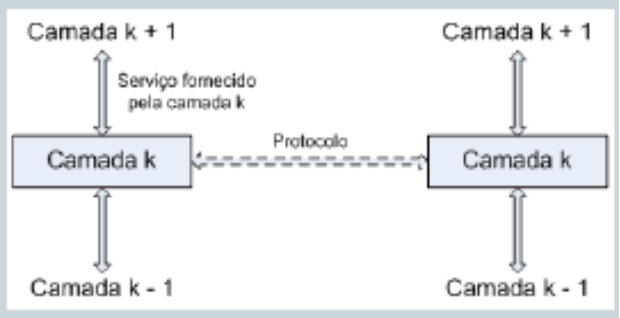

Figura 9 - Camadas do modelo OSI e sua denominação – fonte: [10]

Em resumo, dados PDU de uma camada K são os dados SDU da camada K-1. O processo de encapsulamento realizado pela camada K-1 adiciona cabeçalhos, *bytes* de controle e informações características daquela camada e transforma o SDU em PDU de camada K. Apenas após este processo é possível que a mensagem seja transmitida e recebida por outro dispositivo.

Para entender de maneira mais intuitiva a ideia de organização em camadas do modelo OSI, pode-se fazer uma analogia com uma informação que deve ser transmitida de uma pessoa para outra que mora em um local muito distante: as etapas de serviços, representados no modelo OSI como a troca de informações entre camadas de forma vertical, iniciam-se com uma pessoa ditando mensagem; outra pessoa digitando uma carta contendo a mensagem, que posteriormente será colocada em um envelope e transportada para outra localidade. A forma de comunicação direta entre camadas de um mesmo nível (comunicação horizontal) são os protocolos em estudo; a figura 10 apresenta o fluxo desta analogia:

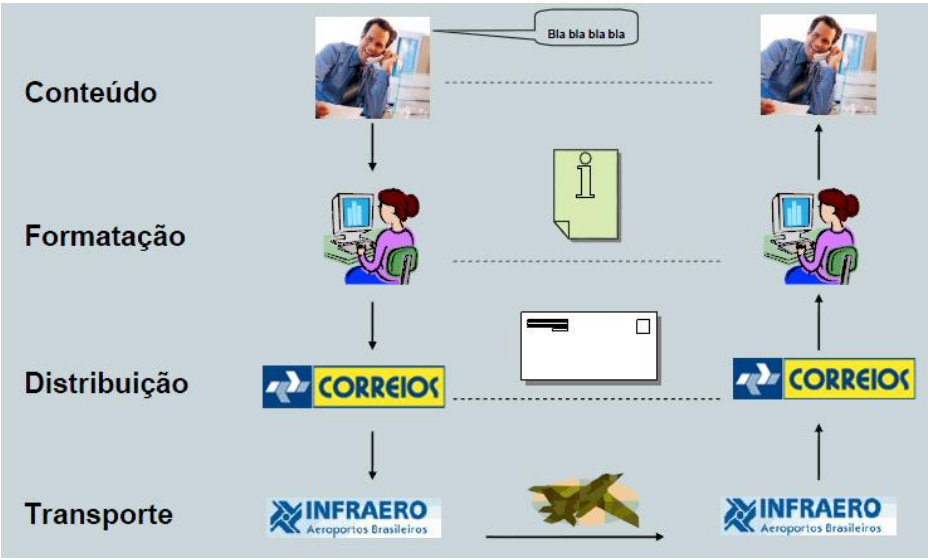

Figura 10 - Modelo OSI: analogia de organização - Carta – fonte: [10]

Detalha-se agora a função específica de cada camada dentro da transmissão como um todo.

2.3.1 Camada Física

A camada Física é a camada que lida com o maior nível de detalhes: é responsável pela transmissão de *bits* de um dispositivo a outro através do meio de comunicação físico. É ela quem lida com os sinais elétricos que representam os níveis lógicos “1” e “0”, além de lidar com as interfaces mecânicas, elétricas e funcionais do meio físico. Dentre os meios de transmissão mais comuns no mercado, podemos citar os cabos de par trançado, cabo coaxial, fibra óptica, etc.

A camada Física de um protocolo deve especificar os valores de tensão que serão atribuídos aos níveis lógicos, qual o tempo de duração de cada *bit*, como será o sentido da transmissão de dados (*simples, half-duplex* ou *full-duplex)*, qual será a função de cada pino dos conectores utilizados e como os dispositivos identificarão o início e o fim de uma conexão.

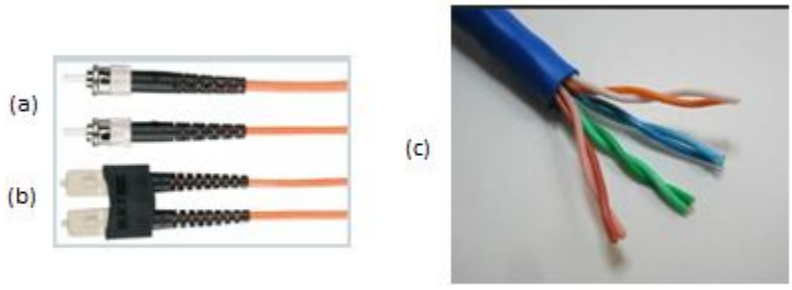

Figura 11 - Meios físicos de transmissão – (a) FO conector ST; (b) FO conector LC; cabo UTP (c)

2.3.2 Camada de Enlace

A camada de Enlace é a responsável pelo tráfego de grupos de *bits*, conhecidos como *frames.* É esta camada que converte o fluxo de dados sem formatação fornecido pela camada física em um fluxo de dados a ser utilizado pela próxima camada, a de Rede.

O envio dos *frames* deve ser feito de forma organizada e numa sequência bem definida, para que a mensagem possa ser reorganizada no lado do dispositivo receptor. Através de mensagens de confirmação, a camada de Enlace pode requisitar nova transmissão de *frames* já enviados caso não receba a confirmação de recebimento de algum quadro. Como citado anteriormente, as camadas inferiores fornecem dados para as superiores, portando a camada de Enlace recebe dados da camada Física.

É de responsabilidade da camada de Enlace controlar o acesso ao meio de transmissão (tecnologias como *token, CSMA,* por exemplo), além de regular o fluxo de dados entre máquinas com diferentes velocidades de transmissão. Também é nesta camada que se define o começo e o fim de cada quadro, gerando uma sequência de *bits*

padrão que se repetirá em todos os *frames* de dados e que são específicos em cada protocolo.

2.3.3 Camada de Rede

A camada de rede define o endereço de cada dispositivo na rede de computadores: através destes endereços é possível direcionar corretamente cada mensagem ao seu destinatário, podendo viajar através de redes distintas. É esta camada quem gerencia o congestionamento e tráfego de pacotes na rede, escolhendo as melhores rotas para diminuir o tempo de transmissão da mensagem (fato mais perceptível em redes de grande distância). Destaca-se que é possível realizar a troca de mensagens entre redes diferentes e com protocolos diferentes (e como consequência, *frames* com tamanhos diferentes).

2.3.4 Camada de Transporte

A camada de Transporte é responsável pela transmissão fim-a-fim, ou seja, gerencia a transmissão da mensagem desde a origem (transmissor) até o fim (receptor). Pode receber dados da camada acima dela, a camada de Sessão, e se a mensagem não puder ser transmitida de uma vez em um só *frame*, pode quebrá-la em pedaços menores para se adequar ao molde ditado pelo protocolo.

Um importante papel da camada de Transporte é a proteção das camadas superiores a ela em relação a mudanças no *hardware* de transmissão: é comum observar a evolução dos meios físicos de transmissão (como a utilização de fibras óticas em redes de alta velocidade, fato que vem se tornando comum nas redes atuais). Se a cada mudança de *hardware* fosse necessário realizar uma readequação nos aplicativos e algoritmos de transmissão, as redes ficariam estagnadas em meios de transmissão antigos pelo fato de ser muito complexo reorganizar toda a parte de *software* a cada mudança física, que é bastante comum. Dessa forma, a estruturação em camadas apresentada no modelo OSI é uma importante característica que possibilita o avanço nos métodos físicos de comunicação sem a preocupação de acarretar prejuízos à parte de *software*.

O *input* desta camada tem origem na camada de Sessão, que, ao requisitar uma conexão de transporte, obriga a camada a criar uma conexão de rede. A possibilidade de criar diversas conexões de rede para a transmissão de uma mesma mensagem ocasiona uma melhora no desempenho de transmissão conhecido como *throughput.* Por outro lado, também possibilita a multiplexação de várias conexões de transporte em uma mesma conexão de rede, reduzindo os custos de banda da rede. Tais ações são denominadas controle de fluxo.

2.3.5 Camada de Sessão

Esta camada é responsável pela criação de sessões entre aplicativos localizados em dispositivos diferentes, fato necessário para realizar procedimentos de *login.* Tais sessões são criadas a partir de um controle de sincronismo que é feito criando-se pontos de sincronização, que servem como *milestones* caso a conexão entre dispositivos seja perdida (é possível verificar até que ponto a mensagem já foi transmitida na ocorrência de erros).

Nos casos de arquiteturas de comunicação que acessam o meio físico através de *token,* é esta camada que realiza o gerenciamento do *token* (situações em que a rede não permite que mais de um dispositivo acesse o meio de comunicação simultaneamente).

2.3.6 Camada de Apresentação

A camada de Apresentação possui grande importância para transmissão de dados que requerem segurança contra acesso de dispositivos não autorizados: é ela que define o padrão de codificação de dados a ser utilizado, fornecendo serviços de criptografia, conversão de caracteres, entre outros que vários aplicativos diferentes poder requisitar.

2.3.7 Camada de Aplicação

Esta é a camada que realiza a interface com os aplicativos que requisitam acesso à rede de comunicação. Ela define serviços genéricos, como: suporte para transferência de arquivos, troca de mensagens, etc. Em outras palavras, é a porta de entrada do modelo OSI para os aplicativos que precisam se comunicar com outro dispositivo.

Capítulo 3 – Protocolo Modbus

Neste capítulo, foca-se o estudo do protocolo de comunicação Modbus, criado no final da década de setenta pela empresa Modicon (atualmente, Schneider Electric®) para ser utilizado na comunicação de controladores lógicos programáveis, os PLC's. Diferente dos protocolos baseados no padrão IEC61850 (que vêm crescendo a cada dia no número de usuários), o Modbus já possui espaço conquistado no mercado por ser um protocolo simples, robusto e de baixo custo, sendo muito utilizado em ambientes de **comunicação industrial**.

3.3 - Dados Históricos

Em sua criação, o protocolo Modbus foi projetado especialmente para aplicações industriais, ou seja, comunicação de dispositivos eletrônicos em ambientes com muitos ruídos e interferências. Pelo fato de ser um protocolo de livre acesso pelo domínio público, sua longevidade está garantida no mercado. Sua grande aceitação também pode ser explicada pela facilidade de instalação e manutenção, somados ao fato de aceitar quase que por completo a comunicação entre dispositivos de diferentes fabricantes (fato que se tornou ideal básico do IEC61850, como será visto em capítulo posterior).

Modbus é um protocolo da camada de aplicação do modelo OSI: ele possibilita a comunicação entre dispositivos baseando-se na relação servidor/cliente: mensagens são geradas apenas a partir de uma requisição. Os dados deste protocolo podem trafegar em redes TCP/IP através do encapsulamento da mensagem em *frames Ethernet*, apesar do protocolo ser tipicamente para meios de comunicação serial. A figura 12 mostra um exemplo de utilização do protocolo em uma rede que utiliza diferentes tipos de meios físicos.

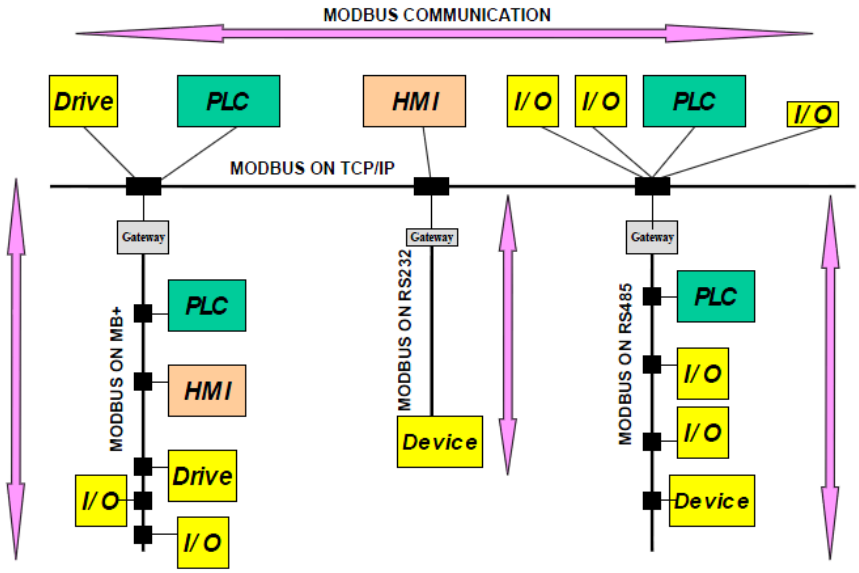

Figura 12 - Exemplo de sistema c/ protocolo Modbus - fonte: [2]

3.4 - Definições básicas

3.4.1 Estrutura do *frame* de dados

O protocolo Modbus é definido como um protocolo do tipo requisição/resposta, ou seja, somente a partir do elemento da mensagem denominado código de função - *function code* -, o servidor consegue entender que tipo de solicitação está sendo feita pelo cliente – não há gerações espontâneas de mensagem. A figura 13 mostra quais são as partes constituintes de um *frame* Modbus genérico:

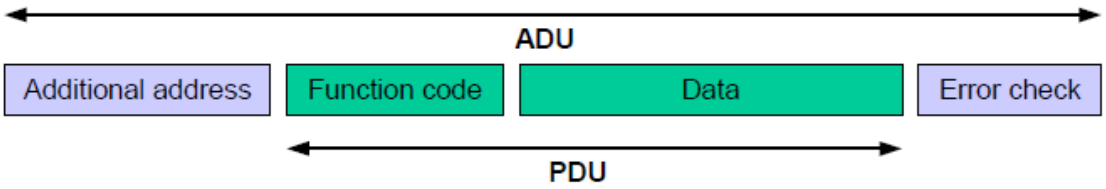

Figura 13 – Estrutura do frame Modbus – fonte: [2]

A unidade de dados do protocolo (*protocol data unit – PDU)* é composta pelo campo de dados – *data* – e pelo campo que contém o código de função solicitada pelo cliente ao servidor – *function code*. A utilização do protocolo em redes específicas pode gerar a necessidade de a mensagem carregar alguns dados adicionais, localizados no campo *additional address.* Agrupando os campos PDU, *additional address* e o campo de verificação de erros – *error check* – têm-se a unidade de dados de aplicação, a ADU (*application data unit*). O *frame* mostrado na figura acima é montado no início da transmissão, quando o cliente manda para o servidor a ação desejada.

O campo *Function Code* possui tamanho de um *byte*; é este campo que determina qual ação deverá ser tomada pelo servidor. Considera-se o número 0 um código inválido e, além disso, reserva-se a faixa de números [128;255] para códigos de exceção (que serão explicados com maiores detalhes a seguir).

O campo *Data* contém dados que podem ser necessários para realizar a ação especificada no campo *Function Code,* como registradores de endereço, informações de quantidades de dados, etc. Entretanto, existem funções que não necessitam de nenhuma informação adicional (o servidor sabe qual ação deve ser realizada apenas com a informação contida no campo *Function Code)*; nestes casos, o campo de dados possui tamanho 0.

Assim que o ADU, contendo a requisição feita pelo cliente, é recebido pelo servidor, podem ocorrer duas situações: se nenhum erro acontecer, o campo *Data Field* da resposta irá conter o dado solicitado pelo cliente, enquanto o campo *Function Code* é apenas ecoado de volta para o cliente com a mesma mensagem que foi recebida. Entretanto, em caso de ocorrência de erro, tanto o campo *Data Field* como o *Function Code* apresentarão códigos de exceção, que determinarão qual ação deverá ser tomada pelo cliente em consequência do erro apresentado. Ambos os casos estão esquematizados nas Figura 14 - Resposta livre de erro" e na Figura 15 - Resposta decorrente de erro":

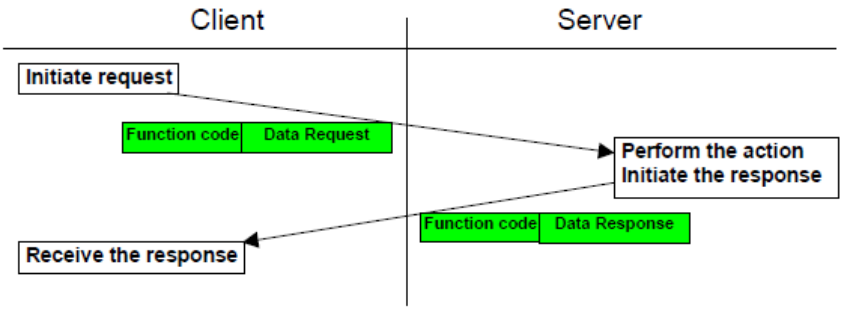

Figura 14 - Resposta livre de erro – fonte: [2]

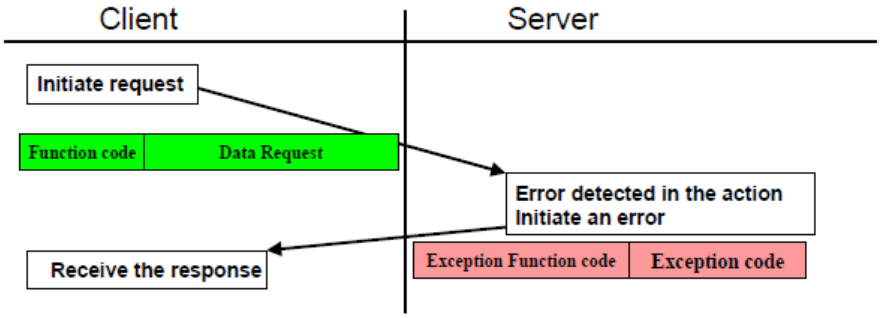

Figura 15 - Resposta decorrente de erro – fonte: [2]

Estabelece-se um tempo máximo de espera pela resposta vinda do servidor, visto que podem ocorrer situações em que um erro pode fazer com que o PDU de resposta nunca chegue, comprometendo o processo de comunicação.

Em resumo, o protocolo MODBUS estabelece três tipos de PDU's: a unidade de dados de requisição: *Request PDU,* sinalizada pelo bloco *mb_req_pdu;* a unidade de dados de resposta: *Response PDU,* sinalizada pelo bloco *mb_rsp_pdu;* e por último, a unidade de dados de resposta de exceção (decorrentes de erros no processo de comunicação): *mb_excep_rsp_pdu.* Tais blocos de *bytes* apresentam estrutura bem definida e carregam informações da função requisitada/respondida e também do conjunto de dados atrelado àquela função (que não é obrigatório). Estes blocos possuem a seguinte composição de *bytes*:

mb_req_pdu = {function_code, request_data}
function_code = [1 byte] (o código da função requisitada pelo cliente);
request_data = [n bytes] (código não obrigatório atrelado à função)

mb_rsp_pdu = {function_code, response_data}
function_code = [1 byte] MODBUS function code (é o código ecoado da requisição, em casos em que não houve ocorrência de nenhum erro).
response_data = [n bytes] (código não obrigatório atrelado à função)

mb_excep_rsp_pdu = {exception-function_code, request_data}
exception-function_code = [1 byte] MODBUS function code + 0x80 (código de função de exceção que sinaliza a ocorrência de um erro)
exception_code = [1 byte]

3.4.2 Os códigos de função (*function codes)*

Existem três tipos básicos de *function codes* disponíveis: códigos de função públicos (*public function codes)* – são bem definidos, com função única (nenhum outro código compartilha sua funcionalidade), reconhecido oficialmente e, portanto, devidamente testado e de funcionalidade garantida; códigos de função definidos pelo usuário (*user-defined function codes)* – possuem espaço reservados nas faixas 65 a 72 e 100 a 110 (números decimais) e, por serem definidos pelo usuário, criam funções não oficiais (e, portanto, não asseguradas e testadas, excluindo a garantia de funcionalidade e unicidade dos códigos pertencentes à categoria anterior); por último, os códigos de funções reservados (*reserved function codes),* que já são utilizados por determinadas empresas e que possuem exclusividade de sua utilização. A figura 16 mostra como é feita esta distribuição sequencial de códigos de função:

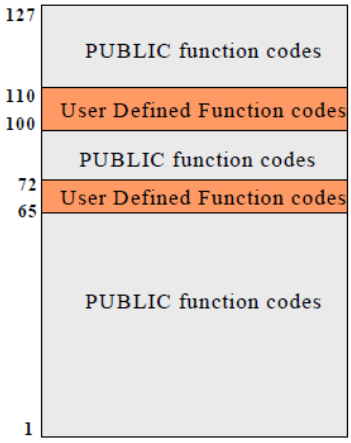

Figura 16 - Distribuição dos tipos de function codes – *fonte:* [2]

3.3 - Modelo de dados e diagrama de comunicação

O protocolo Modbus separa seus modelos de dados baseando-se em quatro tipos primários de tabelas: cada uma delas possui características inerentes ao tipo de objeto abordado e ao tipo de acesso requisitado ao dado, e é possível selecionar individualmente até 65536 itens por tabela. Os tipos de dados estão representados na Tabela 1.

Tabelas Primárias	Tipo de Objeto	Tipo de acesso	Observações
Entradas Discretas	*Bit* único	Somente leitura	Dados deste tipo podem vir de dispositivos de I/O
Bobinas	*Bit* único	Leitura/Gravação	Dados deste tipo podem ser alterados por aplicativos
Registradores de Entrada	Palavra de 16 *bits*	Somente leitura	Dados deste tipo podem vir de dispositivos de I/O
Registradores de Memória	Palavra de 16 *bits*	Leitura/Gravação	Dados deste tipo podem ser alterados por aplicativos

Tabela 1 - Tabelas primárias de dados MODBUS – fonte: [2]

O processo de recebimento de uma requisição vinda do dispositivo cliente pode ser mais bem visualizado na Figura 17: ao receber uma requisição, o servidor primeiramente valida o *function code;* em seguida, valida o endereço do dado e do conteúdo do dado e, só após estes estágios, realiza a execução em si do *function code.* Entre todas as validações, podem ocorrer falhas que geram diferentes códigos de exceção (*exception codes)*, que são enviados ao cliente juntamente com a função de exceção. O processo seguido tanto pelos dispositivos *client* quanto pelo do tipo *server* serão detalhados no tópico – Camada de Aplicação de Comunicação.

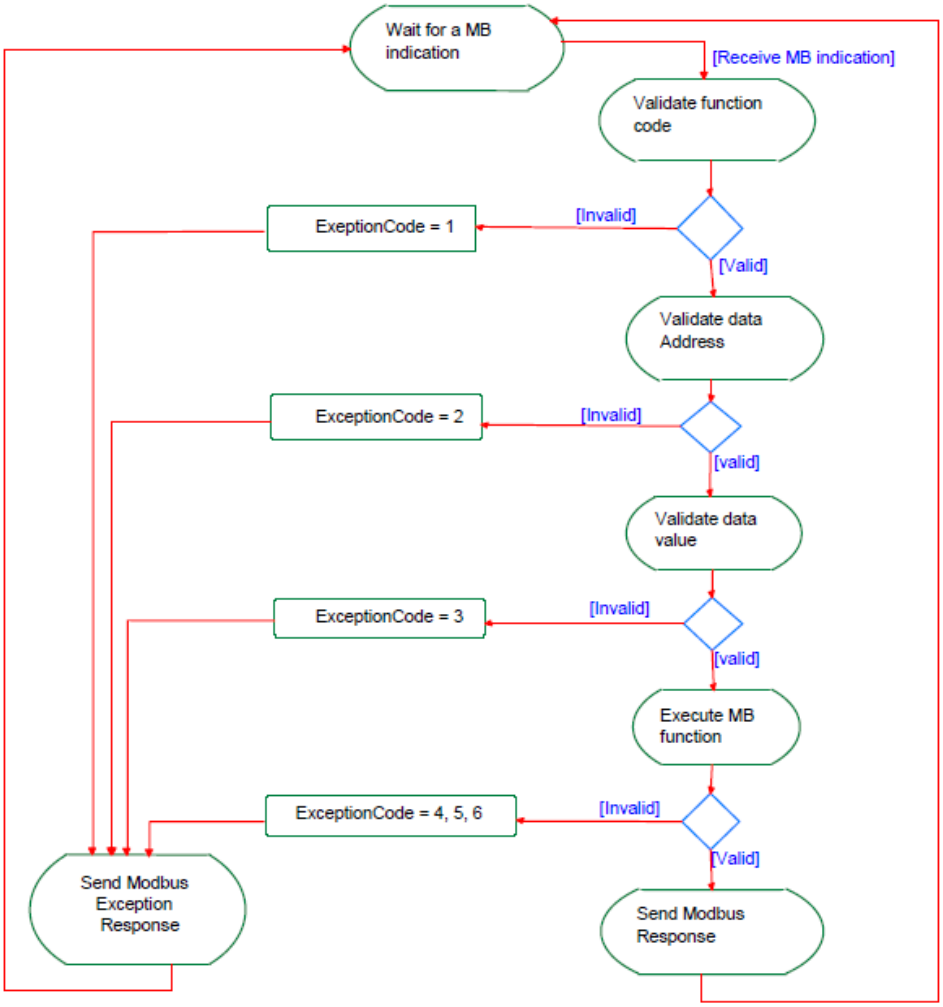

Figura 17 - Fluxograma de comunicação MODBUS - Fonte: [2]

Dentre as possíveis situações de erros, podem ocorrer: o cliente não recebe o *frame* de resposta por falha no meio de comunicação - neste caso, o tempo de espera da resposta é ultrapassado, e uma nova requisição é enviada; o servidor recebe o *frame* de requisição, mas a mensagem possui erros que podem ser de diversos tipos (paridade, dados corrompidos, etc.) – neste caso, nenhuma resposta é enviada ao cliente e o tempo de resposta também é ultrapassado; o servidor recebe a requisição, mas esta requisição é uma solicitação que não possui “sentido” – por exemplo, realizar a leitura de uma variável que não existe em determinado dispositivo: neste último caso, o cliente recebe as respostas de exceção.

Como já citado, nos casos livres de erros, a *function code* do pedido é apenas ecoada na resposta: todas as *function codes* possuem o *bit* mais significante (MSB) de valor 0; portanto, nos casos de ocorrência de erros de comunicação, o servidor acaba setando o MSB, justificando o fato do código de exceção ser somado de *80h* em relação ao código da requisição. Desta forma, o dispositivo cliente reconhece a ocorrência do erro e passa a analisar o campo de dados que pode conter informações a respeito do erro.

Para tornar esta relação cliente/servidor mais clara, examina-se o exemplo da figura 18. Neste exemplo, o cliente transmite ao servidor a necessidade de realizar a função 01h, que corresponde à ação de leitura do estado de uma saída de endereço *04A1h*, sendo esta a única saída a ser lida nesta ação (informação contida no *byte* "*Quantity of Outputs*" – *0001h*).

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	01	Function	81
Starting Address Hi	04	Exception Code	02
Starting Address Lo	A1		
Quantity of Outputs Hi	00		
Quantity of Outputs Lo	01		

Figura 18 - Exemplo de requisição seguida de erro – fonte: [2]

Neste caso, a saída especificada não existe no dispositivo de endereço especificado; dessa forma, o servidor responde com a função acrescida de 80h (setando o *bit* mais significativo) e com o código de exceção 02h, que representa endereço de dado ilegal na tabela de funções de exceção.

3.4 – Encapsulamento de mensagens Modbus em redes TCP/IP

A partir do modelo de dados básico das mensagens Modbus apresentado anteriormente, é possível aprofundar um pouco mais nos detalhes da aplicação do protocolo em redes *Ethernet* baseadas no protocolo de transporte TCP/IP. A definição cliente/servidor pode ser mais bem detalhada na Figura 19, que cita as diferentes classificações de mensagens possíveis neste protocolo:

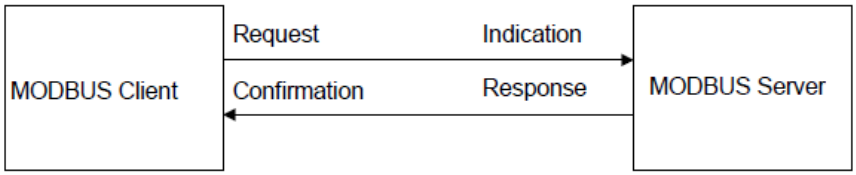

Figura 19 - Tipos de mensagens na relação cliente/servidor - fonte: [3]

As categorias de mensagens possuem nomes bastante intuitivos: a mensagem de requisição (*Request*) se dá no início da comunicação, momento em que o cliente envia através da rede a informação contendo o que deve ser lido do servidor; chegando neste último, o dispositivo destinatário tem indicação (*indication*) da ação que deve ser realizada. Fazendo o caminho de volta, o servidor transmite ao cliente a mensagem do tipo resposta

(*response*), que indica ao iniciador da comunicação a confirmação (*confirmation)* da transação.

Esta sequência de troca de dados se dá, por exemplo, entre IED's de uma mesma rede, aplicativos de supervisão e dispositivos ao nível de *bay*, etc. Uma melhor visualização de um possível sistema com aplicação Modbus baseado em *Ethernet* está representado na Figura 20, em que se misturam dispositivos conectados num mesmo meio *Ethernet* com uma sub-rede de meio *serial,* utilizando como interface um *gateway.*

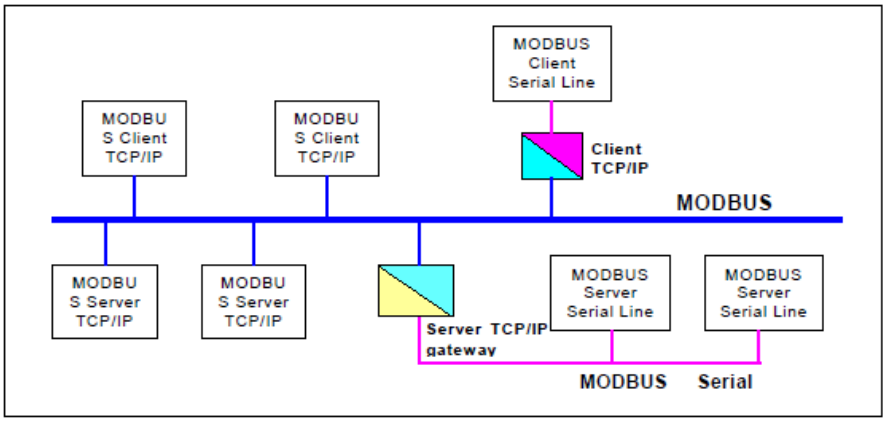

Figura 20 - Exemplo de arquitetura Modbus TCP/IP – fonte: [3]

Em sistemas de energia, uma das principais características necessárias para o correto monitoramento da rede é a informação de tempo dos eventos ocorridos: é primordial ter o conhecimento se determinados eventos ocorreram ao mesmo tempo, exigindo sincronismo de tempo perfeito entre aparelhos conseguido através de dispositivos GPS. A comunicação Modbus serial por si só não possui estampa de tempo, fato que poderia se tornar um fator impeditivo de sua utilização em tais sistemas; entretanto, o encapsulamento das mensagens Modbus para posterior tráfego em redes *Ethernet* consegue adicionar esta informação ao *frame* de encapsulamento a partir dos dados contidos no IED.

Focando agora no *frame* da mensagem, a estrutura vista na Figura 13 – Estrutura do *frame* Modbus – fonte: sofre algumas alterações ao encapsular-se a uma rede TCP/IP: elimina-se o bloco de verificação de erro e o bloco de endereço, dando lugar ao bloco *Modbus Application Protocol Header – MBAP,* ou seja, uma espécie de "bloco-cabeçalho" que é composto pelos dados apresentados na Tabela 2:

Campos	Tamanho	Descrição	Cliente	Servidor
Identificador da Transação	2 *bytes*	Identificador de uma transação do tipo requisição/resposta	É iniciado pelo cliente	É copiado pelo servidor a partir da requisição
Identificador do Protocolo	2 *bytes*	0 = protocolo Modbus	É iniciado pelo cliente	É copiado pelo servidor a partir da requisição
Tamanho da mensagem	2 *bytes*	Contém o número de *bytes* na sequência	É iniciado pelo cliente em mensagens de requisição	É iniciado pelo servidor em mensagens de resposta
Identificador da unidade	1 *byte*	Identificador de um dispositivo escravo remoto	É iniciado pelo cliente	É copiado pelo servidor a partir da requisição

Tabela 2 - composição do cabeçalho MBAP – fonte: [3]

O identificador de transação – *transaction identifier* – é utilizado no pareamento da mensagem: o servidor apenas copia na resposta o identificador que foi recebido na requisição, relacionando corretamente cada resposta à sua respectiva pergunta; o identificador de protocolo – *protocol identifier* – é utilizado em multiplexação interna da mensagem, e é definido no padrão Modbus pelo valor 0; o campo referente ao tamanho da mensagem – *length* - é um contador de *bytes* que inclui tanto os próprios dados da mensagem como o identificador de unidade – *unit identifier* - utilizado na comunicação através de componentes que agrupam diversos dispositivos Modbus conectados por uma linha *serial* através de um único endereço de IP, como roteadores, *gateways, bridges, etc.* Este campo é preenchido pelo dispositivo cliente e deve ser apenas copiado pelo servidor na mensagem de resposta. A Figura 21 mostra a nova disposição do *frame* Modbus:

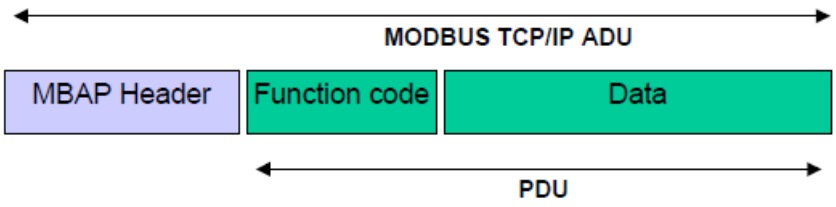

Figura 21 - Frame Modbus em redes TCP/IP - fonte: [3]

3.5 – Descrição funcional

3.5.1 – Modelo genérico de uma arquitetura de comunicação Modbus

Para entender a dinâmica da comunicação entre dispositivos adeptos do protocolo Modbus, é necessário estabelecer o fluxo dos dados desde o momento da criação de uma

requisição até o recebimento de uma confirmação a respeito deste pedido. Um modelo genérico de um fluxo de troca de mensagens Modbus poderia apresentar todo dispositivo como cliente e servidor: em redes de energia, é usual a parametrização de todos os IED's como clientes e servidores, possibilitando assim que todo IED seja capaz de receber e requisitar dados de outros IED's, tendo como consequência a possibilidade da criação de lógicas de intertravamento. A Figura 22 mostra como uma arquitetura de comunicação baseada em Modbus está organizada:

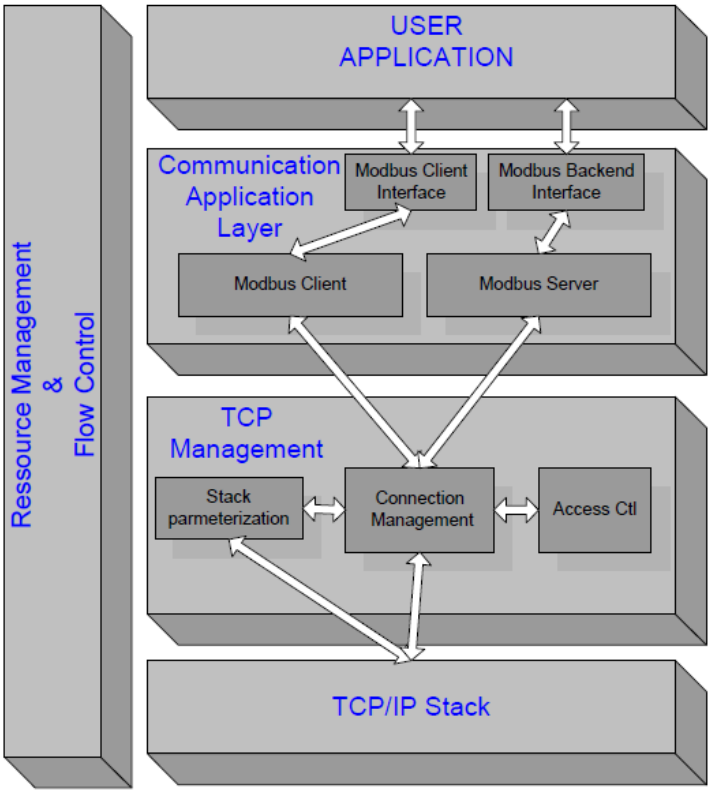

Figura 22 - Organização de uma arquitetura Modbus – fonte: [3]

A quantidade de interfaces de um dispositivo depende de sua funcionalidade na rede: dispositivos definidos como clientes e servidores terão interfaces para cada uma dessas definições. Observando a figura acima, nota-se que é possível delegar o controle da troca de mensagens ao nível *User Application* – aplicação nível usuário - através da interface *Modbus Client.* A partir de uma mensagem do tipo requisição criada pelo usuário baseando-se em parâmetros fornecidos pelo nível de aplicação, utiliza-se a interface Modbus para alcançar o nível de cliente, esperando até que haja uma resposta de confirmação vinda do servidor.

Do outro lado desta ponte de comunicação está o *Modbus Server*, que aguarda o recebimento da mensagem de requisição. Confirmado o recebimento, o módulo servidor inicia um processo ou de leitura, ou de escrita, ou outra ação imposta pelo módulo de cliente. Todo este processo é completamente invisível ao nível de aplicação do usuário, que apenas aguarda a chegada da resposta referente à requisição enviada. Em suma, a

principal tarefa do módulo servidor é aguardar a chegada de uma mensagem requisição, processá-la e transmitir de volta a confirmação da ação tomada (ou a sinalização de algum possível erro). Vale ressaltar a interface entre o servidor e o nível de aplicação do usuário, a interface *Backend:* ela é responsável pela definição dos objetos presentes no nível de aplicação.

É evidente que todo este processo de gerenciamento entre os módulos de cliente, servidor e interfaces cria a necessidade de um módulo de controle: o módulo *Connection Management* é o responsável por esta ação, realizando o controle dos canais de comunicação *Ethernet* por nível de aplicação de usuário, ou então de forma independente através de um módulo dedicado (desde que o nível de usuário tenha total acesso a todos os dados deste controle). O próximo tópico explica com mais detalhes como este módulo de controle é dividido.

3.5.2 – Módulo de gerenciamento de conexões

Para que seja possível efetuar uma troca de dados utilizando o protocolo Modbus, é necessário primeiramente estabelecer uma conexão TCP entre o cliente e o servidor. Esta conexão pode ser criada tanto pela aplicação do usuário ou então pode ser criada automaticamente pelo módulo de gerenciamento de conexões TCP: no primeiro caso, existe uma maior flexibilidade ao usuário, mas ocorre um aumento na complexidade do programa de interface ao usuário; no segundo caso, este gerenciamento de novas conexões TCP fica completamente invisível à aplicação do usuário, que se responsabiliza apenas pela emissão e recebimento de mensagens. A segunda opção é a mais comum nos sistemas de comunicação Modbus, além de que as conexões que são estabelecidas ao iniciar-se uma transição Modbus permanecem “ligadas” não apenas durante a transmissão, mas até que algum comando de *reset* seja recebido.

Vale ressaltar que diversas transações Modbus podem ser efetuadas numa mesma conexão TCP, sendo de responsabilidade do *transaction identifier* (visto no tópico 3.4) identificar quais requisições correspondem a quais respostas. Além disso, em situações de tráfego de dados bidirecional (situações em que componentes atuam tanto como cliente como servidor), é necessário existir duas conexões distintas para tráfego de dados de cliente e tráfego de dados de servidor. Cabe ao módulo de gerenciamento controlar a quantidade de conexões estabelecidas; caso o número limite se exceda, sugere-se a exclusão da conexão mais antiga.

No quesito segurança, o módulo de gerenciamento de conexões possui a ferramenta “*acess control option”* – opção de controle ao acesso – que possibilita ao servidor negar o fornecimento da mensagem de resposta a uma mensagem de requisição vinda de um cliente não autorizado. Para que tal ferramenta seja utilizada, é necessário que o cliente forneça uma lista de IP’s contendo quais são os clientes autorizados a receber mensagens dos servidores. Por *default,* todo endereço de IP não incluso na lista é considerado um dispositivo não autorizado.

Para que haja um melhor desempenho no controle de conexões iniciadas/encerradas, o módulo divide as conexões em duas categorias principais: a primeira

categoria é denominada *priority connection pool* – as conexões de prioridade: tais conexões nunca são fechadas por requisições locais, pois são configuradas a partir de uma lista de IP's de dispositivos prioritários que, ao demandarem a abertura de uma nova conexão, retirarão desta "cesta" de conexões prioritárias a quantidade necessária para suprir sua necessidade. É necessário estipular uma quantidade máxima de conexões prioritárias por dispositivo, visando não "monopolizar" a cesta de conexões. Os dispositivos que pertencem a esta lista são denominados dispositivos *marcados.*

Na segunda categoria de tipo de conexões, *non-priority connection pool,* as conexões não prioritárias, enquadram-se as conexões realizadas por dispositivos não listados; neste caso, aplica-se a regra citada anteriormente: ao receber uma requisição de nova conexão, e se todo o grupo disponível já tiver sido utilizado, fecha-se a conexão mais antiga para dar espaço a mais nova.

Como já citado, a transmissão de dados é iniciada a partir do envio de uma mensagem de requisição. Para que este envio seja possível, é necessário que a conexão TCP já esteja estabelecida; esta conexão é determinada a partir do endereço IP do dispositivo servidor da transação. A conexão TCP não pode ser encerrada enquanto a mensagem estiver sendo transmitida, e a existência de diversas conexões até um mesmo dispositivo não impossibilita o envio de dados (cabe ao módulo de gerenciamento decidir qual conexão utilizar). Vale ressaltar que o dispositivo cliente não precisa aguardar a finalização de transações antigas para iniciar uma nova transmissão, mas é de sua responsabilidade sinalizar o encerramento da conexão quando toda a mensagem for transmitida.

3.5.3 – Camada de Aplicação de Comunicação

4 Dispositivos Modbus do tipo Client

O funcionamento dos dispositivos de categoria "cliente" no protocolo Modbus (blocos em destaque na figura abaixo) pode ser esquematizado de acordo com o fluxograma da Figura 24. Neste fluxograma, pode-se visualizar que dispositivos deste tipo podem receber três diferentes tipos de evento: a demanda vinda do aplicativo do usuário para criar uma mensagem de requisição (*Modbus request);* uma mensagem de resposta vinda do módulo de controle TCP (*Modbus response);* uma *flag* sinalizando que o tempo de resposta expirou (*time-out response).*

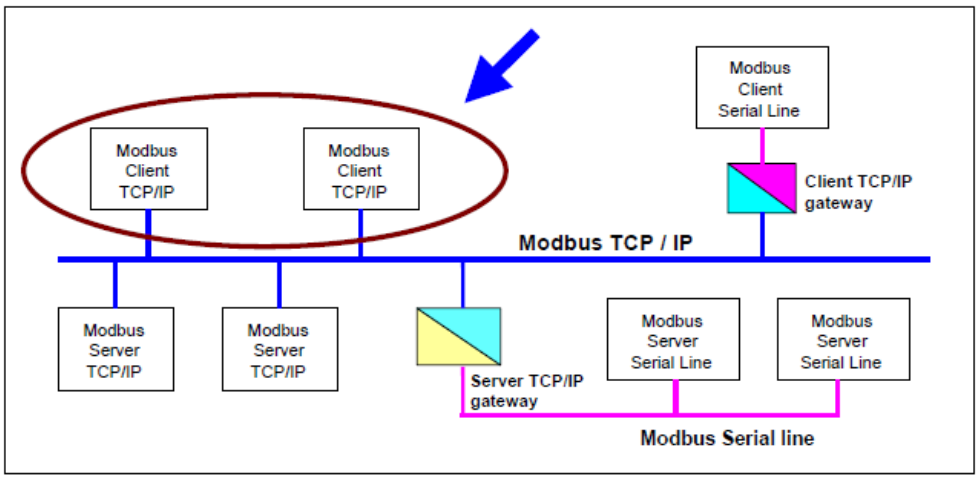

Figura 23 - Destaque para dispositivos da categoria "Cliente" - fonte: [3]

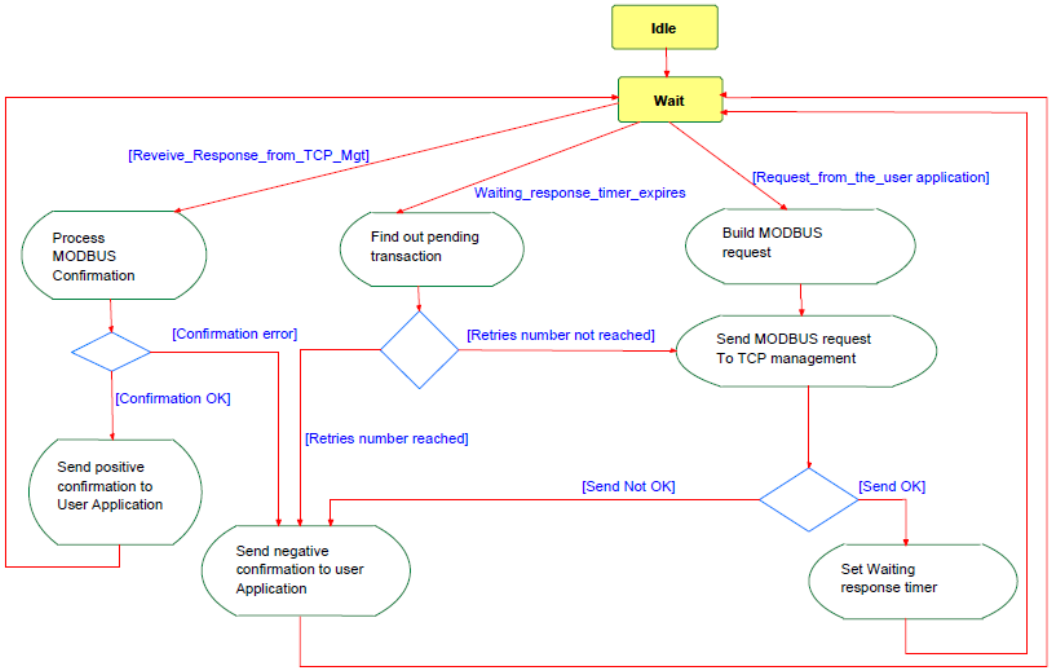

Figura 24 - Fluxograma de ações de um dispositivo tipo Client *- fonte: [3]*

Detalha-se a seguir cada um destes três casos com maiores detalhes:

5 Construção de uma mensagem tipo *Request*

No primeiro caso, o dispositivo *client* deve codificar a mensagem de requisição de serviço a ser mandada ao *server* e enviá-la utilizando o módulo de gerenciamento de conexões TCP. O processo de construção de uma mensagem tipo *request* pode ser

subdividido em sub-tarefas: primeiramente, deve-se determinar quais informações devem ser armazenadas, de forma que, assim que a mensagem de resposta chegue, seja possível relacionar os dados da mensagem *request* aos dados da mensagem *response.* O próximo passo é construir o *frame* Modbus encapsulado (analisado na Figura 21) a partir de dados fornecidos pelo aplicativo de usuário: somente a partir do fornecimento destes dados é possível construir o cabeçalho e o PDU que constituem o *frame*. Por último, realiza-se a transmissão da mensagem ao módulo de gerenciamento de conexões, que precisa do endereço IP do dispositivo *server* para que a mensagem seja corretamente direcionada. A figura a seguir resume os passos deste processo:

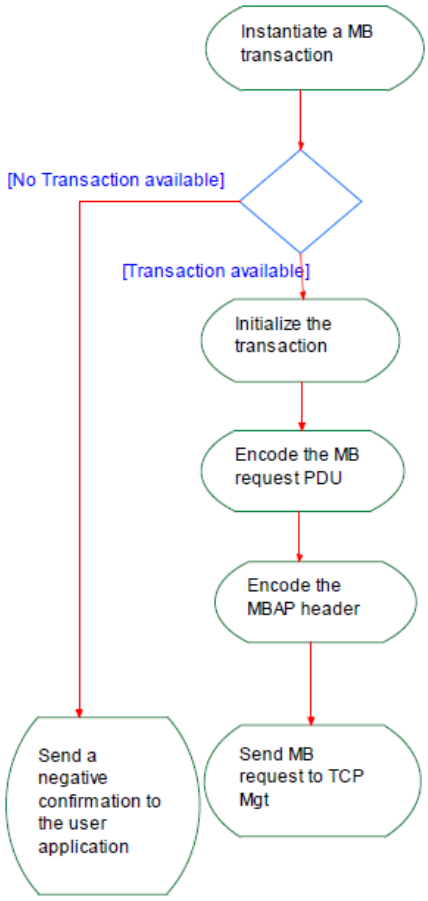

Figura 25 - Fluxograma de construção de mensagens tipo Request - *fonte: [3]*

6 Recepção de uma mensagem tipo *Response*

Neste segundo caso, o dispositivo precisa analisar o conteúdo da mensagem tipo *response* vinda de um servidor e informar ao aplicativo de usuário qual tipo de ação foi ou deve ser tomada. A primeira ação realizada é a verificação do bloco *Transaction Identifier* do cabeçalho, parte que mostra a qual mensagem tipo *request* aquela resposta está atrelada. Se não houver nenhuma mensagem de requisição aguardando resposta, tal mensagem é descartada. Entretanto, se a mensagem tipo *response* for de fato resposta a

uma requisição pendente, esta mensagem passará por uma análise de seu conteúdo para que seja reportada ao aplicativo de usuário.

A análise pela qual a mensagem é submetida consiste dos seguintes passos: verificação do bloco *Protocol Identifier*, que deve ser 0 (vide Tabela 2 - composição do cabeçalho MBAP); verificação do tamanho da mensagem; verificação da origem da mensagem: se a resposta se originou de um dispositivo conectado diretamente na rede TCP/IP, a própria identificação de conexão TCP é suficiente para identificar o dispositivo *server,* sem ambiguidade; entretanto, se a mensagem tem como origem uma sub-rede (vinda de um *gateway, bridge, router),* é necessário verificar o bloco *Unit Identifier,* que possui a informação do *server* que enviou a mensagem. Por último, ocorre a verificação do bloco PDU: se o *function code* recebido for o mesmo que foi enviado, uma confirmação positiva é enviada ao aplicativo de usuário; se o bloco apresentar o *function code* adicionado de 80h, significa que um erro ocorreu, ocasionando no envio de uma confirmação positiva de exceção ao aplicativo de usuário. Se o *function code* encontrado não for nem um código de exceção e nem o código transmitido na requisição, transmite-se uma confirmação negativa à aplicação de usuário, sinalizando um erro de comunicação. Destaca-se que uma confirmação positiva significa que o dispositivo *server* recebeu a mensagem de requisição, mas não implica que o dispositivo conseguiu realizar a ação requisitada; a falha na realização do ato requisitado é de responsabilidade dos códigos de exceção. Esta cadeia de decisões pode ser melhor visualizada na Figura 26:

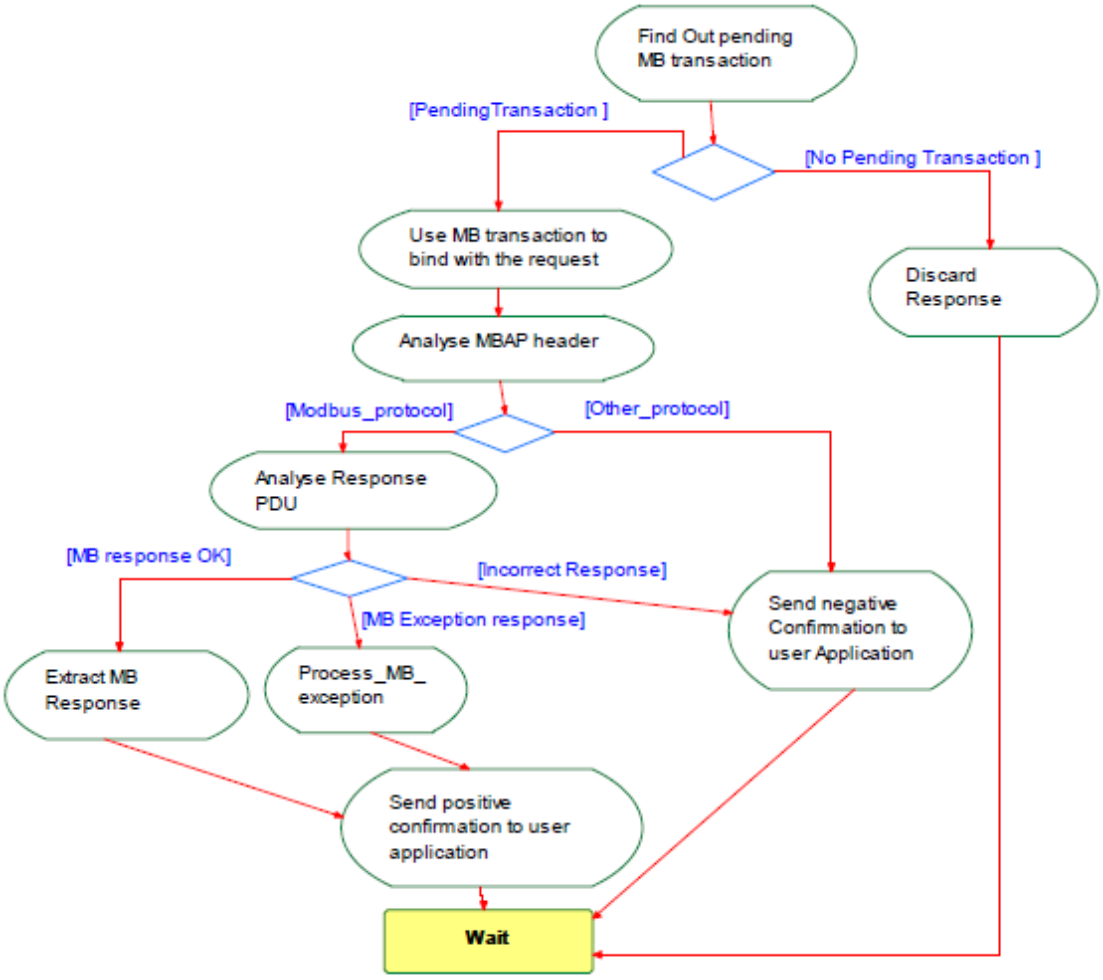

Figura 26 - Fluxograma de ações de confirmação ao User Application - *fonte: [3]*

7 Tempo de resposta expirado – *Time-out*

Neste último caso, cabe ao dispositivo *client* decidir se envia uma nova mensagem ao *server* requisitando novamente a informação, ou se basta comunicar à aplicação de usuário que a comunicação não obteve mensagem de resposta no tempo previsto. Não existe nenhum parâmetro de tempo pré-determinado para caracterizar o *time-out* de uma mensagem, visto que as diferenças de arquiteturas e tecnologias utilizadas na criação de redes de comunicação interferem diretamente na velocidade com que os dados trafegam na rede. Entretanto, o bom senso diz que os tempos de *time-out* devem ser estabelecidos com folgas superiores aos tempos esperados, evitando congestionamento na rede com informações repetidas. Em redes de aplicações não críticas, é comum encontrar intervalos de tempo excedido na ordem de segundos.

3.5.4 – *Dispositivos Modbus do tipo Server*

Os dispositivos do tipo *server* possuem o papel de oferecer acesso a aplicativos e a serviços de dispositivos remotos do tipo *client.* A Figura 27 destaca estes dispositivos na rede que vêm sendo utilizada como exemplo neste capítulo.

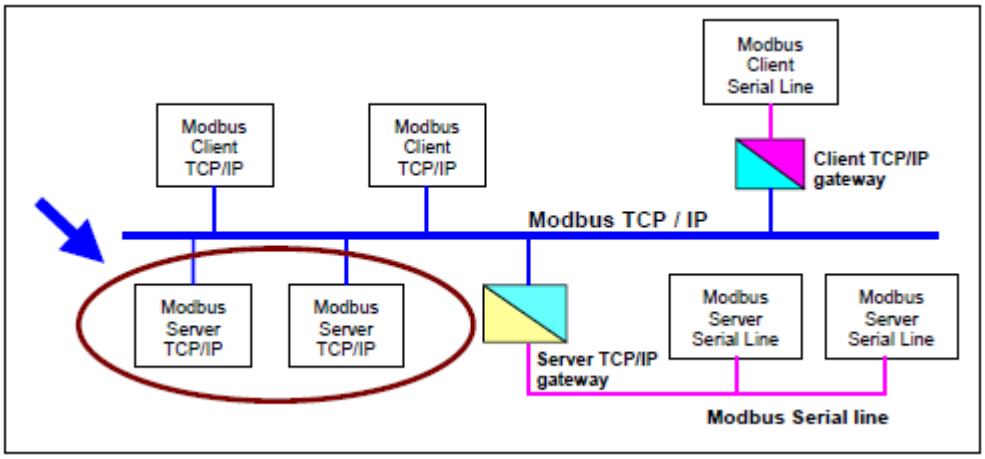

Figura 27 - Destaque para dispositivos do tipo Server - *fonte:* [3]

O tipo de acesso realizado aos dispositivos *server* depende do aplicativo do usuário: o acesso pode ser simplesmente para adquirir ou gravar uma variável, ou pode ser para iniciar um serviço específico atrelado a um objeto de aplicação. Cabe ao servidor Modbus analisar uma mensagem do tipo *request* enviada pela rede, processar a ação indicada na mensagem e retornar o resultado através de uma mensagem do tipo *response.*

Na Figura 28, são resumidas as possíveis ações entre o recebimento de uma requisição até o envio da resposta. Percebe-se que alguns serviços podem ser processados diretamente pelo servidor, de forma independente do aplicativo de usuário, enquanto outros necessitam de pelo menos uma iteração; alguns serviços especiais requerem uma interface específica para serem iniciados, denominada “*Modbus Back End Service”:* estes serviços são iniciados a partir de uma sequência específica de ações de requisição/resposta, que são monitorados por esta interface até que determinada condição seja alcançada.

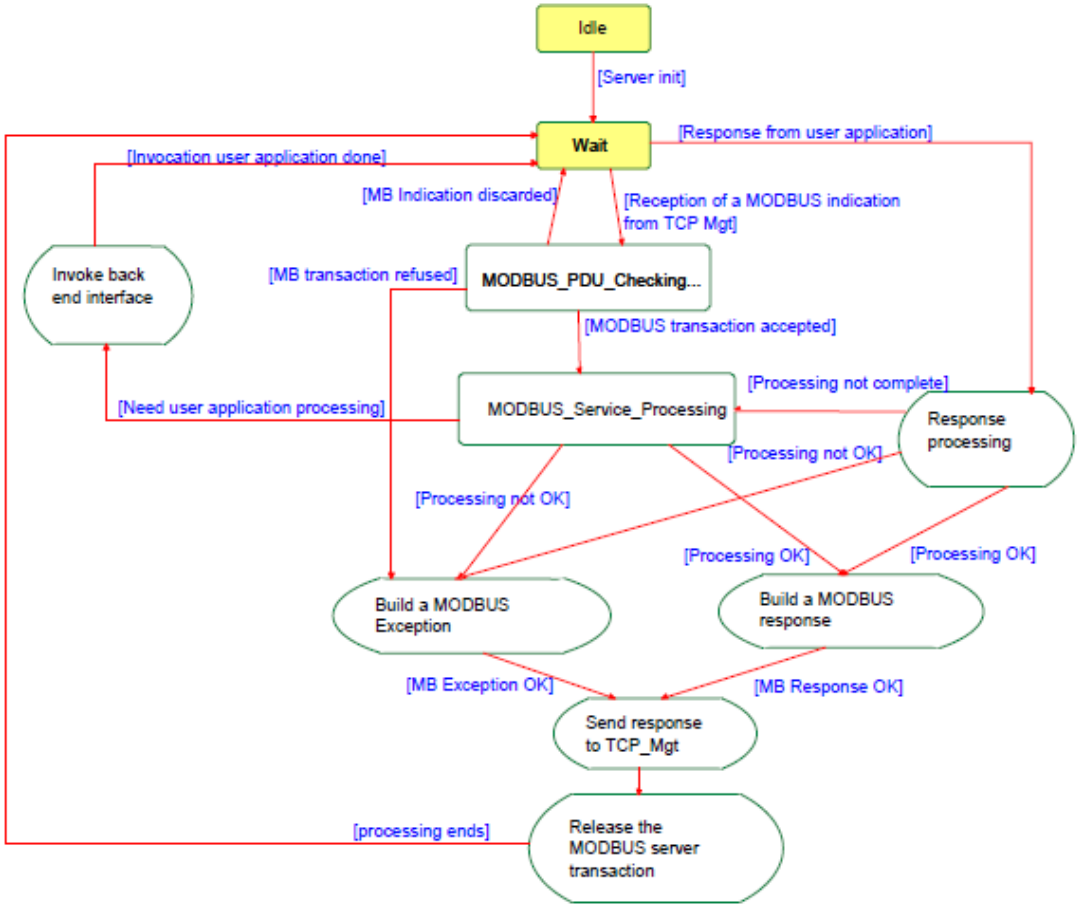

Figura 28 - Fluxograma de ações de um dispositivo Server - *fonte:* [3]

Servidores podem aceitar receber diversas requisições simultaneamente de diferentes dispositivos *client,* dependendo da capacidade de memória e de processamento do dispositivo. É importante saber que o número de conexões simultâneas a um mesmo dispositivo tipo *server* é inversamente proporcional ao tempo de resposta de cada requisição, individualmente falando.

Analisando detalhadamente a primeira ação realizada pelo dispositivo *server* ao receber uma mensagem vinda do módulo de gerenciamento TCP (bloco *MODBUS_PDU_Checking* da Figura 28), pode-se ampliar este bloco no fluxograma mais detalhado mostrado na Figura 29: primeiramente, verifica-se o cabeçalho da mensagem para se certificar de que se trata realmente de uma mensagem de protocolo Modbus; em seguida, avalia-se o tipo de transação requisitada, para finalmente iniciar o processo indicado. Nota-se que, caso o servidor tenha atingido o número máximo de processamento de requisições, é retornada a mensagem de servidor ocupado (*Server Busy).*

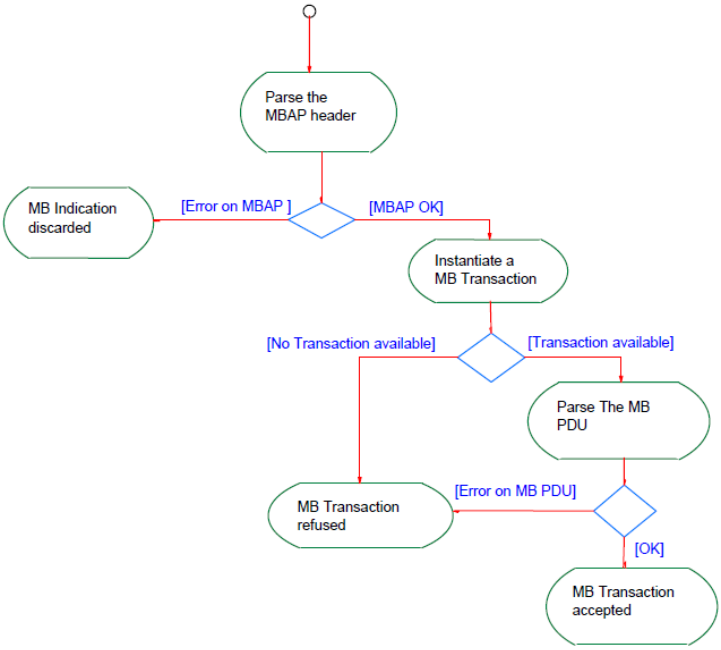

Figura 29 - Detalhe do bloco de análise do PDU - fonte: [3]

Nos casos em que o servidor não está ocupado e que a transação é aceita e está disponível, são aquisitados os seguintes parâmetros: a identificação da conexão TCP que foi utilizada no transporte da mensagem (informação fornecida pelo módulo de conexão TCP); a identificação da transação Modbus (presente no cabeçalho do *frame*); o identificador da unidade (também presente no cabeçalho). De posse de todos estes dados, processa-se o código da função da mensagem, podendo resultar em um código de exceção (servidor não consegue realizar a ação, por algum motivo) ou o início do processamento do serviço Modbus – *Modbus Service Processing,* que possui um quadro de ações possíveis detalhado no fluxograma da Figura 30:

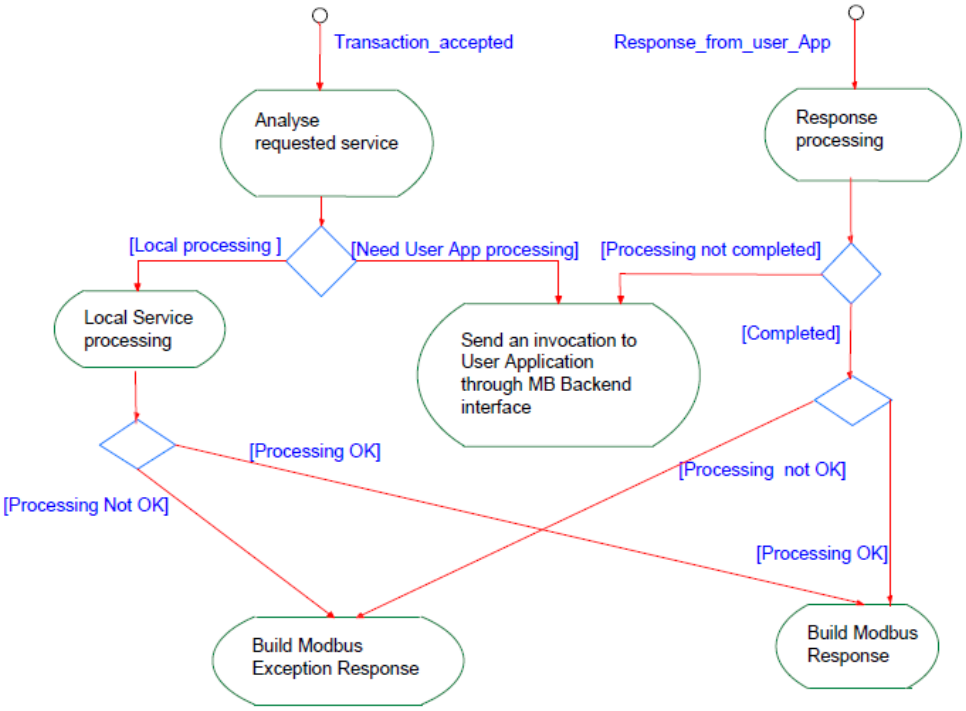

Figura 30 - Fluxograma de ações - processamento de serviços - fonte: [3]

A análise do fluxograma da figura acima mostra dois possíveis caminhos para o processamento dos serviços: caso o dispositivo servidor seja capaz de acessar diretamente dados contidos no aplicativo de usuário, o processamento é feito localmente (no próprio servidor), sem a necessidade do uso da interface *BackEnd.* Já em dispositivos multiprocessados em que as camadas de comunicação e as camadas de aplicação de usuário são entidades diferentes, podem ocorrer tanto situações de processamentos de serviços de forma local como a necessidade da utilização da interface *BackEnd* para controlar a troca de informação entre estas duas entidades.

Finalmente, a última etapa da comunicação – a mensagem de resposta – deve ser montada para posterior envio ao dispositivo cliente. Como já visto no tópico “Os códigos de função (*function codes*)”, a mensagem do tipo *response* pode indicar tanto uma resposta positiva (ecoando a *function code)* ou uma resposta de exceção, contendo informações a respeito do erro detectado durante o processamento (*function code* + 80h). Os códigos de exceção mais comuns estão representados na Tabela 3:

Código de exceção	Nome Modbus	Descrição
01	Código de função ilegal	Código de função não conhecido pelo servidor
02	Endereço de dados ilegal	Dependente da requisição
03	Valor de dado ilegal	Dependente da requisição
04	Falha de servidor	Servidor falhou durante a execução
05	Reconhecimento	Servidor reconhece a requisição, mas por ser muito longa, ainda não envia a resposta
06	Servidor ocupado	Servidor não recebe a requisição. Cabe ao cliente decidir quando será feito o reenvio
0A	Erro de Gateway	Endereço de gateway não disponível
0B	Erro de Gateway	Dispositivo alvo não responde

Tabela 3 - Códigos de exceção - fonte: [3]

Monta-se também o cabeçalho da mensagem de resposta, que contém as informações do identificador de unidade (copiado da requisição), tamanho (Modbus PDU + *byte* do *Unit Identifier*), identificador de protocolo (copiado da requisição, 0 para Modbus) e identificador de transação (também copiado da requisição, justamente para atrelar corretamente a resposta à pergunta). Com todos estes dados preparados e montados no *frame*, o servidor utiliza o módulo de gerenciamento de conexões TCP para devolver a mensagem ao dispositivo *client.*

Capítulo 4 - DNP3 – *Distributed Network Protocol*

Neste capítulo, apresenta-se uma breve descrição do funcionamento do conjunto de protocolos denominado DNP – Protocolo de Rede Distribuída, mais especificadamente a versão 3.0 do conjunto – DNP3.

O protocolo DNP3 foi extensivamente utilizado em sistemas de aplicações de energia e água; a utilização em outros campos é um pouco mais rara. Seu desenvolvimento teve como enfoque possibilitar a comunicação entre componentes de sistemas SCADA, interligando o sistema supervisório com os RTU's e os IED's. Atualmente, o crescimento do número de novos sistemas com este protocolo sofreu redução pelo fato de novos sistemas estarem dando preferência ao padrão IEC61850, mais eficiente e de menor custo; entretanto, o protocolo DNP3 ainda é bastante utilizado no campo de energia e distribuição, e, muitas vezes, opta-se por manter um protocolo já existente em uma planta do que recomeçar da estaca zero.

O protocolo DNP3 possui características mais complexas se comprado ao protocolo Modbus (visto no capítulo anterior); entretanto, por possuir uma maior complexidade, tornou-se mais robusto, eficiente e interoperável do que este último. Analisando o modelo OSI, apresentado no

Capítulo 2 – Revisão de Conceitos de Redes, caracteriza-se como um protocolo de nível 2 (camada de Enlace), possibilitando ações de multiplexação, fragmentação de dados, checagem de erros, priorização no envio de dados (característica muito importante em aplicações de energia) e definições de funções de transporte e também de aplicação (camadas 4 e 7, respectivamente).

4.1 - Dados Históricos

O protocolo DNP3 teve sua criação impulsionada pela demora da finalização do padrão IEC60870-5; o mercado ansiava por um padrão que focasse no quesito interoperabilidade de equipamentos de diferentes marcas utilizados nos sistemas SCADA. Dessa forma, no início da década de noventa, a empresa Westronic® utilizou os moldes inacabados do padrão IEC60870-5 para criar um protocolo que atendesse imediatamente e, principalmente, os requisitos das plantas de energia e água do território norte-americano: o DNP3.

Buscou-se a criação de um padrão que fosse imune às interferências eletromagnéticas (fortemente presentes nas áreas de aplicação) e que possibilitasse a transmissão de dados em meios físicos de baixa qualidade. Por ter sido criado de forma a suprir necessidades momentâneas, o protocolo obteve baixo desempenho no quesito

segurança nos primeiros anos de utilização. A área de *smart grids* muitas vezes requer que o sistema seja acessível por clientes em diferentes localidades através de redes *Ethernet*, fato que expõe o sistema a possíveis ataques via Internet. Dessa maneira, o protocolo correu sério risco de ser abandonado logo em seu início, obrigando seus utilizadores a criar soluções de segurança que garantissem a confiabilidade dos sistemas. Atualmente, tais níveis de segurança foram atingidos através de algoritmos sofisticados de criptografia dos dados, obtendo a aprovação por parte da IEC6231-5, padrão que regula os níveis de proteção inclusive de outros protocolos.

4.2 - Características Gerais

Seguindo o mesmo conceito do modelo OSI apresentado no Capítulo 2, o protocolo DNP3 pode ser dividido da maneira apresentada na Figura 31. O processamento efetivo das mensagens ocorre na camada de Aplicação; na maioria dos casos, o tamanho da mensagem transmitida é superior ao tamanho do *frame* padrão da camada de Enlace, sendo necessária a fragmentação da mensagem – realizada pela camada de "Pseudo-Transporte" – antes de repassá-la à camada inferior.

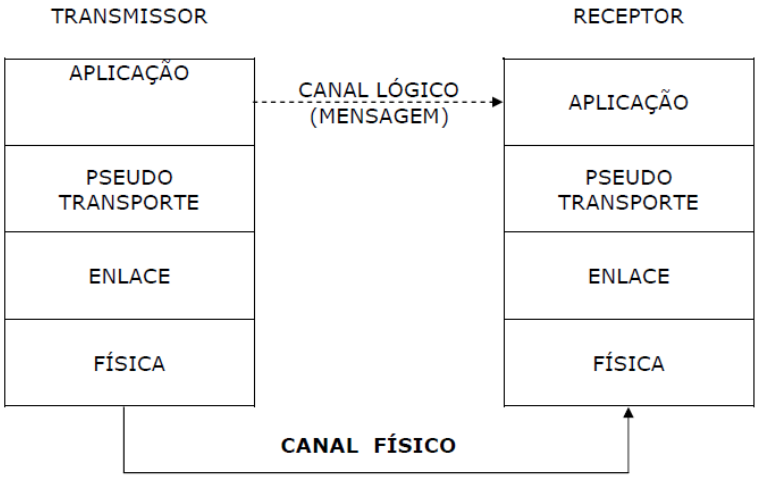

Figura 31 - Camadas do protocolo DNP3 - fonte: [4]

Em suma, a camada de Enlace tem a função de estabelecimento e controle do meio físico que conecta o dispositivo transmissor ao receptor, enquanto a camada Física é responsável, como o próprio nome diz, pela transmissão dos elementos básicos da mensagem (o *byte*).

Para facilitar o entendimento de quais são os passos realizados entre dois dispositivos que se comunicam em DNP3, a figura 32 apresenta um fluxograma genérico da sucessão de ações que podem ocorrer.

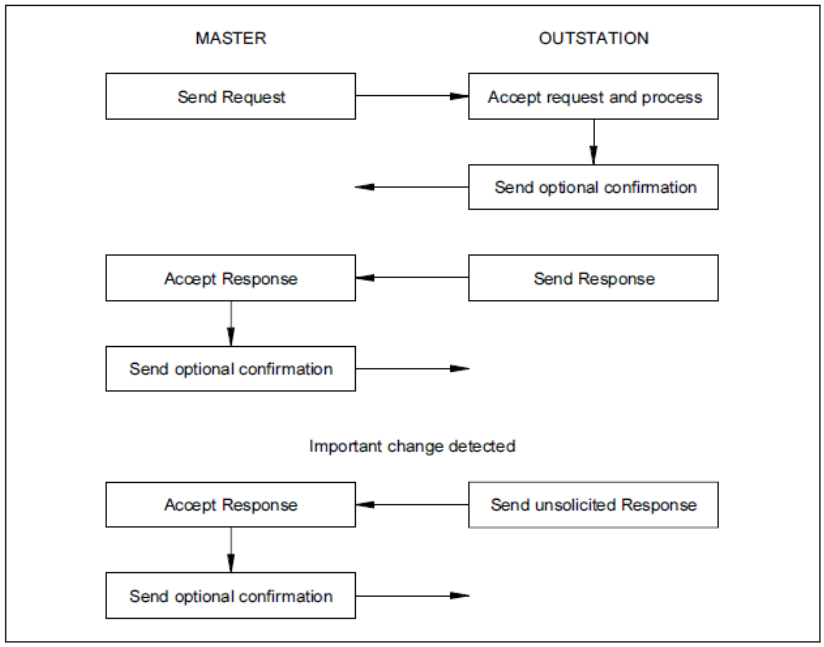

Figura 32 - Fluxograma de ações durante requisição - fonte: [6]

De acordo com a figura acima, o dispositivo mestre manda uma mensagem de requisição a partir da camada de aplicação que, ao ser interpretada pelo dispositivo destinatário, responde com uma mensagem de a partir de sua própria camada de aplicação. Será detalhada no Capítulo 6 uma modalidade de mensagens que surgem espontaneamente, ou seja, mensagens de resposta que chegam ao dispositivo mestre sem requisição; em tais situações, o mestre pode receber mensagens não solicitadas durante o processo de requisição de outra, mas o escravo só pode gerar uma mensagem espontânea quando tiver acabado de responder uma solicitação passada pelo dispositivo mestre.

Uma requisição partindo de um dispositivo mestre para um dispositivo escravo em particular só pode ser feita se todas as requisições anteriores tiverem tido um “desfecho” (resposta recebida, erro confirmado, por exemplo). Não é possível mandar uma requisição a um mesmo dispositivo escravo caso exista alguma transação “pendente”, ou seja, caso o dispositivo mestre esteja aguardando uma resposta.

4.2.1 – Definições de *frames* e divisão de mensagens

- Estrutura básica

As mensagens trocadas entre as camadas de Enlace são divididas em blocos de tamanho fixo: tais mensagens iniciam-se com blocos FT3 (representado na Figura 34 e como *block 0* na Figura 33), seguidos de blocos de dados opcionais que sempre são acompanhados por dois octetos de checagem de erros (blocos CRC).

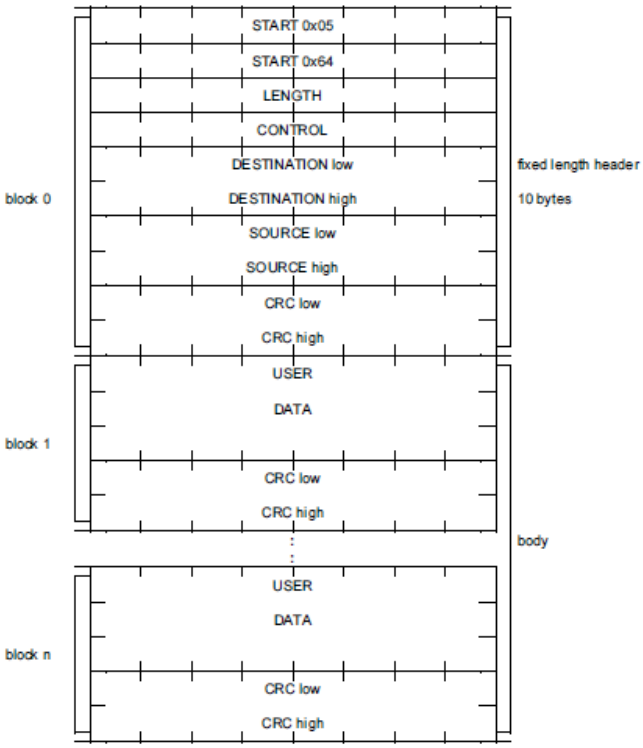

Figura 33 - Layout padrão das mensagens DNP3 - fonte: [6]

Os blocos cabeçalho possuem tamanho total de dez *bytes*, cinco destes denominados "*bytes* úteis": eles carregam as informações de destino e de origem da mensagem, conforme mostra a seguinte figura:

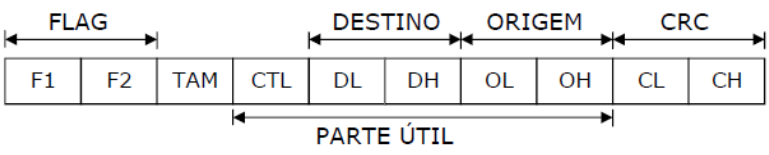

Figura 34 - Frame FT3 (frame de cabeçalho) - fonte: [4]

Os primeiros dois *bytes* são os *flags* que sinalizam que a mensagem foi baseada no protocolo DNP3, e possuem valores fixos “0x05” e “0x64”; o *byte* seguinte – TAM – carrega a informação do tamanho referente aos “*bytes* úteis” – neste caso, cinco *bytes*. Adentrando o bloco útil do *frame*, o *byte* CTL possui um código que pode ser visto na Figura 35. O MSB DIR carrega a informação da direção de transmissão da mensagem: quando 0, a mensagem vai da estação de controle para a remota; quando 1, o sentido inverso. O *bit* PRM mostra se a mensagem originou-se no dispositivo primário: se PRM=1, trata-se de uma mensagem de requisição; caso contrário, trata-se de uma resposta originária de um dispositivo remoto. O *bit* FCB – *Frame Count Bit*- vai alternando seu estado a cada transmissão de mensagem, mostrando que ela está sendo recebida/transmitida de fato. O *bit* seguinte, FCV – *Frame Count bit Valid* – mostra se o *bit* anterior é valido (FCV=1) ou se deve ser ignorado (FCV=0).

Os *bits* seguintes (representados na figura 35 como FUNC TION CO DE) carregam a informação do tipo da mensagem: esta informação depende do *bit* PRM, ou seja, os códigos possuem significados diferentes caso o dispositivo de origem da mensagem seja o primário ou o secundário. As Tabelas 4 e 5 mostram qual o significado para os dispositivos de controle e remotos, respectivamente.

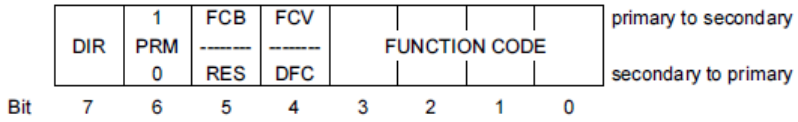

Figura 35 - Byte de controle da camada de Enlace CTL - fonte: [6]

FUNÇÃO	TIPO DA MENSAGEM	SERVICO	FCV
0	SEND – CONFIRM	RESET LINK	0
1	SEND – CONFIRM	RESET PROCESS	0
2	SEND – CONFIRM	TEST LINK	1
3	SEND – CONFIRM	USER DATA	1
4	SEND – NO REPLY	UNCONFIRMED USER DATA	0
5		Não Usado	-
6		Não Usado	-
7		Não Usado	-
8		Não Usado	-
9	REQUEST - RESPOND	REQUEST LINK STATUS	0
10		Não Usado	-
11		Não Usado	-
12		Não Usado	-
13		Não Usado	-
14		Não Usado	-
15		Não Usado	-

Tabela 4 - Function codes para dispositivos primários (controle) - fonte: [4]

FUNÇÃO	TIPO DA MENSAGEM	SERVICO
0	CONFIRM	ACK – reconhecimento positivo
1	CONFIRM	NACK – reconhecimento negativo
2		Não Usado
3		Não Usado
4		Não Usado
5		Não Usado
6		Não Usado
7		Não Usado
8		Não Usado
9		Não Usado
10		Não Usado
11	RESPOND	LINK STATUS (DFC = 0; DFC = 1)
12		Não Usado
13		Não Usado
14		Serviço não está funcionando
15		Serviço não implementado

Tabela 5 - Function codes para dispositivos secundário (remotos) - fonte: [4]

Continuando a análise do bloco útil, seguem dois *bytes* contendo o endereço de destino da mensagem (primeiro o *byte* menos significativo, seguido do mais significativo) e, logo em seguida, os dois *bytes* referentes ao endereço de origem da mensagem (também na ordem LSB e MSB); por último, seguem os dois *bytes* de checagem de erros (LSB do código cíclico seguido de seu MSB).

O endereçamento dos dispositivos pertencentes a uma rede DNP3 segue uma regra simples: dispositivos de controle são endereçados de 1 a 2000, enquanto dispositivos remotos (RTU's ou IED's, por exemplo) são endereçados de 2001 a 65534. O último endereço 65535 (0xFFFF) é reservado para mensagens do tipo *broadcast,* ou seja, mensagens com este endereço de destino são entregues a todos os dispositivos da rede.

As mensagens de tamanho variável possuem mais do que cinco *bytes* úteis, podendo ter no máximo 255 *bytes* úteis. A formação destas mensagens ocorre após a fragmentação ocorrer na camada de Pseudo-Transporte, seguindo para a camada de Enlace. Estas mensagens carregam informações referentes ao controle da camada de Enlace, ao controle da camada de Pseudo-Transporte e também dados referentes à camada de aplicação.

- Mensagens da Camada de Pseudo-Transporte

As mensagens que possuem tamanho superior ao *frame* FT3 e que são fragmentadas na camada de Pseudo-Transporte, originárias da camada de Aplicação, são denominadas *Application Data Units – APDU's.* O tamanho máximo desta mensagem é de 2048 *bytes*, mas sua transmissão pela camada de Enlace só pode se realizada se este bloco for separado em "fatias" de 249 *bytes*. A cada fatia desta é adicionado um *byte* denominado *Transport Header* – cabeçalho de transporte, ou simplesmente TH – formando assim o bloco *Transport Message* – Mensagem de Transporte. Desta forma, esta mensagem de transporte pode ter, no máximo, 250 *bytes*.

O *byte* TH possui o papel de organizar a fragmentação da mensagem originária da camada de Aplicação. Este *byte* possui a composição mostrada na Figura 36: o primeiro *bit* – FIN, de *final* –, quando setado, indica que a fatia de mensagem que o segue é a fatia final da mensagem original; quando é zero, indica que mais fatias ainda estão por vir. Já o segundo *bit* – FIR, de *first* – mostra que, quando setado, a fatia que o segue é a primeira fatia da mensagem original. Os *bits* representados na figura seguinte como SEQ, UEN e CIA (*bits* 3, 2 e 1) carregam o número da sequência dos fragmentos originários de uma mesma mensagem. Entre dois fragmentos contendo o *bit* FIR=1 e FIN=1, os *bits* da sequência devem ser consecutivos. A presença de códigos repetidos pertencentes à fragmentação de uma mesma mensagem significa que ocorreu uma retransmissão de informação.

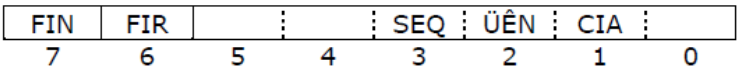

Figura 36 - Byte cabeçalho de transmissão (TH) - fonte: [4]

Em seguida, anexa-se ao início do bloco “Mensagem de Transporte + fatia” um bloco de mesma organização da mensagem de tamanho fixo vista anteriormente denominado *Data Link Header* – cabeçalho de dados de conexão. O resultado está representado na figura abaixo:

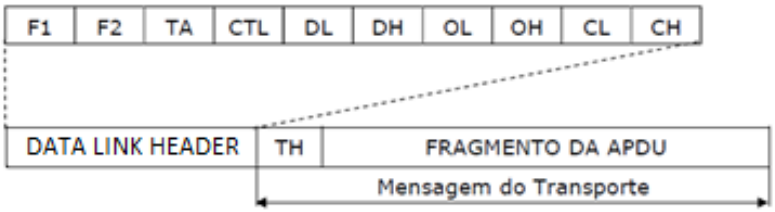

Figura 37 - Processo de fragmentação da ADU – fonte: [4]

Formado o bloco mostrado na figura acima, inicia-se o processo de inclusão dos *bytes* de checagem de erros, os blocos CRC. Inicialmente, incluem-se dois *bytes* CRC após o bloco de cabeçalho DLH; a partir deste bloco, a cada 16 *bytes* vindos da camada de Pseudo-Transporte, incluem-se mais 2 *bytes* CRC até a última fatia. Mesmo se esta última for menor de 16 *bytes*, fecha-se o bloco total da mensagem de tamanho variável com outros 2 *bytes* CRC. O resultado final pode ser visto na Figura 38:

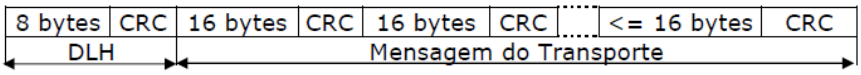

Figura 38 - Inclusão de bytes de checagem de erros na mensagem de transporte - fonte: [4]

- Mensagens da Camada de Aplicação

Analisando as mensagens geradas na camada de Aplicação (ou seja, antes da fragmentação que ocorre na camada de Pseudo-Transporte), verifica-se que elas podem ser formadas por diversas ADU's, seguidas bloco a bloco. Focando numa unidade ADU (antes desta unidade ser fragmentada), cada uma é composta pela soma de dois blocos: o primeiro bloco é denominado "Informação de Controle da Aplicação" – *Application Control Information,* ou apenas APCI. Como o próprio nome diz, este bloco contém informações de controle da mensagem. O segundo bloco é denominado Unidade de Dado de Serviço de Aplicação – *Application Service Data Unit, ASDU* – bloco que carrega informações que serão processadas pelo dispositivo receptor.

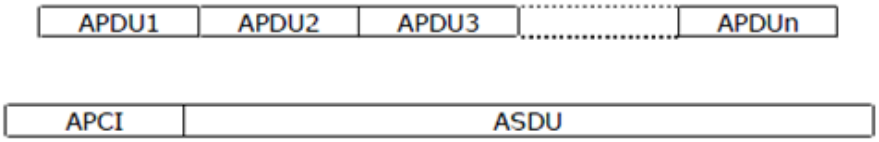

Figura 39 - Mensagens da camada de aplicação e suas subdivisões - fonte: [4]

O primeiro bloco constituinte do *Application Protocol Data Unit* – o APCI – pode assumir formatos diferentes dependendo do sentido de transmissão da mensagem: caso seja uma mensagem de requisição, o bloco APCI é formado por um *byte* denominado APC (*Application Control*) seguido de um *byte* denominado FUN (*Function)*; caso a mensagem seja de resposta, o bloco APCI é composto pelos mesmos dois *bytes* citados anteriormente seguidos de dois *bytes* denominados *Internal* e *Indication.* A montagem dos tipos do bloco APCI pode ser vista na Figura 40, e em seguida detalha-se cada elemento que compõe estas mensagens.

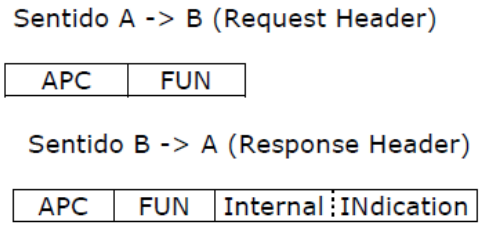

Figura 40 - Tipos de bloco APCI - fonte: [4]

O *byte* APC possui a composição semelhante ao bloco TH, conforme mostra a Figura 41. O *byte* APC também possui dois *bits* denominados FIR e FIN que sinalizam se o ADU que ele pertence é o primeiro ou o último ADU da mensagem de aplicação. O *bit* CON, quando setado, sinaliza que a aplicação que receber esta mensagem precisa devolver uma

confirmação de envio para o transmissor. Já o *bit* UNS, quando setado, mostra que a mensagem a qual está inserido não foi solicitada por nenhuma aplicação, mas que sua geração foi espontânea (situação que será vista na prática no Capitulo 6). Os *bits* seguintes – SE, QUEN, CI e A seguem o mesmo princípio dos *bits* do *byte* TH: eles apresentam o número da sequência em que a mensagem de aplicação foi fragmentada, e entre duas mensagens que carregam *bytes* APC com *bits* FIR e FIN setados, estes *bits* de sequência devem ser consecutivos.

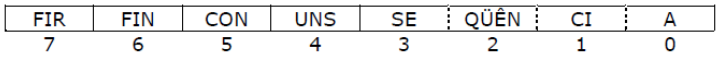

Figura 41 - composição do byte de controle de aplicação APC - fonte: [4]

O *byte* FUN, como o próprio nome indica, mostra qual a função daquela mensagem de aplicação. Novamente, os códigos de função possuem significados diferentes caso o dispositivo seja primário ou secundário. Caso o dispositivo seja de controle, existem códigos para aplicações de transferência de dados, telecomandos, congelamento de acumuladores, controle de aplicação, configuração, sincronização de tempo, por exemplo. Caso o dispositivo seja remoto, podem-se encontrar funções de confirmação, resposta ou de mensagem não solicitada.

Os dois últimos *bytes* do bloco APCI sentido “remoto -> controle” (blocos exclusivos deste sentido) – *Internal* e *Indication* – são obrigatórios ao final de todas as mensagens de resposta. O primeiro *byte* (representado na Figura 42) é composto pelos seguintes *bits*: *bit BRD*, que é setado quando um comando é recebido por *broadcast* e zerado na próxima transmissão a ser realizada pelo dispositivo mestre*; bits CL1*, *CL2* e *CL3*, que quando setados (de forma exclusiva entre eles) mostram a qual tipo de classe pertencem os dados disponíveis – o dispositivo requisitante deverá adequar sua próxima mensagem de requisição para a classe correspondente; *bit NTM*, que quando setado, indica a necessidade do dispositivo requisitante sincronizar o horário da mensagem (é zerado no momento de envio da informação de data/hora); *bit PML* , setado quando uma ou mais saídas estão em modo local e não estão passíveis de controle via rede DNP3 (fator extremamente importante em sistemas de energia, que exige total controle local principalmente em situações de reparos); *bit DVT*, que indica situação anormal do dispositivo remoto; *bit RST,* setado quando o dispositivo remoto tiver sido reiniciado.

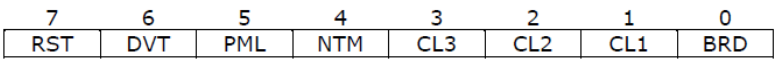

Figura 42 - composição do byte Internal *- fonte: [4]*

O *byte* final que compõe a mensagem APCI de resposta – *Indication* – representado na Figura 43 possui os seguintes *bits*: *bit FNI*, que quando setado, indica código de função não implementada no dispositivo remoto que a mensagem foi direcionada; *bit OUN*, em nível

lógico “1” quando o objeto solicitado não existe no dispositivo ao qual a mensagem foi direcionada (situação bastante utilizada em casos de *debugging)*; *bit BOV*, setado quando algum *buffer* possui seu limite máximo atingido; *bit OPE*, que indica que a requisição passada pelo dispositivo de controle já estava em execução antes do recebimento da mensagem; *bit CCR*, setado quando o objeto requisitado está corrompido, forçando o dispositivo mestre a atualizar este dado e a informar ao usuário sobre este fato (importante notar que, em muitos casos, objetos corrompidos podem reiniciar/desabilitar o dispositivo remoto, não sendo possível receber este “aviso prévio” de dado corrompido); *bits NUS* (seis e sete) não são utilizados (padrão igual à zero).

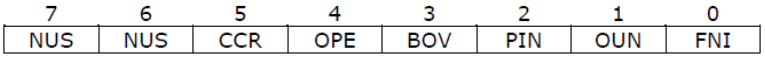

Figura 43 - Composição do byte Indication - fonte: [4]

O segundo bloco que compõe uma unidade de dados de aplicação – ADU – é o bloco ASDU – *Application Service Data Unit.* É neste bloco que toda a informação da aplicação de um dispositivo é passada para o outro dispositivo do link de comunicação. O bloco ASDU pode ser formado por vários blocos de informação, e cada um destes blocos é formado por um cabeçalho denominado DUI – *Data Unit Identifier* – seguido de um bloco de dados relacionados ao tipo de objeto especificado no cabeçalho. A Figura 44 mostra esquematicamente como fragmentar o bloco ASDU em suas partes integrantes:

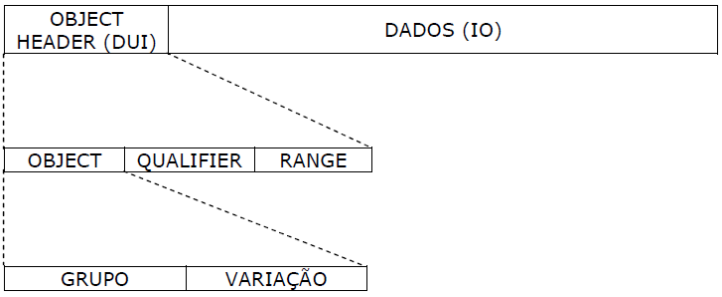

Figura 44 - Partes componentes do bloco ASDU - fonte: [4]

Analisando primeiramente o bloco cabeçalho de objeto DUI, verifica-se que este é composto por três outros blocos: *Object, Qualifier* e *Range.* O bloco *Object,* formado por dois *bytes*, carrega informações referentes ao grupo e à variação que pertence o dado daquela mensagem – e esta combinação de informações define unicamente o objeto.

O segundo bloco constituinte do cabeçalho DUI é o *byte* QUALIFIER – identificado na Figura 45 - que contém informações de como interpretar a sequência de dados. Este *byte* possui também informação de como interpretar e especificar o *byte* seguinte, RANGE; o código de qualificação carregado nos quatro *bits* menos significativos mostra ao dispositivo receptor que tipo de informação está sendo recebida, de acordo com a Tabela 6. Já os *bits*

4, 5 e 6 – constituintes do bloco *Index Size* – carregam informações de como interpretar o tamanho dos índices ou dos dados indexados no *byte* RANGE a cada objeto de acordo com a Tabela 7.

R	Index Size			Qualifier Code			
7	6	5	4	3	2	1	0

Figura 45 - Composição do byte QUALIFIER - fonte: [4]

0 (0x0) – índices START e STOP com 8 bits
1 (0x1) – índices START e STOP com 16 bits
2 (0x2) – índices START e STOP com 32 bits
3 (0x3) – endereços absolutos START e STOP com 8 bits
4 (0x4) – endereços absolutos START e STOP com 16 bits
5 (0x5) – endereços absolutos START e STOP com 32 bits
6 (0x6) – Campo RANGE não existe
7 (0x7) – quantidade de objetos com 8 bits
8 (0x8) – quantidade de objetos com 16 bits
9 (0x9) – quantidade de objetos com 32 bits
10 (0xA) – não usado

11 (0xB) – uso especial (vide significado do Index Size neste caso)
12 (0xC) – não usado
13 (0xD) – não usado
14 (0xE) – não usado
15 (0xF) – não usado

Tabela 6 - Códigos de qualificação de dados - fonte: [4]

0 – objetos estão compactados, sem índices antes deles
1 – objetos estão prefixados com índices de 1 byte
2 – objetos estão prefixados com índices de 2 bytes
3 – objetos estão prefixados com índices de 4 bytes
4 – objetos estão prefixados com tamanho do objeto de 1 byte
5 – objetos estão prefixados com tamanho do objeto de 2 bytes
6 – objetos estão prefixados com tamanho do objeto de 4 bytes
7 – reservado

Tabela 7 - Lista de códigos de indexação - fonte: [4]

Em resumo, partindo da mensagem gerada na camada de Aplicação (citada por último), a informação a ser transmitida vai sendo fragmentada e montada na ordem representada na Figura 46.

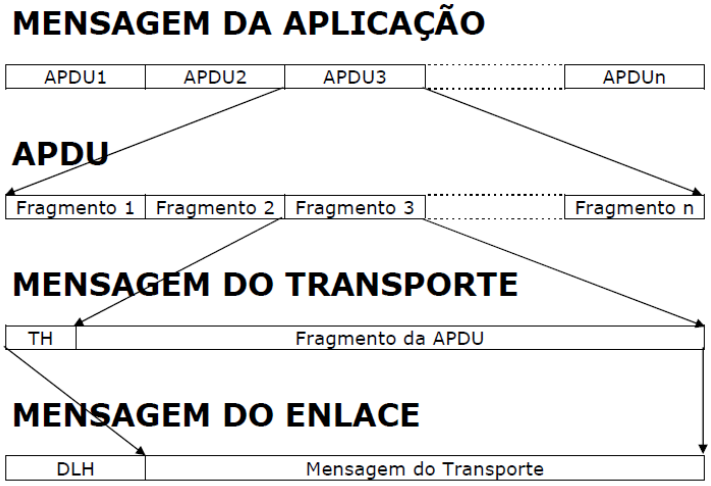

Figura 46 - Processo de fragmentação da mensagem - fonte: [4]

4.2.2 - Colisão entre mensagens

Em situações em que um dispositivo remoto gera uma mensagem não requisitada ao mesmo tempo em que o dispositivo de controle gera uma requisição direcionada justamente ao que acabou de gerar a mensagem espontânea, ocorre o caso em que ambos recebem mensagens de solicitações enquanto esperavam mensagens de confirmação. Nestas situações, a solução a ser tomada depende do tipo de requisição que foi imposta pelo dispositivo mestre.

Focando no dispositivo mestre, este sempre processará imediatamente a mensagem espontânea recebida, mesmo que esteja esperando uma resposta para uma requisição postada anteriormente. Se for o caso, a mensagem de confirmação de recebimento da mensagem espontânea também é transmitida imediatamente. Analisando agora o dispositivo escravo, ele geralmente responderá imediatamente a uma requisição recebida exceto quando a requisição for de leitura de dados: esta diferenciação é feita para evitar duplicação ou perda de dados. As Figuras 47 e 48 abaixo demostram os dois casos através de uma linha do tempo: na primeira imagem, a requisição de sequência 7 é respondida imediatamente pelo escravo, enquanto a confirmação da mensagem não solicitada (sequência 24) é enviada por último. A segunda imagem mostra que o dispositivo escravo “ignora” a requisição de sequência 2 e aguarda a resposta da resposta espontânea de sequência 18, respondendo à requisição apenas ao final do processo.

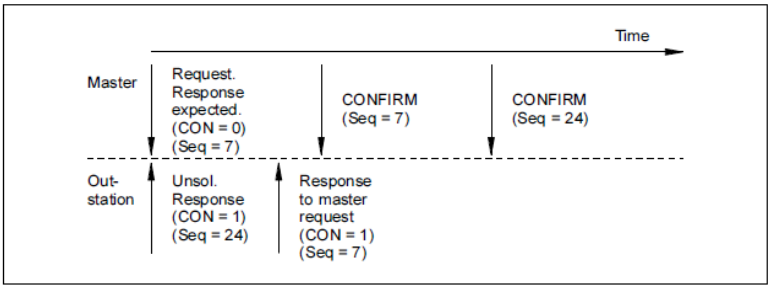

Figura 47 - Exemplo de resposta imediata à requisição - fonte: [6]

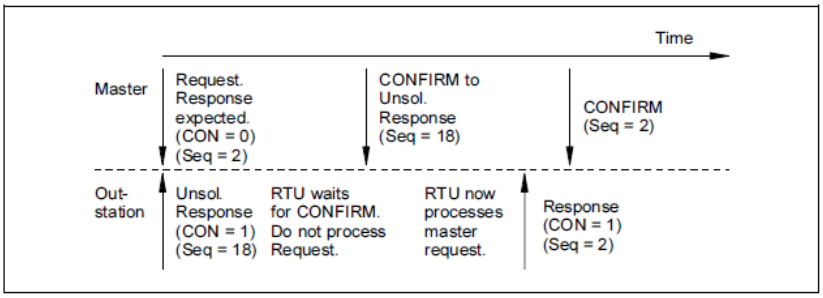

Figura 48 - Exemplo de priorização de resposta espontânea - fonte: [6]

4.2.3 – Níveis de implementação

Dependendo do tipo de aplicação em que o protocolo DNP3 está inserido, o usuário pode optar por diferentes níveis de implementação, sendo o nível três o mais detalhado subconjunto de aplicação possível e o mais indicado para ser utilizado em sistemas de energia, pois este possui características que são essenciais para esta área de aplicação, fato que tornaria a utilização dos níveis um e dois insuficientes para garantir a funcionalidade e segurança do sistema.

A implementação de nível três garante as seguintes funcionalidades aos dispositivos primários e secundários: leituras de objetos de classe de dados; leitura de diversos objetos específicos e suas variações; leitura de saídas binárias e analógicas; congelamento de acumuladores; leitura de contadores congelados; operações de controle para saída binárias e analógicas; escrita de data/hora; medição de *delays* de entrega; definição e redefinição de classes de objetos; habilitação ou desabilitação de respostas não solicitadas para classes de objetos.

Dentre as características citadas acima, é desejável em sistemas de energia a possibilidade de habilitar/desabilitar respostas não solicitadas durante janelas de manutenção: supondo a desenergização programada de um ramo alimentador de uma indústria, o sistema de controle pode informar aos dispositivos remotos de proteção de que a subtensão daquela área é esperada, evitando assim o possível acionamento acidental de uma proteção geral, parando toda a planta. Dessa forma, o sistema de controle consegue

desabilitar, durante um período programado, as mensagens não solicitadas destes dispositivos, evitando a desenergização de áreas desnecessárias.

O canal utilizado para este tipo de implementação são os do tipo *full-duplex*, visto que os dispositivos de remotos precisam ter sempre liberdade de envio de mensagens não solicitadas aos dispositivos de controle; tal fato não seria possível na implementação de nível um, que utiliza comunicação *half-duplex* (os dispositivos remotos não transmitem mensagens espontaneamente, mas apenas quando requisitados).

Capítulo 5 - Protocolo IEC 61850

Com a constante busca por melhorias em desempenho de comunicação de redes de energia, buscou-se a criação de protocolos que maximizassem a segurança da rede sem piorar o quesito desempenho, aliando um baixo custo de instalação (e manutenção) e de fácil entendimento de funcionamento e manutenção por parte do usuário. O padrão IEC61850 foi desenvolvido para equilibrar esta balança na área de automação, proteção e controle de subestações de energia, utilizando, para isso, tecnologia de ponta das áreas de processamento de dados e redes. A Figura 49 abaixo mostra que o protocolo pode ser empregado em todos os níveis do sistema, desde o supervisório até o nível de processo.

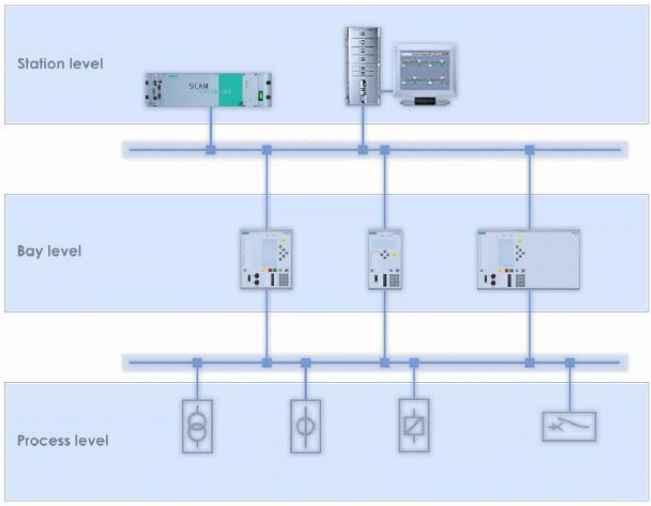

Figura 49 - Comunicação através de níveis com IEC61850 – fonte: [11]

Desde seu lançamento no ano de 2002, a utilização deste protocolo vem sendo ampliada mundialmente nos sistemas de controle e monitoramento de subestações de energia devido às características previamente estabelecidas pela IEC que buscavam justamente transpor as barreiras citadas. Foca-se neste capítulo a utilização do padrão IEC61850 em subestações baseadas em redes LAN de tecnologia *Ethernet*.

5.1 - Dados Históricos

Como já citado, um dos principais objetivos no desenvolvimento de novos protocolos a serem utilizados no controle e supervisão de subestações de energia era obter a possibilidade de trabalhar com equipamentos de diferentes marcas sem prejuízo de desempenho ou problemas de interface. Com este objetivo em mente, focou-se na criação

de um único protocolo a ser utilizado na subestação como um todo, em todos os níveis de comunicação, preocupando-se com a **vida útil** do protocolo, ou seja, a presença de certa flexibilidade na arquitetura que possibilitasse sua utilização não apenas em um curto período de tempo.

Esta pesquisa iniciou-se no ano de 1995 contando com um grupo de 60 pesquisadores ao redor do mundo, divididos em três grupos de trabalho. O padrão IEC61850 foi finalmente lançado em 2002, e sua utilização vem sendo ampliada no ramo de automação e controle de subestações de energia desde então, comprovando o sucesso do projeto.

Escolheu-se o padrão *Ethernet* como meio de comunicação simplesmente pela sua grande aceitação no mercado: desde o final da década de 70, seu uso vem sendo expandido ao redor do mundo e hoje se tornou uma ótima escolha para quem busca aliar baixo custo, desempenho e facilidade de instalação/manutenção. Destaca-se mais adiante que, apesar de se ter escolhido um meio de comunicação padrão para o protocolo, este não é exclusivamente dependente daquele; IEC 61850 consegue se modificar para acompanhar uma possível mudança na tendência de comunicação, pois está presente em todas as camadas do modelo OSI. Além disso, o cabeamento padronizado e já familiar (anteriormente apenas em ambientes corporativos) suporta a utilização de diferentes tipos de meios físicos de comunicação (fibra óptica, par trançado, coaxial, etc.) e atende às necessidades de blindagem eletromagnética regulada pela EMC na maioria dos ambientes industriais.

5.2 - Principais características

O padrão IEC61850 é dividido em quatorze partes, cada uma focando em um aspecto do sistema de automação de energia. O conjunto total oferece uma descrição completa do funcionamento da rede. Na Figura 50, apresenta-se a maneira com que a norma é dividida.

Aspectos do Sistema	
Parte 1:	Introdução e *Overview*
Parte 2:	Glossário
Parte 3: Parte 4:	Requisitos Gerais Gerenciamento de Projeto e Sistema
Parte 5:	Requisitos de Comunicação para Funções e *Device Models*
Configuração	
Parte 6:	Descrição da Configuração da Linguagem, relacionados aos IEDs, para Comunicação em Subestações
Testando	
Parte 10:	Teste de Conformidade

Modo dos Dados	
Parte 7-4:	Compatibilidade dos *Logical Node Classes* e *Data Classes*
Parte 7-3:	*Data Classes* comuns
Serviços de Comunicações Abstratos	
Parte 7-2:	*Abstract Communication Services (ACSI)*
Parte 7-1:	Princípios e Modelos
Mapeamento para a Rede de Comunicação real (SCSM)	
Parte 8-1:	Mapeamento para MMS and para ISO/IEC 8802-3
Parte 9-1:	*Sampled Values over Serial Unidirectional Multidrop Point-to-Point link*
Parte 9-2:	*Sampled values over ISO 8802-3*

Figura 50 - Divisão da norma IEC 61850. Fonte: [10]

Destaca-se novamente o fato de que o protocolo independe da transformação da tecnologia de comunicação – se no futuro for descoberto algum meio de comunicação mais vantajoso que a *Ethernet*, não será preciso abandonar o protocolo e sequer modificar as aplicações, os arranjos lógicos e as funções de comunicação já existentes no sistema: as alterações necessárias para que o sistema se adapte engloba apenas as camadas inferiores à camada de transporte do modelo OSI, pois esta isola alterações ocorridas nos meios de comunicação da estrutura lógica de dados presente nas camadas superiores. IEC61850 está fortemente ligado ao modelo de dados de objetos: isso significa que a norma é baseada nos componentes comuns de subestações (disjuntores, controladores, por exemplo). É este aspecto que garante a **longevidade** do protocolo; o rápido avanço tecnológico existente na parte física não se torna um problema para a estruturação lógica dos dispositivos que compõe a rede de proteção.

5.2.1 - Estrutura de Dados

Neste ponto, deve-se deixar claro de que forma o padrão IEC61850 torna possível a **interoperabilidade** entre diferentes marcas do mercado: a padronização não é feita sobre funções ou sobre algoritmos (cada empresa possui a liberdade de escolher como os dados são manipulados dentro de seus equipamentos e normalmente tais informações são extremamente sigilosas); o protocolo atua na quebra das funções em partes cada vez menores, partes denominadas "*function units".* A partir deste ponto, são feitos padrões de interfaceamento entre as *function units*, e estas interfaces serão realizadas entre dados de entrada, dados de saída e dados particulares de determinada aplicação.

Ao conjunto composto pelas *function units* e suas interfaces denominou-se "*Logical Node"* – nós lógicos*.* A Figura 51 mostra um exemplo destes *Logical Nodes:* funções primárias podem ser agrupadas com informações de entrada/saída de dados para formar estes nós lógicos, como: PDIS – proteção por distância (*distance*); XCBR – disjuntor (*circuit breaker*); CILO – intertravamento (*interlocking*). Existem mais de noventa nós lógicos definidos no padrão IEC61850, criando uma boa base de componentes para descrever situações reais.

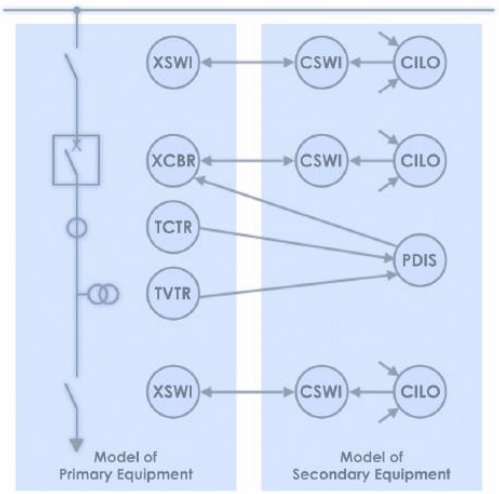

Figura 51 - Logical Nodes – *fonte: [11]*

Nesta figura, observa-se que cada equipamento é representado por um nó lógico (de cima para baixo temos: uma chave seccionadora, um disjuntor, um transformador de corrente, um transformador de potência e outra chave seccionadora); estes são os modelos primários de equipamentos. Já na parte de modelos secundários, é possível ver que os nós de intertravamento (CILO) recebem informações e, caso algum estado parametrizado seja ativado, envia uma mensagem para o nó CSWI ativando o intertravamento; desta forma, estes podem enviar comandos de chaveamento para os modelos primários (as seccionadoras e o disjuntor). Os transformadores de tensão e corrente enviam informações ao nó de proteção de distância, que pode eventualmente enviar o comando de *trigger* ao disjuntor em situações de falha.

Os dados do padrão IEC61850 são modelados orientados a objeto e possuem estampa de validade; o padrão agrupa nós lógicos de acordo com seus aspectos funcionais e sob dispositivos lógicos (*logical devices) – LD's.* Dessa forma, os nós lógicos dos equipamentos primários são relacionados às interfaces de I/O. No caso dos relés digitais de proteção, estes são mapeados como se fossem diversos dispositivos lógicos agrupados em um mesmo IED.

O objetivo de padronizar a comunicação entre os IED's fez com que o padrão exigisse um arquivo capaz de inserir um dispositivo em um sistema de comunicação; este arquivo, denominado "*IED Configuration Description"* – ou ICD – é descrito por uma linguagem de configuração da subestação baseada em XML, denominado SCL (*Substation Configuration Language)*. Neste arquivo determinam-se informações que estabelecem a alocação de diferentes nós lógicos em seus respectivos IED's, o tipo de canal de comunicação e funcionalidades específicas de cada equipamento. A linguagem SCL representa, portanto, uma importante ferramenta na geração da base de dados das diversas camadas da rede. Pelo fato dos dados serem gerados de forma automática (estruturas de dados bem definidas em nós lógicos), reduz-se o tempo de *setup* da rede e a probabilidade de surgirem erros em processes de conversão e endereçamentos especiais. O arquivo SCL descreve por completo a lógica instalada em uma rede de energia com linguagem de alto nível configurável, sendo uma característica bastante desejável no aspecto de manutenção de redes.

Outra característica que vem sendo colocada em destaque na implementação de novos sistemas de controle e monitoramento é o tempo de aviso de determinado evento pré-estabelecido para um alvo que possui interesse em receber *logs* de tais eventos; IEC61850 possui diversos mecanismos que asseguram o recebimento por parte do cliente interessado nestes *logs*, aumentando a confiabilidade e diminuindo o tempo de resposta de uma possível ação corretiva, por exemplo. Além disso, o padrão possibilita a criação de um rico banco de dados contendo informações previamente escolhidas pelo cliente (eventos de possíveis falhas no sistema, anomalia de alguma variável monitorada, ou até mesmo dados esporádicos para efeitos estatísticos). O fato de utilizar apenas um canal de comunicação para a transmissão dos dados – em tempo real, de forma sincronizada via *Ethernet* – torna o sistema mais confiável e reduz os custos de instalação.

O IEC61850 possui o diferencial de definir elementos do sistema em análise de uma forma clara utilizando linguagem de fácil compreensão dentro do contexto de subestações de energia. A característica que realmente destaca este padrão dos demais é a seguinte: são criados modelos que carregam informações do sistema, como: requisitos de

comunicação, características funcionais, como dispor a estrutura de dados de um dispositivo, convenções acerca dos nomes utilizados para representar os dados, características funcionais, estrutura de dados utilizados pelos dispositivos. Estes modelos são mapeados para um tipo específico de perfil de protocolo IEC61850, que é otimizado para sua área de atuação dentro do sistema de energia, como áreas de sistema supervisório e de controle, sistema de acesso de dados ou interfaceamento dos transdutores. A determinação destes modelos de forma separada ao protocolo é a base da longevidade do padrão IEC61850, pois torna-o flexível para ser utilizado em tecnologias de rede futuras sem alterar o modelo já existente.

5.2.2 – *Mensagens GOOSE*

Foi criado no IEC61850 um modelo de controle denominado *Generic Substation Events* (sigla GSE). Este modelo foi criado com o intuito de viabilizar a transmissão horizontal de dados através de toda a rede da subestação de forma rápida e confiável, sempre atentando em não desequilibrar a balança desempenho *versus* confiabilidade. Este modelo de controle faz com que a informação referente a um evento seja transmitida a todos os componentes da rede em questão (transmissão *broadcast,* pela qual a mensagem é enviada para a camada mais inferior do modelo OSI*).* Ainda dentro deste modelo de controle, criaram-se mais duas subdivisões: *Generic Object Oriented Substation Event (GOOSE)* e *Generic Substation State Event (GSSE).*

No modelo GSSE, apenas informações de *status* podem ser transmitidas pela rede, e ao invés de utilizar um conjunto de dados na transmissão como no modelo GOOSE, utiliza-se uma lista de *status* (uma *string* de *bits*). Apesar deste modelo apresenta maior rapidez de transmissão por apresentar um formato mais simples de dados, seu uso está sendo posto de lado pelo crescimento da utilização do modelo GOOSE.

Mensagens transmitidas por serviços que não são de tempo crítico (aplicações de controle, e não de proteção), transitam por todas as camadas do modelo OSI. Possuem muito mais ferramentas de validação, checagem de erros e serviços de segurança do que as mensagens GOOSE, mas não atendem no quesito velocidade - fator primordial em situações de proteção. Por esse motivo, mensagens via GOOSE são transmitidas em conjunto com as mensagens a nível vertical diretamente nas duas primeiras camadas do modelo OSI, obtendo tempo de transmissão muito mais curto do que as mensagens do tipo cliente/servidor. A imagem 52 mostra os diferentes caminhos pelos quais passam mensagens de nível vertical (não urgentes) e mensagens GOOSE (de prioridade) durante a comunicação de dois dispositivos via IEC61850 (IED_A e IED_B):

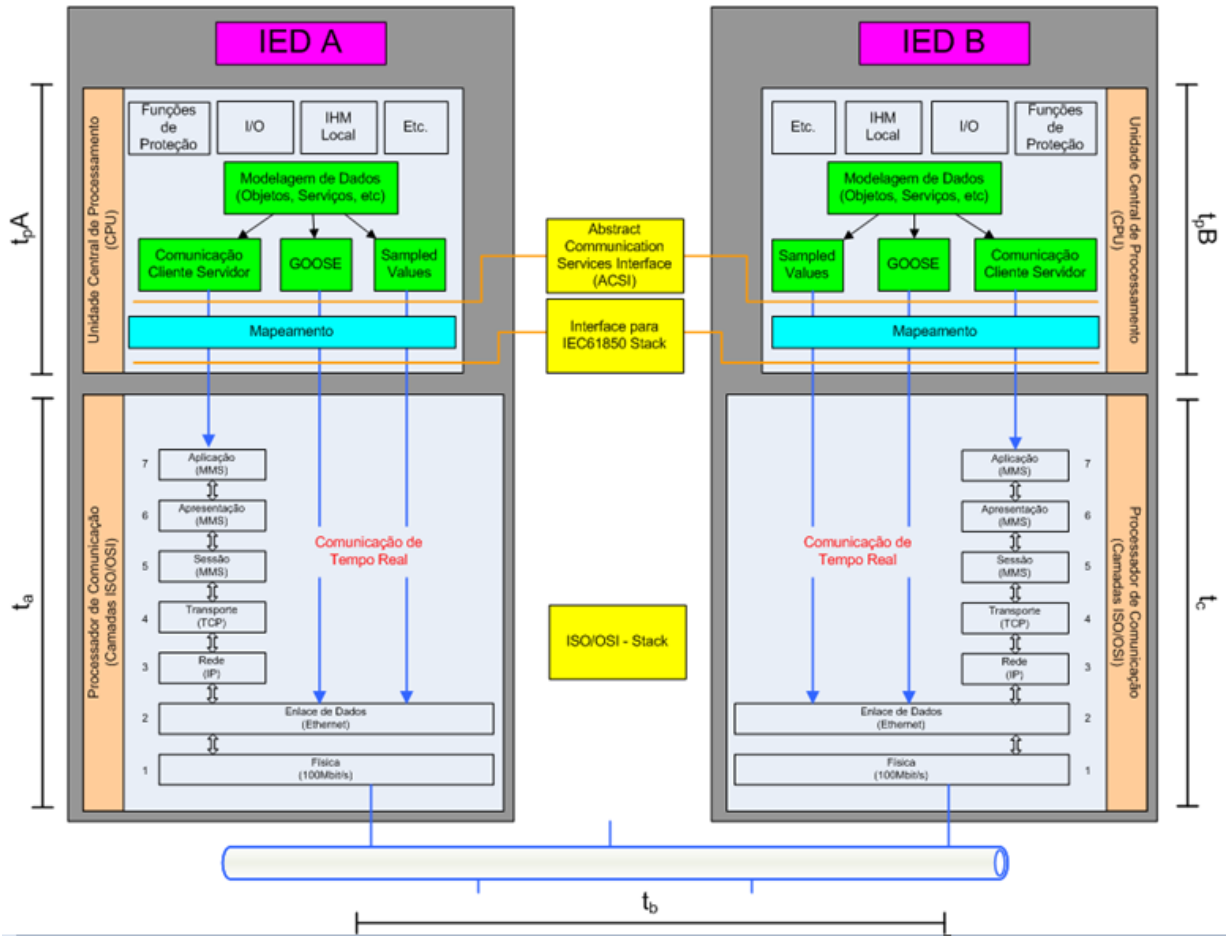

Figura 52 - Comparação tempo de transmissão: GOOSE x cliente/servidor - fonte: [13]

Neste ponto, vale destacar que entende-se como tempo de transmissão da mensagem o intervalo desde o momento que o dispositivo transmissor coloca os dados no *stack* do modelo OSI (camada de aplicação) até a chegada na última camada do modelo OSI do dispositivo destinatário (também a camada de aplicação). Os tempos de processamento dos dados, que podem variar de acordo com o tipo de proteção atuante no dispositivo (TpA e TpB na imagem acima), não participam da contagem do tempo de transmissão que é formado por $t_a + t_b + t_c$ (vide imagem acima).

Nas mensagens do tipo GOOSE, informações como bloqueio, posição e *trips* de disjuntores são transmitidas de forma rápida e confiável, em um tempo máximo de quatro milisegundos. IEC61850 utiliza a rede *Ethernet* para transmitir horizontalmente, entre IED's, este tipo de informação crítica para todos os dispositivos da rede, utilizando apenas as duas camadas mais inferiores do modelo OSI (Física e de Enlace). A mensagem é "capturada" apenas pelos dispositivos interessados na mensagem, aumentando assim a velocidade de transmissão dos dados.

A Figura 53 mostra um exemplo de ligação entre diversos IED's que se comunicam horizontalmente via mensagens GOOSE através de uma rede IEC61850. Verticalmente, os dispositivos IEDs comunicam-se com a estação de controle (que possui uma interface IHM) e com um *gateway* para acesso de outras redes, possibilitando um controle remoto via Internet, por exemplo.

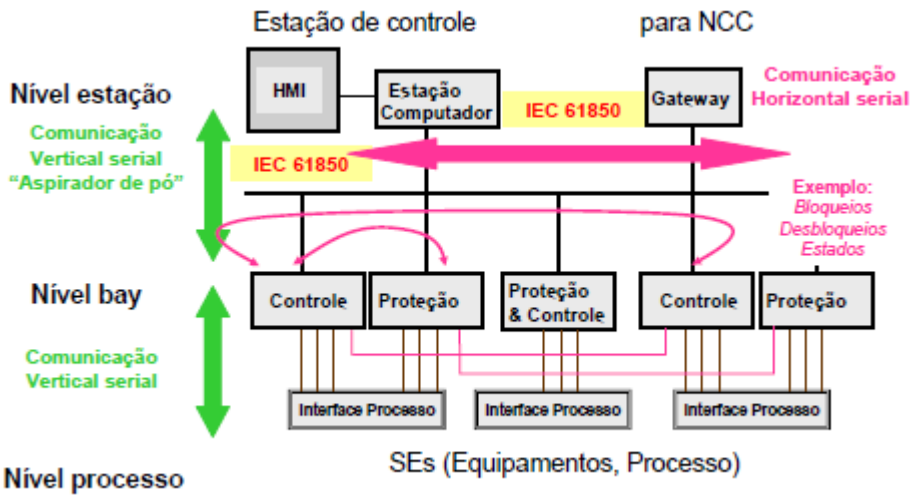

Figura 53 - Representação de uma rede de controle - fonte: [9]

Destaca-se neste ponto a possibilidade do sistema se comunicar tanto como uma rede *peer-to-peer* (comunicação ponto a ponto entre IED's em um mesmo nível), como numa relação *cliente-server* (comunicação entre dispositivos em diferentes níveis em que o server fornece toda informação solicitada pelo cliente). Pode parecer um tanto ousado substituir a clássica solução de sinalização de *status* via cabos portando sinais elétricos por sinalizações via mensagens GOOSE: em caso de perda do barramento de comunicação horizontal dos relés, os dispositivos não poderão se comunicar e as lógicas de intertravamento serão prejudicadas. Entretanto, a proteção individual de cada relé não é afetada, e as vantagens superam a pequena possibilidade de uma situação extrema de perda total do meio físico de comunicação horizontal. Além disso, em caso de perda do sistema supervisório (comunicação vertical), as mensagens GOOSE continuam a monitorar os *status* críticos a nível horizontal. A Tabela 8 mostra um resumo das vantagens (em verde) e desvantagens (em vermelho) destas possíveis soluções.

Modo tradicional – Fiação	IEC 61850 com GOOSE
Precisa de N*(N-1)/2 ligações para N relés	Relés compartilham uma rede em comum
Reprogramação pode precisar de alterações na fiação	O número de ligações para N relés é N
Não se sabe se as ligações estão funcionando até usá-las	Relés enviam seus *status* para todos os outros, de uma vez, usando GOOSE
Não é necessária mão de obra especializada	*Status* sendo trocados continuamente
Solução aceita em qualquer mercado	Redução de entradas binárias e fios
ciclo de vida ilimitado -> fio será sempre fio	Flexibilidade na programação é independente da fiação
	Confiabilidade: *status* das ligações são conhecidas antes do seu uso
	Mais performance com mais dados
	Alto investimento em componentes de rede
	É necessária mão de obra especializada

Tabela 8 - Comparação: GOOSE x cabeamento clássico – fonte: [10]

Focando a troca de informação entre os IED's, IEC61850 divide todos os dados que fluem através de um nó lógico em objetos bem definidos. A Figura 54 mostra um exemplo de como representar um disjuntor em um sistema: o nó lógico "XCBR" (*circuit breaker*) possui diversos objetos de dados atrelados ao nó.

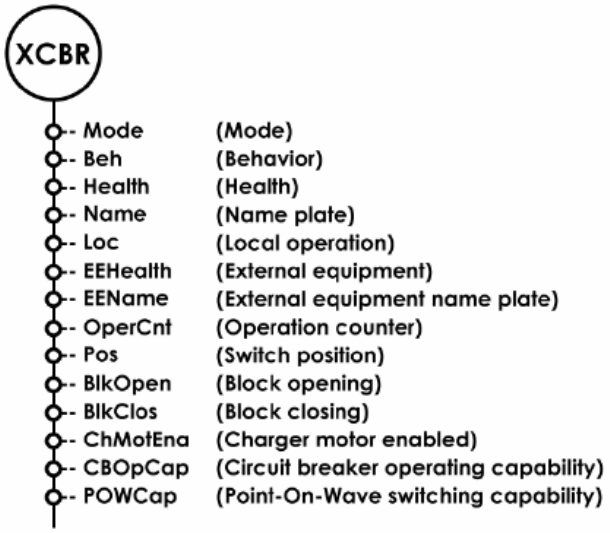

Figura 54 - Nó lógico e deus objetos de dados – Fonte: [11]

O padrão IEC61850 dita a seguinte sintaxe para localizar um objeto de dado: "*Logical Device (LD)/Logical Node (LN)/Data Object (DO)".* Desta forma, para que o sistema consiga

localizar a informação da posição deste disjuntor (que está parametrizado como um dispositivo de controle), por exemplo, deve-se seguir o seguinte caminho: "*CTRL/XCBR/POS".* Acessando o *data object POS,* pode-se solicitar a mudança de posição do disjuntor alterando a informação atrelada a este objeto – *data atribute.* Em resumo: todo nó lógico (LN) possui objetos de dados atrelados (DO), que possuem informações atreladas (DA), que podem alterar o estado do dispositivo. Será possível visualizar a transmissão de comandos a um disjuntor virtual no Capítulo 6.

5.2.3 - Interface física entre o IED e o meio de comunicação

Pela necessidade de tornar o protocolo IEC61850 um padrão com alta longevidade, a adaptação de cada componente da rede a diferentes meios de comunicação (*Ethernet*, fibra óptica, cabo coaxial, etc.) tornou-se um requisito. Esta interface é feita através de módulos de comunicação que são conectados aos IED's, cada módulo referente a um tipo físico de comunicação. A figura abaixo apresenta um módulo *Ethernet*, responsável por conectar o dispositivo ao sistema por diferentes possibilidades de conexão, como anel, estrela, com uso de um dispositivo *switch*, etc.

Figura 55 - Módulo de Comunicação Ethernet - Fonte: [11]

5.3 - Estudo de casos: comunicação via GOOSE

8 Situação de intertravamento entre três vãos

Nesta parte do capítulo, realiza-se uma análise teórica da disposição e funcionamento da rede de comunicação que interliga relés de proteção digitais, designados a realizar a proteção e monitoramento de diversas secções de subestações. Estes IED's são unidades multifuncionais que podem atuar nas esferas de proteção, medição de sensores, controle e monitoramento. São programados por *software*s específicos para realizar lógicas de intertravamento.

O padrão IEC61850 permite que estes dispositivos microprocessados possam ser controlados remotamente, visto que cada IED possui um endereço IP que possibilita a comunicação através da rede *Ethernet*. Destaca-se o aumento na capacidade de

processamento de dados destes dispositivos, alcançando níveis muito satisfatórios de desempenho e possibilidades de multi-funções em um mesmo dispositivo.

A utilização de mensagens GOOSE possibilita criar um sistema de intertravamento independente da comunicação entre o servidor e o cliente (o relé e a estação central de controle, respectivamente). Este sistema de distribuição de mensagens GOOSE pode ser responsável pela segurança de diversas áreas da subestação, como na proteção de autotransformadores, barras de distribuição, linhas de transmissão, chaves seccionadoras, etc.

Para exemplificar o procedimento necessário para criar um sistema de intertravamento, utiliza-se a simples representação de uma subestação de energia da Figura 56. Nela, observa-se que a alimentação do sistema é feita por duas barras interligadas pelo vão C02 e que o sistema é seccionado em três vãos:

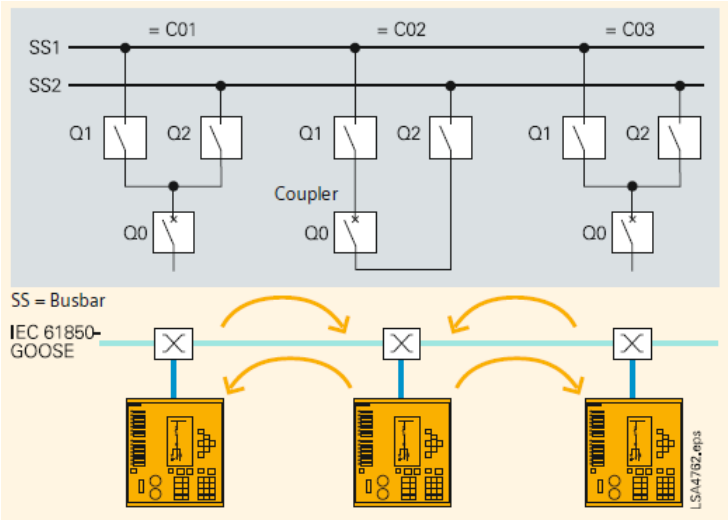

Figura 56 - Representação de uma subestação – fonte: [8]

Para que seja possível criar uma lógica de intertravamento neste sistema, deve haver troca de informações entre os ramos alimentadores (C01 e C03) e o ramo acoplador (C02). O vão acoplador deve informar aos vãos alimentadores se seu *status* é *aberto* ou *fechado*. Caso *fechado,* as seccionadoras dos vãos alimentadores podem ser operadas (até mesmo se os disjuntores destes vãos estiverem fechados), já que as duas barras estarão sob o mesmo potencial.

Por outro lado, é necessário que os vão alimentadores informem ao vão acoplador se as barras alimentadoras estão conectadas a partir de chaves seccionadoras de pelo menos um dos vãos alimentadores (Q1 e Q2): em caso positivo, o vão acoplador não poderá ser aberto, pois caso isto acontecesse, não seria mais possível manobrar as chaves seccionadoras do(s) vão(s) alimentador(es) devido ao risco de criação de arcos de corrente.

Esta simples lógica de intertravamento, que cria estados não permitidos para certas combinações de *status* individuais dos aparelhos, é realizada conectando os relés de

proteção a um único canal de comunicação *Ethernet*, que poderá servir de meio de tráfego tanto para mensagens GOOSE quanto para diversas mensagens de outros *frames* e serviços. A possibilidade de tráfego de mensagens de diferentes *frames* e serviços tornou-se um diferencial importantíssimo deste protocolo, sendo um dos propulsores do padrão na implementação de novas redes de energia.

9 Monitorando a rede

Após a instalação da rede e do início de seu funcionamento, é imprescindível realizar um constante monitoramento dos meios físicos de comunicação e dos dispositivos, gerando avisos no sistema de supervisão caso seja detectada alguma variável fora de valor esperado ou então alguma falha que desconecte alguma parte da rede.

Este monitoramento é realizado em dois pontos do sistema: um deles é cada canal de comunicação *Ethernet*, inspecionando se a conexão está de fato ocorrendo com o *switch* daquela linha. Esta característica possibilita que o sistema procure conexões secundárias em caso de falhas na conexão primária, criando um sistema autocorretivo e diminuindo as chances de falhas graves no sistema. O outro ponto de monitoramento é a avaliação constante do *status* dos dispositivos: a Figura 57 mostra uma possível falha de comunicação entre os relés de proteção que monitoram os vão C01 e C02 do exemplo em estudo, fato que desencadearia a emissão de mensagens GOOSE a nível horizontal.

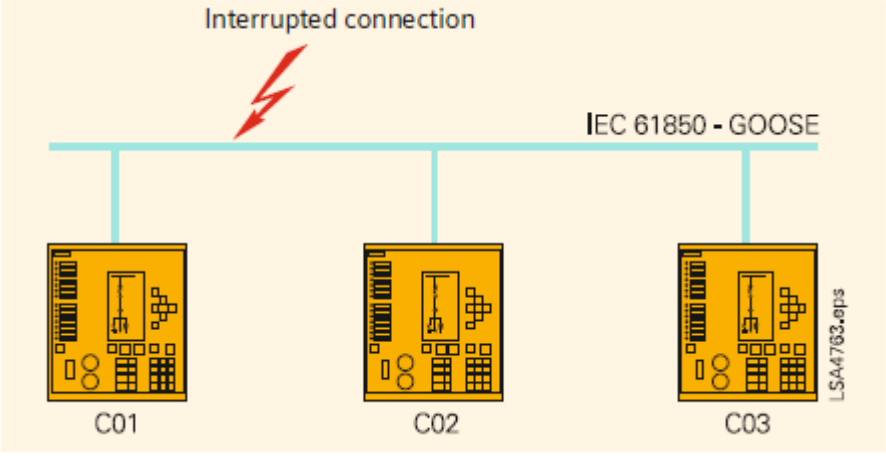

Figura 57 - Possível falha de comunicação entre relés dos vãos C01 e C02 – fonte: [8]

De acordo com a figura acima, se a comunicação entre os relés de proteção que monitoram os vãos C01 e C02 for perdida, as informações do *status* do vão C02 no vão C01 será inválida, enquanto a informação do *status* das chaves seccionadoras do vão C01 no vão C02 também será inválida. Um *bit* de falha de comunicação é setado e a informação de falha de comunicação do canal é mostrada na interface com o usuário dos dispositivos de proteção (pode ser tanto em um supervisório como em um IHM). Nota-se que os dispositivos dos vão C02 e C03 não são afetados pela falha, continuando a comunicação entre eles.

Nesta situação, é papel do sistema de monitoramento fazer com que todas as lógicas de intertravamento que dependem dos *status* que se tornaram inválidos por causa da falha sejam colocadas em um estado seguro e bloqueado. Em resumo, a utilização deste padrão de comunicação possui a vantagem de não depender de um sistema de controle central (a falha em determinado ponto da rede não prejudica o sistema como um todo). A troca de mensagens GOOSE a nível horizontal continua a informar o *status* configurado de cada dispositivo mesmo que ocorra um problema e o sistema supervisório seja perdido.

10 Situação de intertravamento reverso utilizando mensagens GOOSE

A situação descrita neste tópico representa uma boa alternativa financeira para a proteção de barras de distribuição em subestações. Para o funcionamento desta lógica de intertravamento, utiliza-se a proteção de sobrecorrente nos relés de proteção digital.

A figura seguinte representa a rede de distribuição que se deseja proteger contra situações de sobrecorrente. Situações deste tipo podem ocorrer por diferentes motivos, como falha de isolação, queda de condutores nas linhas, ruptura de cabos, etc. O sistema é alimentado por um vão alimentador contendo um transformador abaixador de tensão, que é monitorado (via TC e TP) por um relé de proteção. Este vão alimentador é ligado a uma barra de distribuição que alimenta outros três vãos alimentadores, cada um deles sendo monitorado individualmente por um relé de proteção (também via TC e TP).

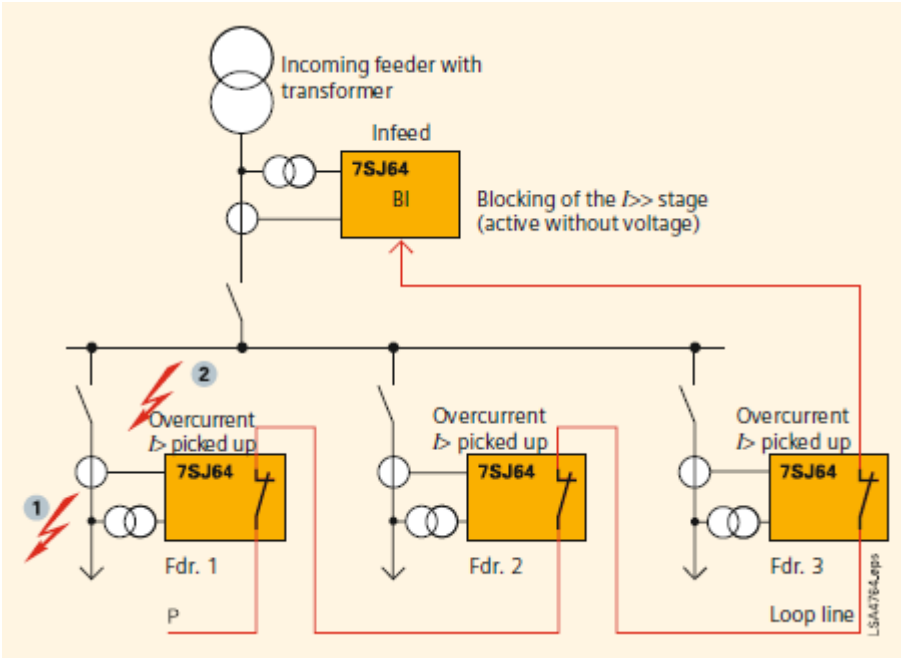

Figura 58 - Sistema de distribuição por barra única – fonte: [8]

Nesta situação, o protocolo IEC61850 utiliza um nó lógico denominado PTOC (do inglês, *Protection Time Overcurrent)* para transmitir uma ação emergencial quando

detectado um valor excessivo de corrente durante um tempo “T” determinado previamente pelo usuário. Caso a corrente monitorada em algum dos TC’s dos vãos alimentadores supere o valor estabelecido pelo usuário, o relé de proteção passa para o estado de *pick-up*, isto é, é sinalizado que a proteção de emergência que monitora sobrecorrente está acionada; a partir deste ponto, o relé aguarda um período fixado de “X” segundos (também parametrizado pelo usuário) que, se superado, ativa o estado de *trip* do disjuntor, ou seja, aciona a saída que abre o disjuntor de proteção conectado àquele relé.

Neste caso, quando ocorre alguma falha de sobrecorrente nos vãos alimentadores (como no ponto 1 da figura acima), o sistema precisa desativar a proteção de sobrecorrente do disjuntor do vão alimentador principal (principalmente se este relé estiver programado para abrir o disjuntor em um tempo menor do que o relé dos vão alimentadores); dessa forma, caso a falha não seja solucionada, apenas o vão alimentador é desativado, mas o restante da planta continua energizado.

Esta comunicação que bloqueia o *trip* do relé de proteção do transformador do vão principal é feita através da entrada binária deste relé: ao acusar a falha de sobrecorrente no ponto 1, o relé do vão alimentador 1 (*Fdr.1)* abre um contato NA que acaba desenergizando a entrada binária do relé do vão principal (*Infeed).* Dessa forma, o relé toma conhecimento de que ocorreu uma falha em um dos vãos alimentadores e que a falha já está sendo vista pelo relé daquele vão, sendo desnecessário ativar o disjuntor do vão alimentador, que acarretaria no desligamento da planta inteira.

Numa outra situação, a falha pode ocorrer em um ponto da barra de distribuição (vide ponto 2 da figura acima): neste caso, não será a proteção dos relés dos vãos alimentadores que será ativada, mas sim a do relé do vão alimentador principal. A proteção do disjuntor principal é colocada em estado de *pick-up,* e caso a falha persista além de um limite de T segundos parametrizada pelo usuário, o estado de *trip* do disjuntor principal é acionado acarretando em sua abertura e no consequente desligamento de toda a planta.

Em ambos os casos descritos acima, o sistema utiliza mensagens tipo GOOSE para informar a todos os IED’s da rede qual o *status* atualizado do sistema: a mensagem contendo a informação de que o estado de *pick-up* não está ativado é enviada de forma cíclica a cada 0.5 segundos (a mensagem em si demora microssegundos para ser passada). A Figura 59 mostra a representação no tempo de como este ciclo é formado:

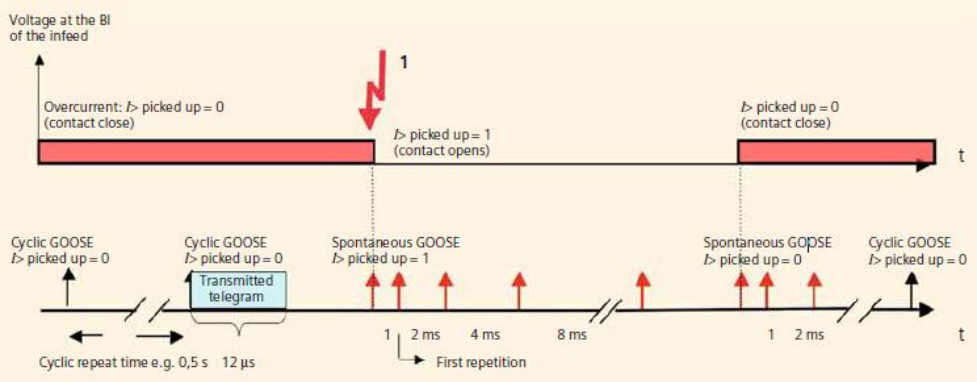

Figura 59 - Ciclo de mensagens GOOSE em sistemas de proteção – fonte: [8]

Tais mensagens contendo o *status* devem ser programadas como de alta prioridade, e o tempo entre cada envio pode ser personalizado pelo usuário; todo IED que está inscrito na lista de *target* recebe a mensagem de todos os IED's. Será possível visualizar o tráfego destas mensagens numa rede experimental no Capítulo 6.

Supondo o primeiro caso, assim que algum relé de proteção pertencente a algum vão alimentador (*Fdr.1* a *Fdr.3)* entra na situação de *pick-up,* o dispositivo dispara de forma espontânea a mensagem contendo o *status pick-up=1;* tal mensagem é repetida em 1ms, 2ms, 4ms e assim por diante. Esta mensagem é transmitida até que o *status* de *pick-up* seja desativado (a falha foi resolvida) ou então até que a mensagem de *trip* acionado passe a ser transmitida. Este mecanismo de distribuição de mensagens de *status* horizontalmente em intervalos exponencialmente crescentes assemelha-se ao CSMA-CD visto anteriormente, e garante matematicamente que a mensagem chegará ao destinatário de interesse.

Capítulo 6 - Testes Práticos

Baseando-se no estudo dos capítulos anteriores, é possível verificar a grande variedade de opções disponíveis para realizar a comunicação entre dispositivos de uma rede. O estudo foi iniciado pelo protocolo Modbus justamente pelo fato de este ser o que apresenta estrutura lógica mais simples se comparado com os outros dois. Entretanto, esta simplicidade começa a ser deixada para trás quando se opta pelo tráfego de mensagens deste tipo em redes *Ethernet*, requisitando o processo de encapsulamento descrito no Capítulo 3.

Quando a situação prática requer uma comunicação entre dispositivos de forma rápida, com poucas informações e serviços, o protocolo Modbus via *serial* atende bem ao propósito: apresenta-se a seguir um exemplo desta situação. Entretanto, a maioria das redes que vêm sendo implementadas no mercado já devem estar preparadas para alcançar uma maior diversidade de serviços trafegando no mesmo meio de comunicação, fato que re*string*e a implementação de redes de comunicação somente com meios físicos *serial.* A grande aceitação dos meios físicos de comunicação *Ethernet*, que possibilitam tráfego de dados de diferentes serviços no mesmo meio, aliando desempenho razoável e baixo custo, tornaram-se a melhor opção para o desenvolvimento de novas redes. Entretanto, este fato não torna as redes baseadas em Modbus *serial* obsoletas, pois como visto no capítulo 3, o encapsulamento das mensagens torna perfeitamente possível a integração de redes deste modelo com redes *Ethernet*.

Analisando a esfera de utilização do protocolo DNP3, observou-se um grande avanço de funcionalidades em relação ao Modbus: no quesito “informação”, já é possível gerar mensagens com estampa de tempo antes do processo de encapsulamento *Ethernet*, fator bastante favorável à utilização deste protocolo em sistemas de energia, em que a informação do tempo de ocorrência de eventos é essencial. Outro fator que pende a balança de decisão a favor do DNP3 é a possibilidade de configurar mensagens espontâneas nos dispositivos remotos, não sendo necessário que os dispositivos de controle tenham de enviar requisições de *status* de tempos em tempos de forma desnecessária: um grande avanço funcional em relação ao Modbus que aumentou a complexidade lógica das mensagens, fator pouco importante devido à boa capacidade e rapidez de processamento de dados dos IED’s atuais.

O capítulo relativo à organização lógica do padrão IEC61850 foi deixado por último justamente para acompanhar o crescimento de funcionalidades dentre os protocolos aqui abordados. Dentre todas as características aqui citadas que impulsionam sua utilização no mercado, talvez a mais marcante seja a presença das rápidas mensagens GOOSE, fator que reduziu extremamente os custos de implementação de cabeamento em subestações e redes em geral. A possibilidade de sinalizar *status* críticos de dispositivos de segurança através de mensagens facilitou não apenas a implementação física, mas também o entendimento da rede em situações de manutenção: com um simples *notebook* portando um *software* visualizador de tráfego de dados conectado ao *switch* de uma rede, é possível ter uma visão geral de todos os *status* de dispositivos (fato que será observado nos próximos

tópicos). Sem dúvida, IEC61850 conquistou clientes ao propor esta mudança drástica de sinalização de *status*, visto que um dos principais fatores que são valorizados na implementação de novas redes é o custo.

Com o intuito de aproximar o leitor das utilizações práticas das normas abordadas anteriormente, apresenta-se nos próximos tópicos testes de comunicação em bancada utilizando os três protocolos abordados até o momento.

6.1 – Teste Protocolo Modbus

O teste em bancada do protocolo Modbus foi realizado utilizando um medidor de modelo ION9600, representado na Figura 61 abaixo, conectado a um *notebook* com interface PCMCIA através de um meio de comunicação serial. Para que fosse possível visualizar o tráfego de informações entre o dispositivo e o *notebook*, utilizou-se o *software ASE2000*. Com este *software*, é possível visualizar as mensagens que trafegam através do meio físico serial e também enviar comandos ao dispositivo.

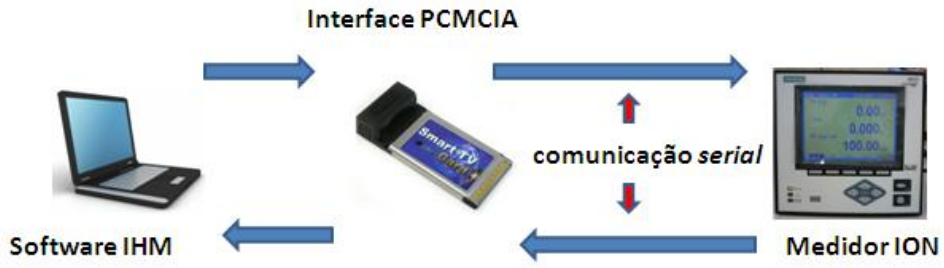

Figura 60 - Layout bancada

Figura 61 - Medidor ION utilizado para teste de comunicação Modbus

A tela principal do *software* – que pode ser vista na Figura 62 - é dividia em três partes: na parte superior, é possível visualizar uma pequena linha do tempo gráfica que sinaliza o momento em que dados estão sendo transmitidos (dados com sentido dispositivo remoto para IHM como também o sentido inverso). Na janela *Line Monitor,* é possível visualizar o conteúdo hexadecimal das mensagens e também uma pequena descrição do tipo de comando que está sendo transmitido. Por último, na janela *Exchange List,* é possível visualizar todos os comandos aos quais o dispositivo conectado pode ser submetido.

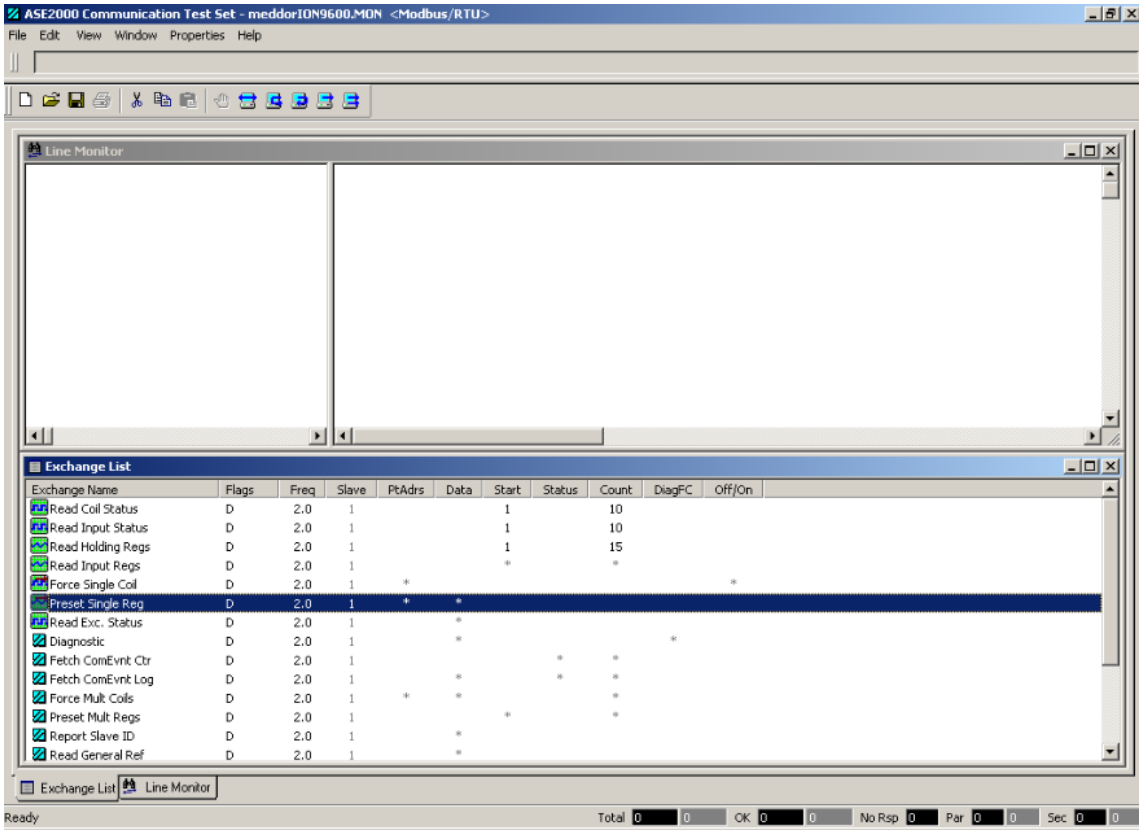

Figura 62 - Tela inicial do software AES2000 - comunicação Modbus

6.1.1 – Análise do tráfego de dados

Como visto nos capítulos teóricos apresentados anteriormente, dentre os protocolos aqui abordados, o Modbus é o mais simples: não possui estampa de tempo, e sua comunicação entre dispositivos é do tipo requisição-resposta: os dispositivos apenas informam os dados requisitados quando recebem a mensagem de requisição do dispositivo cliente.

Nota-se aqui uma das grandes desvantagens da utilização do meio de comunicação serial: diferente do protocolo IEC61850, o meio físico utilizado na comunicação entre dispositivos deve ser exclusivo para esta comunicação: diferentes serviços não podem compartilhar o mesmo meio físico; portanto, todos os dados apresentados na tela pertencem à comunicação entre o medidor e o dispositivo IHM.

Para inicializar a troca de dados entre os dispositivos em bancada, seleciona-se primeiramente o comando *"Read Holding Regs"* na lista "*Exchange list"* – Figura 63. Este comando é uma espécie de "interrogatório" feito pelo dispositivo requisitante, em que ele deseja ser informado a respeito dos valores de todas as variáveis disponíveis no dispositivo servidor.

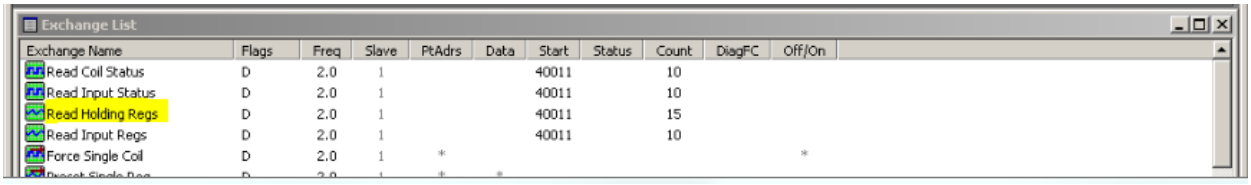

Figura 63 - Envio de comando de leitura de todas as variáveis disponíveis

Assim que o comando descrito acima é enviado, já é possível visualizar tanto no gráfico de dados quanto no monitor o tráfego de mensagens de requisição e resposta, programadas para serem repetidas em um intervalo de tempo pré-definido. A parte superior à linha do tempo no gráfico representa o intervalo de tempo em que *bits* são enviados, enquanto a parte inferior representa o tempo de envio de *bits* recebidos (ambos tendo como referência o dispositivo IHM – de controle).

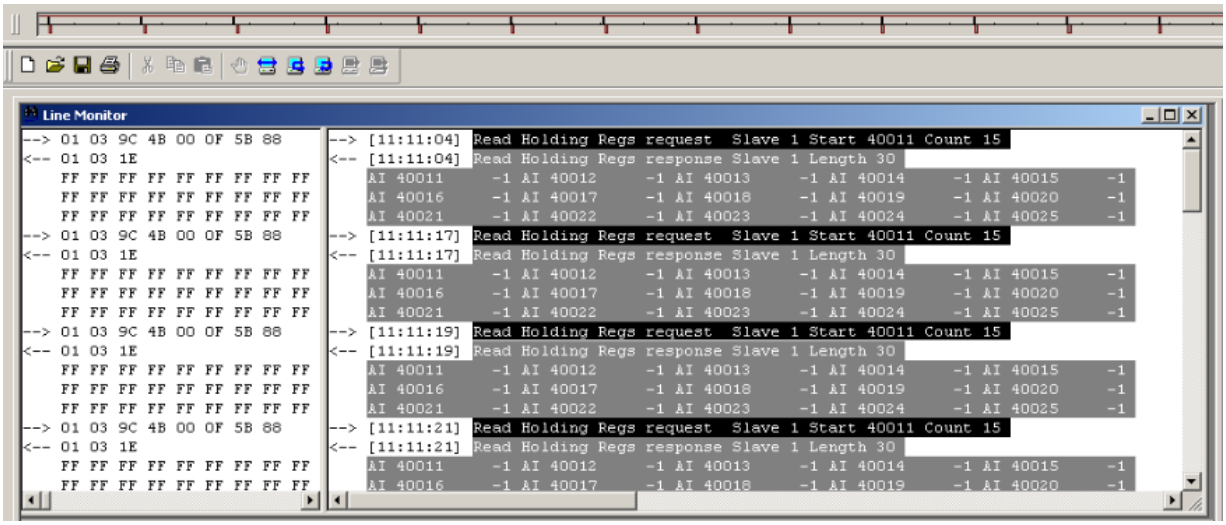

Figura 64 - Visualização do tráfego de dados Modbus

Analisando a figura 64, percebe-se, na tela da esquerda, o código hexadecimal de cada mensagem, enquanto que, na tela da direita, é apresentada uma breve descrição de cada mensagem contendo: o tipo de mensagem (requisição ou resposta); o dispositivo escravo para o qual a mensagem está destinada; a sequência de início da mensagem; tamanho da mensagem. Nota-se que as respostas (grifadas de cor cinza na imagem acima) chegam ao dispositivo IHM quase que instantaneamente. É importante ressaltar que o

tempo mostrado no *software* não é baseado na mensagem (que não possui estampa de tempo), mas sim no relógio do dispositivo IHM. É possível verificar na tela de leitura hexadecimal que todas as variáveis do dispositivo medidor, localizadas nos endereços de 40011 a 40025, estão com valores “FF”. A interface com o operador é bastante simples: apenas informações essenciais estão disponíveis, e a leitura é feita com dados em hexadecimal.

A utilização deste tipo de comunicação Modbus é bastante comum em sistemas menos complexos e que não necessitam de sincronização. Sua utilização com medidores (como neste exemplo) é bastante comum, visto que a leitura de grandezas de um sistema em tempo real deve ser feita da forma mais rápida possível. Em situações em que é necessário integrar uma rede Modbus em outra rede com diferente meio de comunicação (uma rede IEC61850, por exemplo), opta-se por encapsular as mensagens em *frames Ethernet*, criando a possibilidade de adicionar informações não disponíveis no protocolo Modbus, como estampa de tempo, por exemplo.

6.2 – Testes Protocolo DNP3

Para simular a utilização do protocolo DNP3, utilizou-se um dispositivo de controle remoto denominado SICAM E-MIC (Figura 66): ele é composto por um módulo de alimentação, um módulo de controle que realiza ações de lógica programáveis a partir de entradas reais e virtuais disponíveis, e módulos expansíveis de I/O, digitais e analógicos. A arquitetura montada em bancada está representada na figura abaixo:

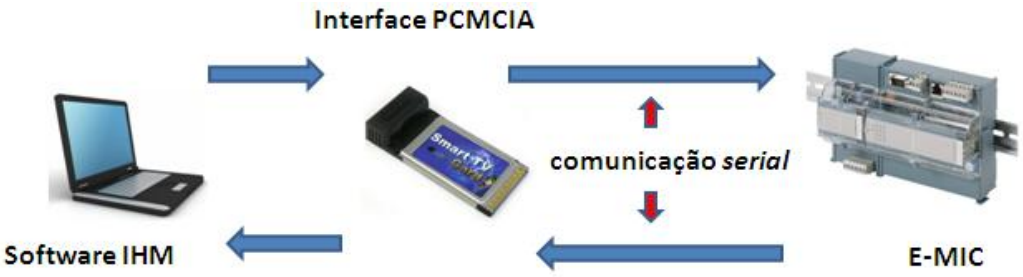

Figura 65 - Layout bancada

Figura 66 - SICAM E-MIC – simulação protocolo DNP3

O *software* utilizado para visualizar o tráfego de dados e simular ações de controle é o mesmo utilizado na simulação do protocolo Modbus, o ASE2000. Da mesma forma que no caso anterior, o tráfego de dados pode ser visto na janela "*Line Monitor",* enquanto as ações de controle disponíveis podem ser visualizadas na janela "*Exchange List"*.

6.2.1 – Análise do tráfego de dados

Diferente da comunicação Modbus, no protocolo DNP3 algumas ações de inicialização são necessárias antes de começar o envio de mensagens de requisição. Primeiramente, é necessário enviar uma mensagem para limpar o canal de comunicação, denominada "*Data Link request".* As posteriores ações só podem ser tomadas após o dispositivo IHM receber a confirmação de tal requisição (*"Ack Response")*. Tais mensagens estão apresentadas na Figura 67. Nota-se que os *bytes* de controle da transmissão estudados no capítulo teórico (DIR, PRM, FCV e DFC) podem ser visualizados na descrição da mensagem.

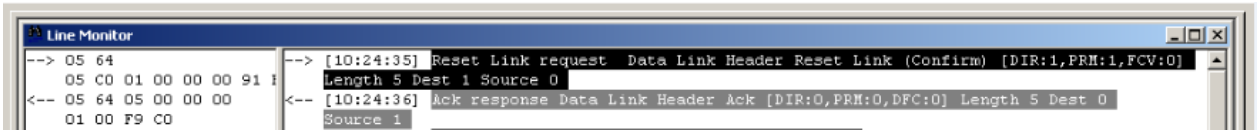

Figura 67 - Inicialização da Transmissão DNP3 - Reset Link Request

Recebida a confirmação de *reset* do link de comunicação, é necessário sincronizar os dispositivos: primeiro, requisita-se a medição do tempo de atraso entre os dispositivos através do comando "*Delay Measurement request",* e após receber a informação, escreve-se o tempo e a data no dispositivo remoto. Tais ações podem ser vistas nas Figuras 68 e 69:

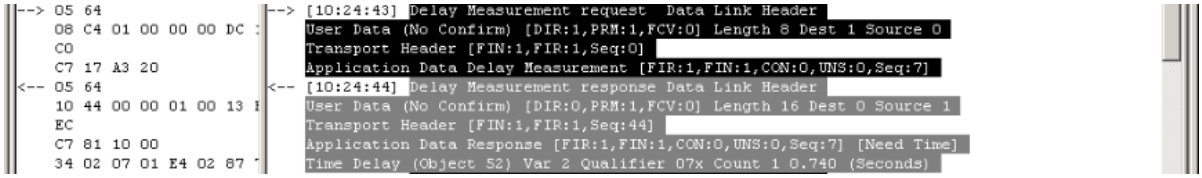

Figura 68 - Requisição de medida de tempo de atraso

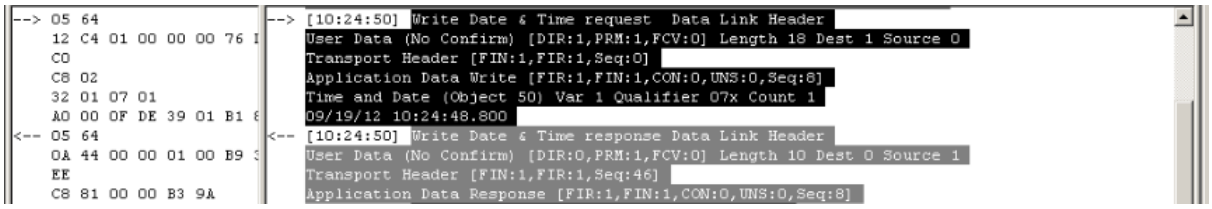

Figura 69 - Requisição de escrita de estampa de data/tempo

Em ambos os casos, é possível verificar informações do cabeçalho de transporte e dos *bits* que carregam informações para remontar, na ordem correta, mensagens que foram transmitidas em pacotes (os *bits* FIR e FIN, vistos no capítulo teórico do protocolo DNP3). Em ambas as situações, o dispositivo remoto manda mensagem de confirmação de sucesso das ações requisitadas (destacadas em cinza nas figuras acima).

Da mesma forma que foi feito no protocolo Modbus, é necessário atualizar o dispositivo cliente a respeito dos valores de todas as variáveis presentes no dispositivo remoto antes de se inicializar a troca de comandos entre os dispositivos. Tal ação pode ser realizada através do comando "*Class1/2/3/0 Data Request":* esta é uma espécie de "*General Interrogation"* do dispositivo remoto, situação em que este é obrigado a enviar ao requisitante o *status* atualizado de todas as variáveis nele armazenadas. Tal situação pode ser vista nas Figuras 70 e 71:

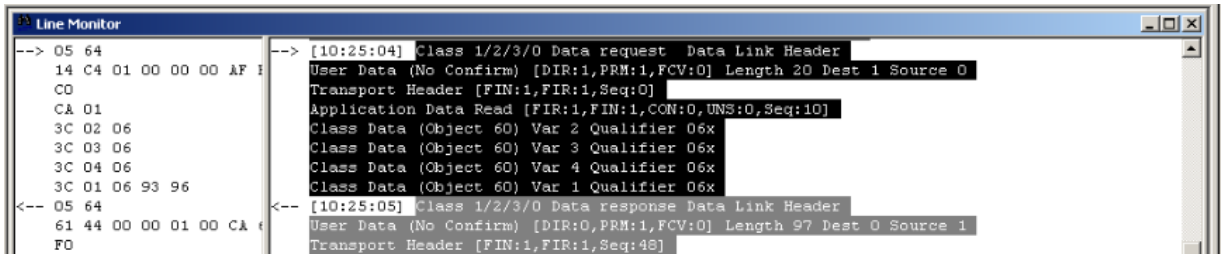

Figura 70 - Atualização de valores de variáveis - Data Request

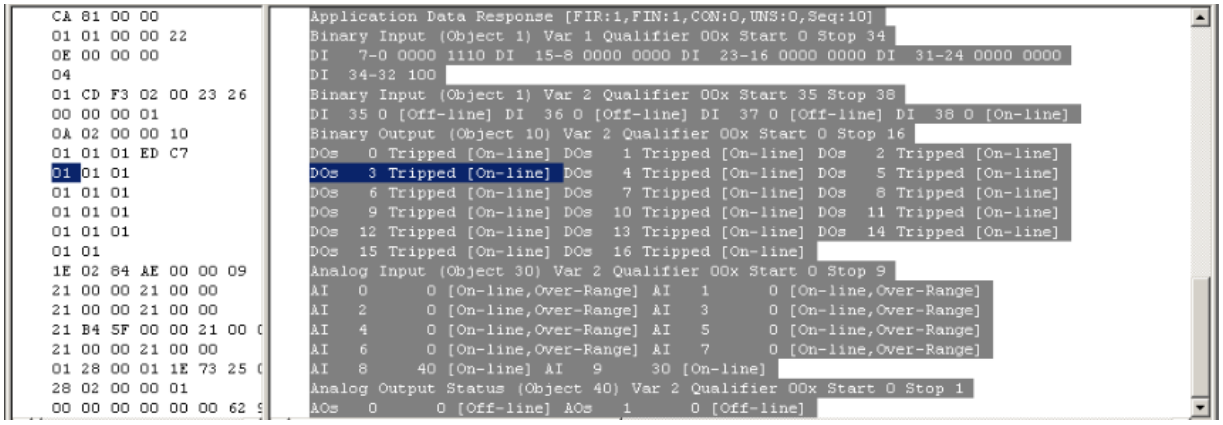

Figura 71 - Atualização de valores de variáveis - Data Response

Podemos visualizar, na resposta do dispositivo remoto, o *status* e os valores das entradas e saídas, digitais e analógicas. Finalizados estes passos de inicialização da comunicação, pode-se enfim iniciar a transmissão de mensagens de requisição de acordo com a necessidade do usuário.

6.2.2 – Relatórios Espontâneos

Diferente da comunicação baseada em Modbus, é possível programar os dispositivos remotos da rede para enviar automaticamente relatórios contendo informações atualizadas dos valores de I/O's digitais toda vez que houver uma mudança de nível lógico. Este método de atualização via "relatórios espontâneos" diminui significantemente o tráfego de dados na rede, já que não é necessário que o dispositivo de controle/supervisão precise enviar a, cada intervalo pré-definido pelo usuário, mensagens de atualização de *status* dos dispositivos remotos: ele será informado de qualquer alteração que ocorrer através destes relatórios não solicitados.

Para simular esta situação, interligou-se uma das saídas auxiliares da fonte de alimentação do dispositivo aos pontos de entrada digital: dessa forma, espera-se o tráfego de relatórios espontâneos de mudanças de nível lógico destas entradas. A imagem a seguir mostra como são apresentados estes relatórios.

Figura 72 - Relatórios espontâneos

Primeiramente, o dispositivo remoto envia uma mensagem ao dispositivo de controle (mensagem com estampa de tempo [10:35:13]) do tipo "*binary input change response"* – mudança de entrada binária. Esta mensagem contém a informação de que a entrada digital de endereço 6 foi para nível lógico "1" na data 09/19/12 às [10:35:08.706]. Percebe-se que existe um pequeno *delay* entre a chegada da mensagem ao dispositivo de supervisão em relação ao momento em que de fato ocorreu a mudança do nível lógico, causado pelos intervalos de *buffer* de atualização do dispositivo. Além desta informação principal, é possível verificar informações referentes ao cabeçalho de transporte (*bits* FIR e FIN), dados da mensagem da camada de aplicação, o tipo da entrada que está variando, entre outras.

Após o envio da mensagem de alteração de entrada, o dispositivo de controle responde com uma mensagem de confirmação de recebimento da informação, destacadas em preto na imagem acima. Todo relatório espontâneo recebido acaba gerando uma mensagem de confirmação.

Assim que o cabo da alimentação é desconectado da entrada digital, o seu nível lógico retorna para "0", gerando um novo relatório espontâneo com o mesmo formato do anterior. Este procedimento ocorre com todos os endereços lógicos presentes no dispositivo.

6.2.3 – Envio de comando de controle de dispositivo

Para simular o tráfego de uma mensagem de controle, testa-se neste tópico o envio de uma requisição de mudança de nível lógico de uma saída binária partindo do *software* de simulação de controle. Para que a saída binária seja alterada, precisa-se primeiro requisitar a seleção do dispositivo: quando uma saída está selecionada por um determinado dispositivo de controle, não existe o risco do dispositivo remoto receber comandos de diferentes dispositivos de controle sobre a mesma saída.

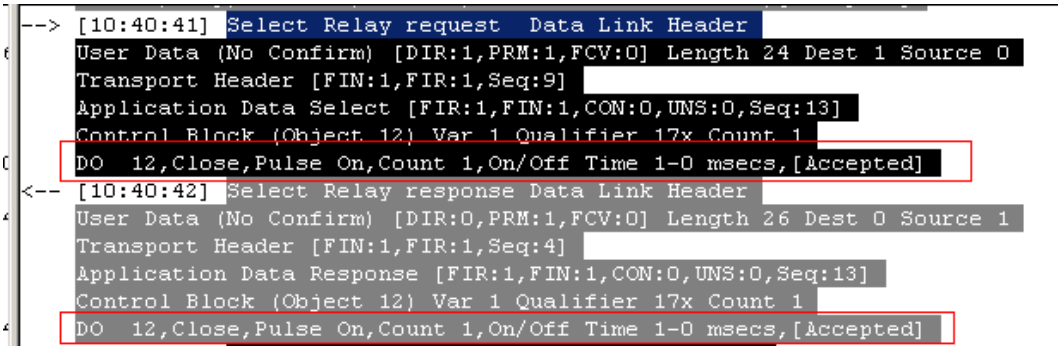

Figura 73 - Seleção de dispositivo

A mensagem destacada em preto na Figura 73 representa a solicitação de seleção da saída binária de endereço 12. É possível verificar que, a partir da seleção, o operador pode enviar à remota um comando de fechamento, um pulso de sinal, um sinal com período de oscilação, etc. Como nenhum outro dispositivo de controle estava selecionando aquela saída digital, a remota responde com a mensagem de aceitação da seleção (destacada em cinza).

Só a partir desta seleção é possível enviar de fato o comando, mostrado na Figura 74; após a mudança do nível lógico da saída, recebe-se a mensagem de confirmação do dispositivo remoto (mensagem destacada em cinza).

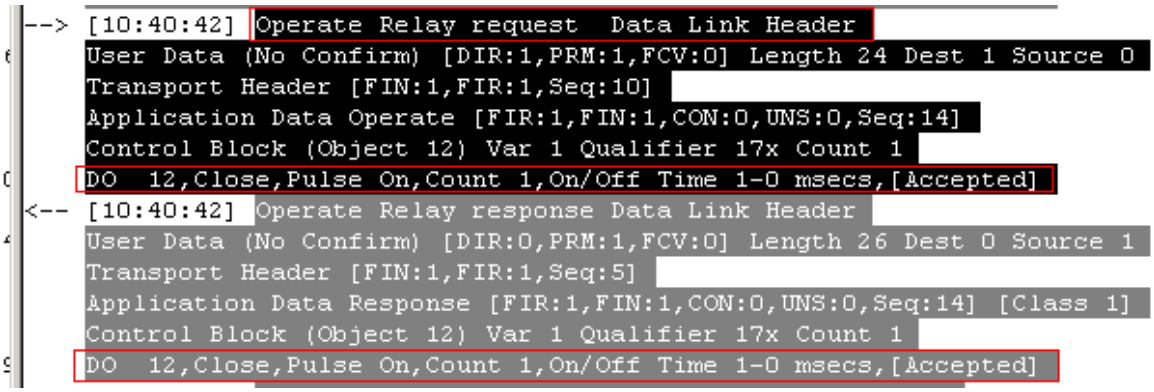

Figura 74 - Mensagens de comando nas binárias de saída

Como visto no tópico anterior, já é esperado um relatório espontâneo sinalizando a mudança de nível lógico da entrada do dispositivo remoto que recebeu o comando de atuação, mostrado na figura 75 abaixo:

```
<-- [10:40:45] Binary Input Change response Data Link Header
    User Data (No Confirm) [DIR:0,PRM:1,FCV:0] Length 22 Dest 0 Source 1
    Transport Header [FIN:1,FIR:1,Seq:6]
    Application Data Unsolicited Message [FIR:1,FIN:1,CON:1,UNS:1,Seq:7]
    Binary Input Change (Object 2) Var 2 Qualifier 17x Count 1
    DI  38 1 [On-line] 09/19/12 10:40:40.207
--> [10:40:45] Confirm request  Data Link Header User Data (No Confirm) [DIR:1,PRM:1,FCV:0]
    Length 8 Dest 1 Source 0
    Transport Header [FIN:1,FIR:1,Seq:11]
    Application Data Confirm [FIR:1,FIN:1,CON:0,UNS:1,Seq:7]
```

Figura 75 - relatório espontâneo decorrente de envio de comando

6.3 – Teste Protocolo IEC61850

Para simular em bancada uma rede que utiliza o protocolo IEC61850, utilizou-se um relé de proteção digital do tipo 7VK61 (Figura 77), um *switch* com portas ópticas (Figura 78) e um *notebook* contendo: *software* de parametrização do relé (Digsi); *software* de monitoramento de tráfego de rede (MMS_Ethereal) e *software* de simulação de interface IHM (IEC_Browser). Tais dispositivos foram conectados de acordo com o *layout* mostrado na figura 76.

Figura 76 - Layout bancada

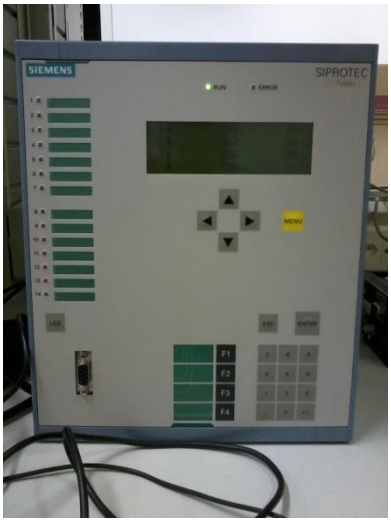

Figura 77 - Relé de proteção digital 7VK

Figura 78 - Switch com interface óptica

O primeiro *software* é responsável por carregar no IED a versão de *firmware* que contém as lógicas de proteção que serão utilizadas pelo relé na rede em que este será inserido, além de todas as lógicas envolvendo os sinais de entrada recebidos pelo dispositivo; é no momento de parametrização do relé que se definem quais serão as proteções habilitadas.

O *software* de monitoramento de tráfego age como um "espião" da rede: todos os dados que estiverem trafegando a nível físico (camada um do modelo OSI) podem ser lidos pelo *software* apenas conectando o *notebook* ao *switch* pertencente à rede. A figura 79 mostra como o usuário consegue ver com facilidade que tipo de informação está transitando em sua rede.

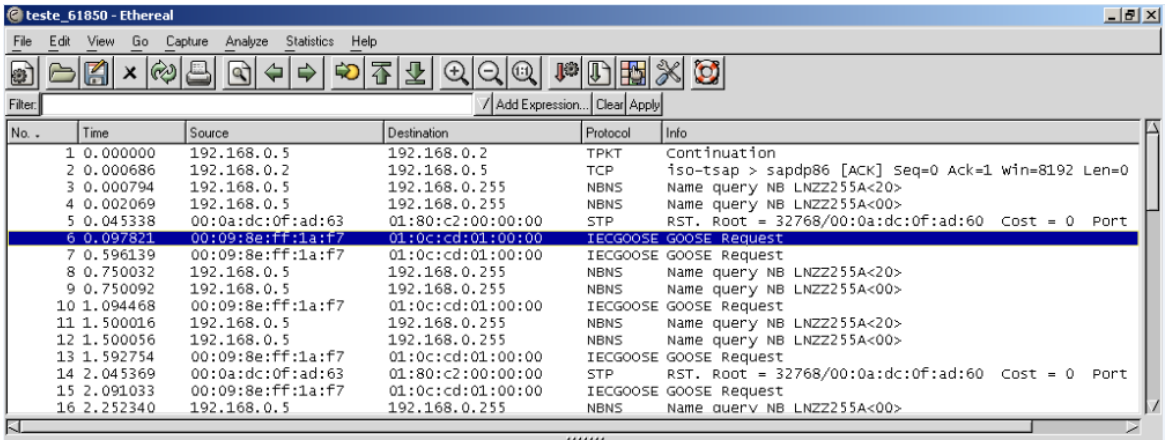

Figura 79 - Ethereal - tráfego de dados de diversos serviços

Destaca-se neste ponto um dos grandes atrativos da utilização do IEC61850: a possibilidade de utilizar o mesmo meio físico da rede para trafegar dados de diversos serviços. Podemos verificar através da coluna *Protocol* que mensagens de diferentes *frames* estão transitando no mesmo meio físico: mensagens GOOSE, TCP, relatórios *unicast*. Como vimos nos tópicos anteriores, esta versatilidade não é aplicável aos protocolos Modbus e DNP3, que necessitam do meio físico de comunicação exclusivamente para o protocolo.

O *software* que simula a existência de uma interface homem-máquina faz também o papel de um sistema de controle que envia comandos ao relé de proteção, como uma espécie de operador de um sistema supervisório. Para que fosse possível visualizar os resultados de uma ação de comando em um dispositivo de bancada, foi necessário simular virtualmente no IED algum dispositivo de proteção que estaria a ele atrelado: optou-se aqui pela simulação de um disjuntor denominado "*DJ_Bay"*, uma das situações mais comuns encontradas em campo, em que um disjuntor realiza a proteção de um vão conectado à uma barra alimentadora. Tal simulação pôde ser realizada através do *software* de parametrização do relé.

Para sinalizar todos os estados possíveis de um disjuntor, são necessários dois *bits* (quatro possibilidades de combinações no total): os estados "aberto/fechado" são

representados por *bits* alternados, enquanto situações em que nenhum dos estados está sinalizado (em processo de abertura/fechamento ou então falha de equipamento) são sinalizadas por *bits* iguais. Além da criação de um disjuntor virtual, criou-se também um sinal virtual, que é comutado pelo acionamento da tecla F4 presente no teclado frontal do relé: esta tecla, ao ser pressionada, alterna o valor lógico deste alarme virtual entre “0” e “1”. Como visto no capítulo teórico do protocolo, todos os dispositivos pertencentes a uma rede 61850 são organizados em nós lógicos. Este alarme virtual, portanto, é armazenado no nó lógico de controle de alarmes do relé, com endereço “*R-VKCTRL/LLN0GOControl_Alarmes”.*

6.3.1 – Transmissão de Mensagens GOOSE

A partir da parametrização do relé, descrita acima, a cada vez que o botão F4 é pressionado, inicia-se o processo de distribuição *multicast* de mensagens GOOSE na rede, indicando uma alteração de estado de uma variável que estava sendo monitorada. A figura 80, abaixo, mostra a sequência de mensagens que é disparada pelo relé na rede via GOOSE (basta aplicar um filtro para selecionar apenas as mensagens deste tipo):

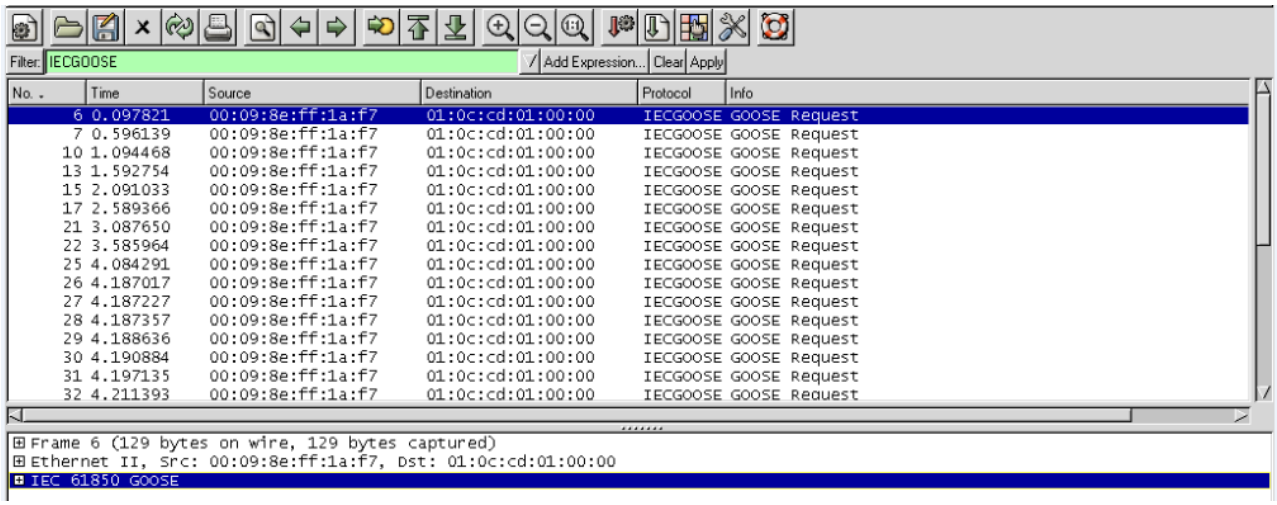

Figura 80 - Distribuição de mensagens GOOSE na rede

É possível verificar que, assim como foi visto no capítulo teórico, o espaçamento de tempo de disparo entre as mensagens GOOSE perto do instante em que a variável sofreu alteração é bastante curto; o intervalo de tempo entre as mensagens vai se espaçando cada vez mais ao se afastar deste ponto. Na figura acima, podemos identificar o exato instante em que cada mensagem GOOSE foi capturada pela interface IHM, o endereço do dispositivo de origem, o endereço do dispositivo de saída e o tipo de mensagem. Se tomarmos o instante de tempo de cada mensagem e organizá-los em uma linha do tempo, teremos a distribuição apresentada no gráfico da Figura 81.

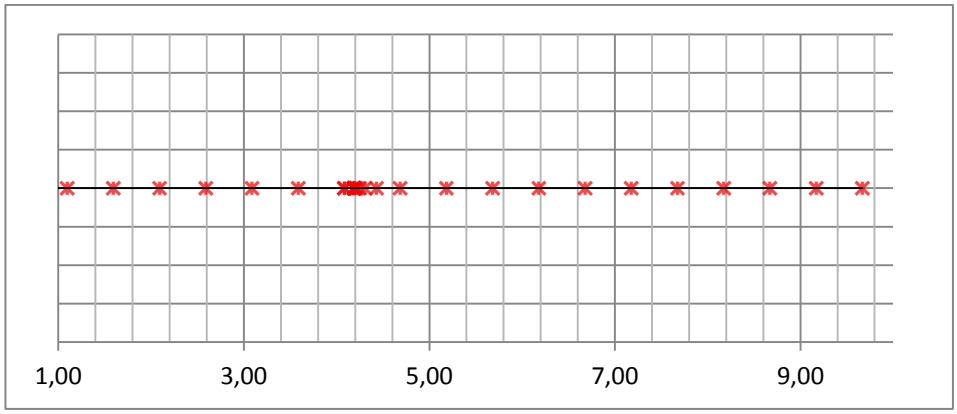

Figura 81 - Linha do tempo de transmissão de mensagens GOOSE

Percebe-se o aumento significativo de tráfego de mensagens GOOSE próximo de quatro segundos, momento em que possivelmente houve uma alteração do nível lógico da variável alarme. Para melhor analisar este instante, a Figura 82 mostra com mais detalhes o disparo de mensagens entre quatro e cinco segundos:

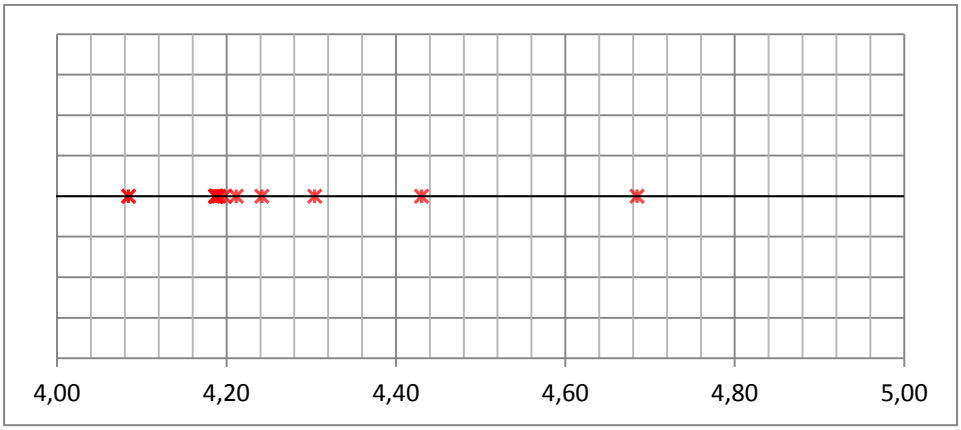

Figura 82 - Detalhe do momento de alteração de variável

Para confirmar a mudança do nível lógico do sinal de alarme que desencadeou este aumento de mensagens GOOSE perto dos 4,2 segundos, basta analisar o *frame* da mensagem antes e depois deste ponto. Selecionando, por exemplo, a mensagem do instante 1,094468 segundos, apresentada na Figura 83, pode-se extrair os seguintes dados: o intervalo entre esta mensagem e a anterior (aproximadamente 0,5 segundos); a localização deste alarme dentro da estrutura da rede no nó lógico “Control_Alarmes”; o tempo em que a mensagem foi gerada no IED; o *status* da variável, naquele instante em “1” (*true*).

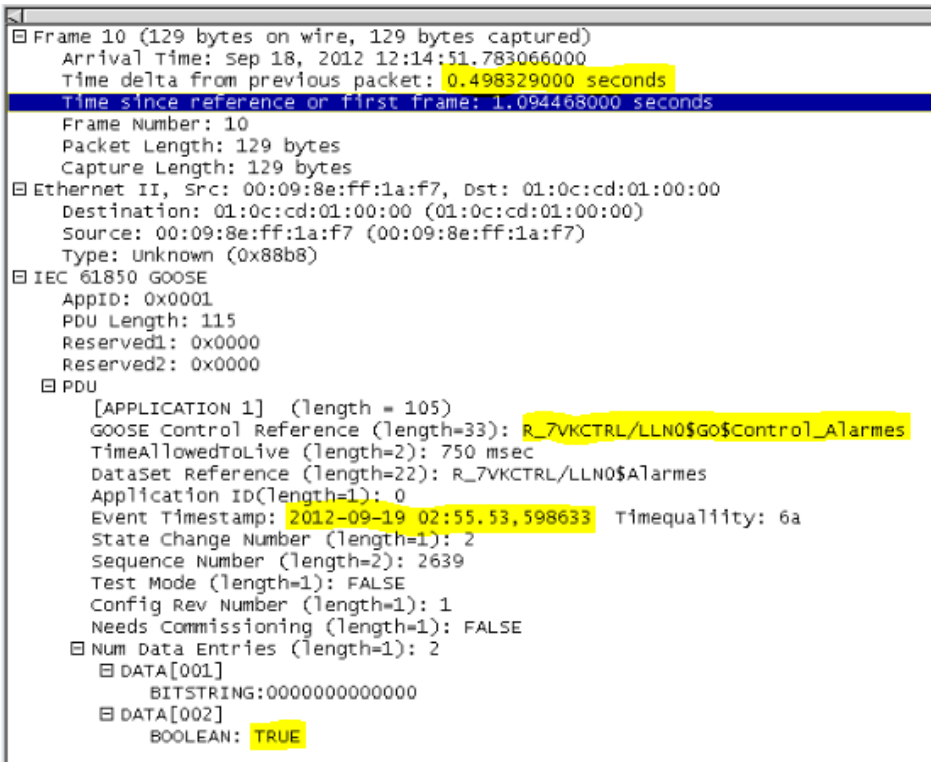

Figura 83 - Mensagem GOOSE anterior à variação de grandeza monitorada

Em seguida, analisa-se uma mensagem localizada após o instante de aumento intensivo do tráfego das mensagens, por exemplo, a que está próxima do marco de 4,7 segundos, como apresentado na Figura 84.

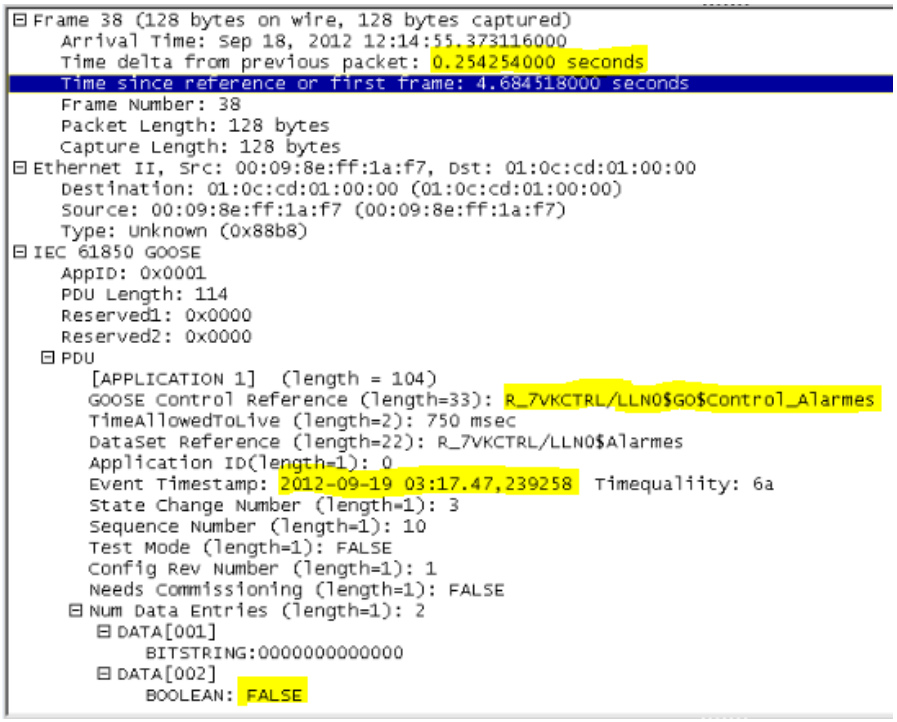

Figura 84 - Mensagem GOOSE após a variação de grandeza monitorada

Pode-se confirmar a mudança do nível lógico da variável (*false*) e também a diminuição do intervalo de tempo entre esta mensagem e a anterior, significantemente menor (aproximadamente 0,25 segundos, praticamente metade do intervalo visto no item anterior). A partir da análise destes *frames* utilizando o *software* de visualização de tráfego de dados, comprova-se o funcionamento da repetição de envio de mensagens GOOSE em intervalos cada vez menores a partir de um determinado evento, garantindo estatisticamente que, em algum momento, pelo menos uma das mensagens de *status* chegará ao dispositivo de destino.

6.3.2 – Transmissão de Relatórios Espontâneos

Como citado na parte teórica de estudo do protocolo IEC61850, uma das grandes vantagens de utilização do protocolo é a possibilidade de seleção de monitoramento apenas de variáveis que interessam ao cliente, não havendo a necessidade de reportar dados de todos os parâmetros presentes no relé de proteção.

Através do *software* que simula o papel de um dispositivo IHM, é possível configurar através de uma janela de seleção todas as variáveis que deverão ser reportadas ao dispositivo de supervisão toda vez que houver uma variação no valor destas, um conjunto de valores denominado *dataset,* que são reportados em relatórios denominados *relatórios espontâneos*. Dessa forma, economiza-se em largura de banda (o número de dados trafegando na rede é reduzido de acordo com a necessidade do sistema), e o entendimento da leitura de dados por parte do operador torna-se muito mais claro (o operador sabe exatamente de quais dados ele está esperando receber informações; não é necessário gastar tempo analisando e separando dados realmente úteis de dados que não estão sendo utilizados). A figura abaixo mostra a simplicidade de seleção de variáveis que irão compor o *dataset* do relé digital:

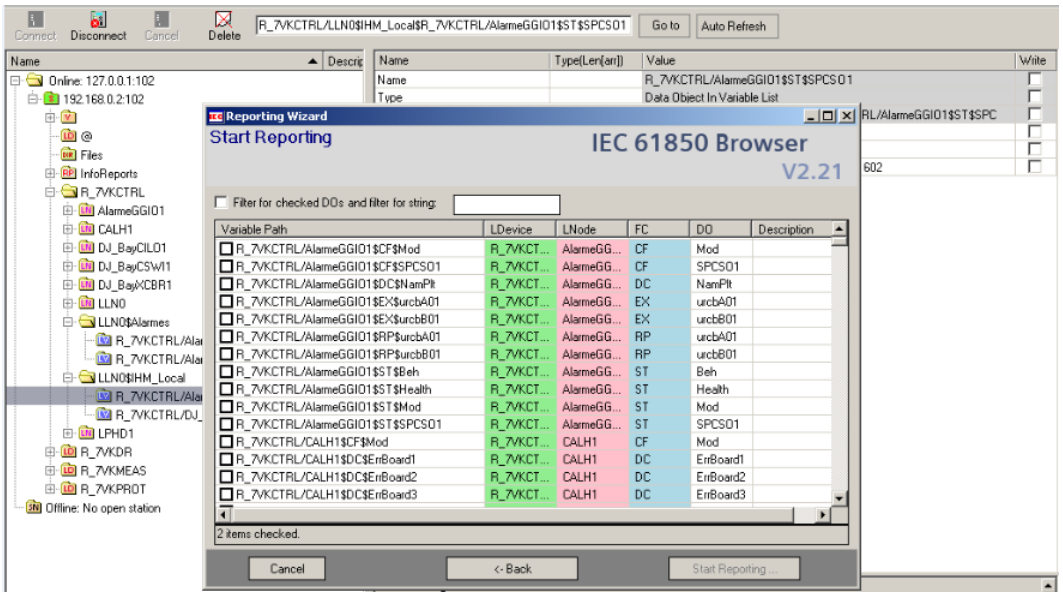

Figura 85 - Software IEC Browser - Seleção de variáveis do dataset

No exemplo aqui proposto, o *dataset* de monitoramento é composto apenas pela variável “Alarme”, ativada pelo botão F4 do painel frontal do relé. Desta forma, o dispositivo de IHM (neste caso, o *notebook* com o *software* de simulação) é inscrito como uma espécie de cliente que receberá a atualização de qualquer mudança de qualquer variável que esteja inclusa no *dataset* destinado apenas a ele.

Nota-se aqui um dos grandes pontos favoráveis à utilização do protocolo IEC61850: um mesmo dispositivo de supervisão pode conter diferentes *datasets* para diferentes dispositivos da rede: todos os dispositivos inscritos no outro receberão informações apenas das variáveis que os interessam, ou seja, nenhuma informação será transmitida sem que exista pelo menos um destinatário esperando-a.

Analisando passo-a-passo como é realizada a transmissão destes relatórios espontâneos, pode-se destacar que, assim que um *dataset* de um determinado dispositivo é criado e seu dispositivo remetente é definido, gera-se automaticamente um relatório denominado “*general interrogation”:* este relatório tem como objetivo mostrar ao dispositivo remetente o valor atualizado de todas as variáveis que a ele interessam. A partir deste momento, só haverá uma nova geração de relatórios se ocorrerem alterações de valores ou se for programada a necessidade de novos relatórios em um tempo definido pelo usuário. A Figura 86 mostra uma sequência de relatórios enviados ao dispositivo IHM decorrente desde o relatório automático (*general interrogation)* até os demais relatórios decorrentes de uma alteração do sinal de alarme (consequência de pressionar o botão F4).

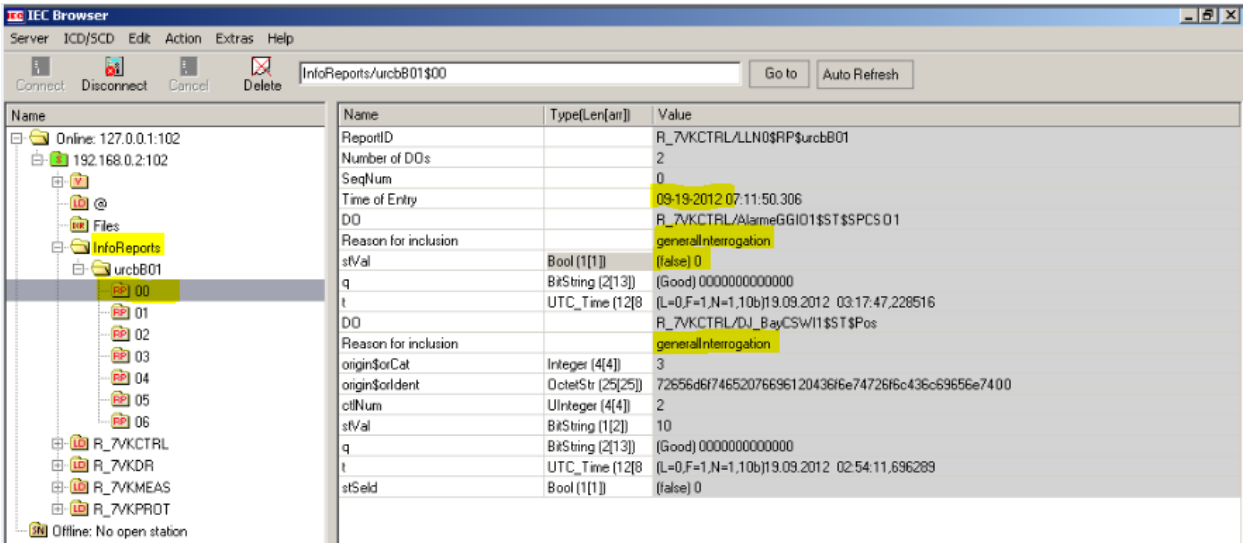

Figura 86 - Relatório General Interrogation

Nota-se, no menu da esquerda, a sequência de todos os relatórios que foram recebidos pela IHM, e, no menu da direita, informações importantes como o valor lógico da variável monitorada, data e hora do relatório e motivo deste relatório estar sendo transmitido. A imagem acima mostra que, no instante de geração daquele relatório, a variável Alarme tinha nível lógico “0”. Pressionando o botão F4, novo relatório espontâneo é gerando contendo as seguintes informações em destaque na Figura 87:

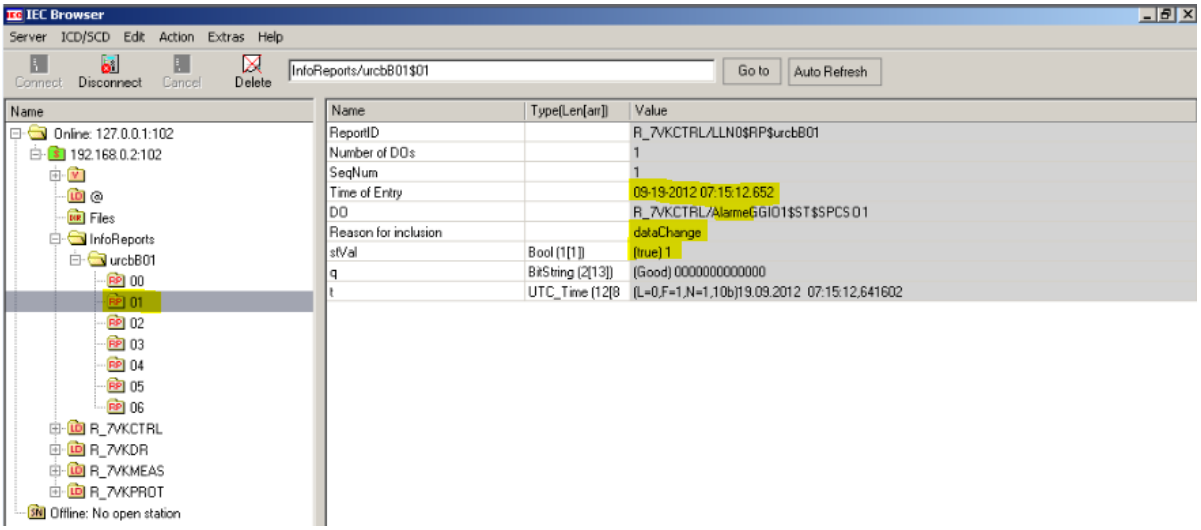

Figura 87 - Relatório espontâneo decorrente de alteração de nível lógico

A imagem acima mostra que o relatório espontâneo seguinte (número 01), gerado na data e hora informada, ocorreu por causa da alteração de nível lógico (*dataChange)* do sinal Alarme.

Através do *software* simulador de IHM, é possível transmitir ao relé o comando de mudança de posição do disjuntor. Por se tratar de objeto de estudo deste exemplo, este dispositivo virtual foi incluso no *dataset* de monitoramento, e, portanto, qualquer alteração relativa à sua posição deve ser notificada ao dispositivo IHM, inscrito na lista de destinatários do *dataset.* A tela de envio de comando para alterar a posição do disjuntor é representada na abaixo:

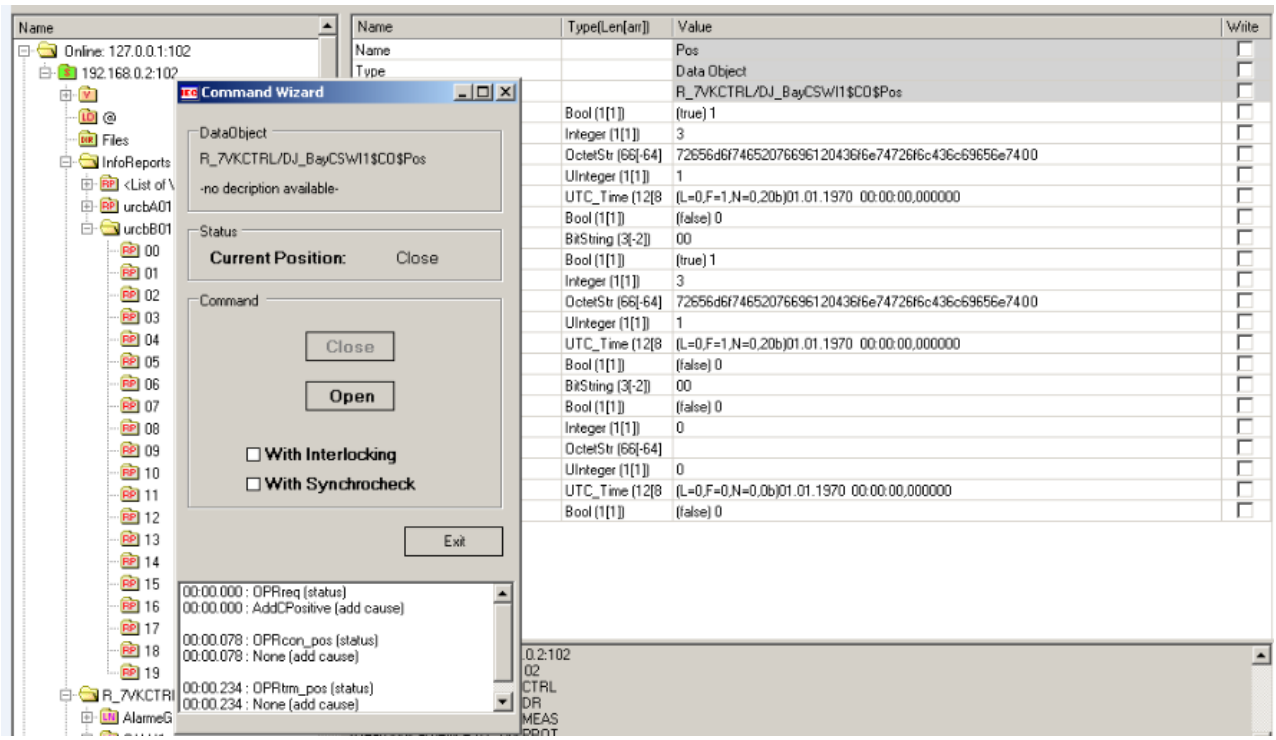

Figura 88 - Comando de controle de posição de disjuntor

A figura acima mostra o resultado do comando de fechamento do disjuntor: é possível verificar que o comando passa por três estágios: requisição da operação de fechamento e confirmação da possibilidade; envio da mensagem de fechamento em si; recebimento da mensagem de confirmação da ação. Antes de ocorrer o envio da mensagem de mudança de um dispositivo, é necessário selecionar o dispositivo e torná-lo exclusivo naquele instante de tempo apenas para quem está enviando a ordem de comando (desta maneira, dois dispositivos de controle não podem controlar o mesmo dispositivo ao mesmo tempo). A seleção de um dispositivo para uma posterior ação de controle também gera um relatório espontâneo do tipo *qualityChange,* e é representado na Figura 89.

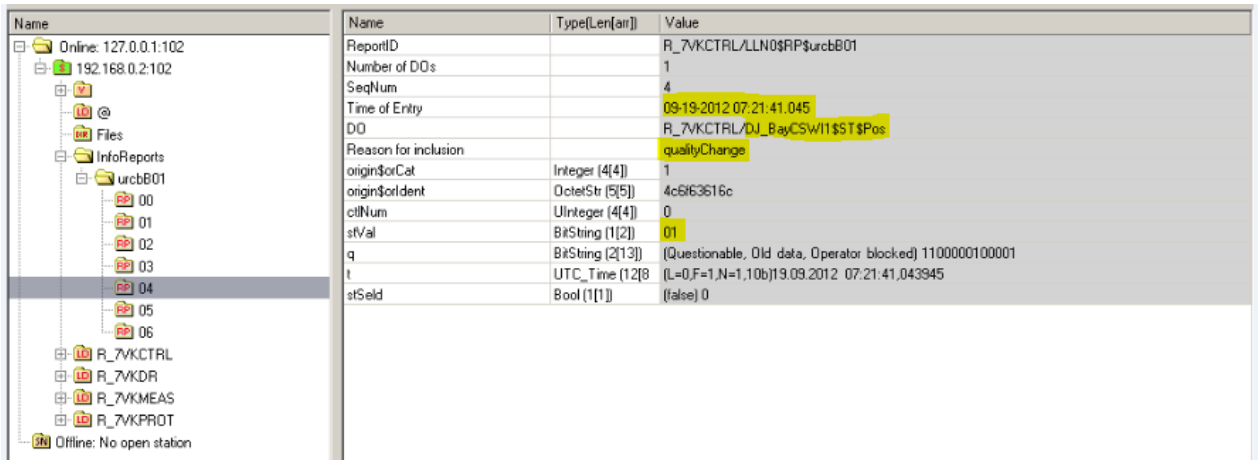

Figura 89 - Relatório espontâneo – QualityChange

Após a seleção do dispositivo e do recebimento da mensagem de confirmação, o sistema de controle tem o “OK” para enviar, enfim, o comando de mudança de posição do disjuntor *DJ_bay*. O resultado pode ser visto nas Figuras 90 e 91, que mostram a transição dos *bits* sinalizadores da posição até o estado “fechado”:

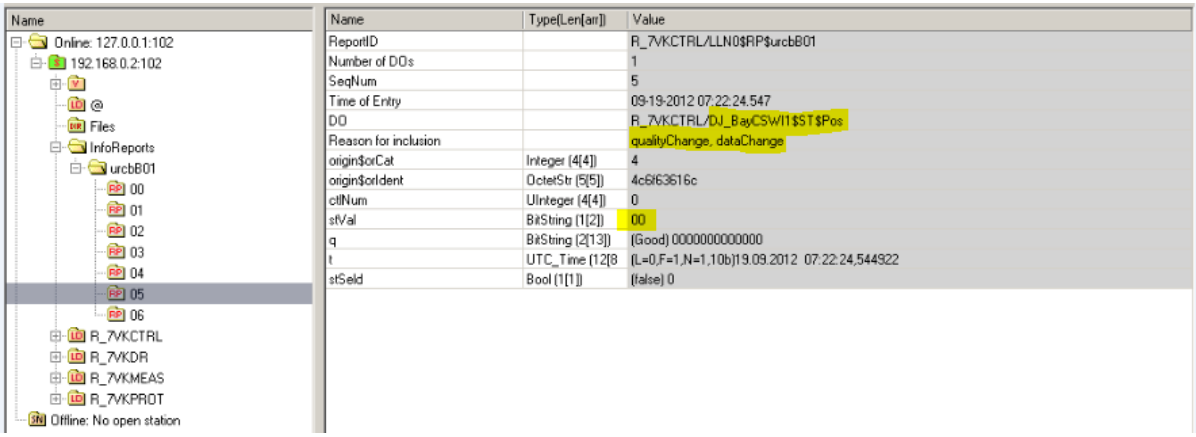

Figura 90 - Relatório espontâneo – dataChange – *fase de transição da posição do disjuntor*

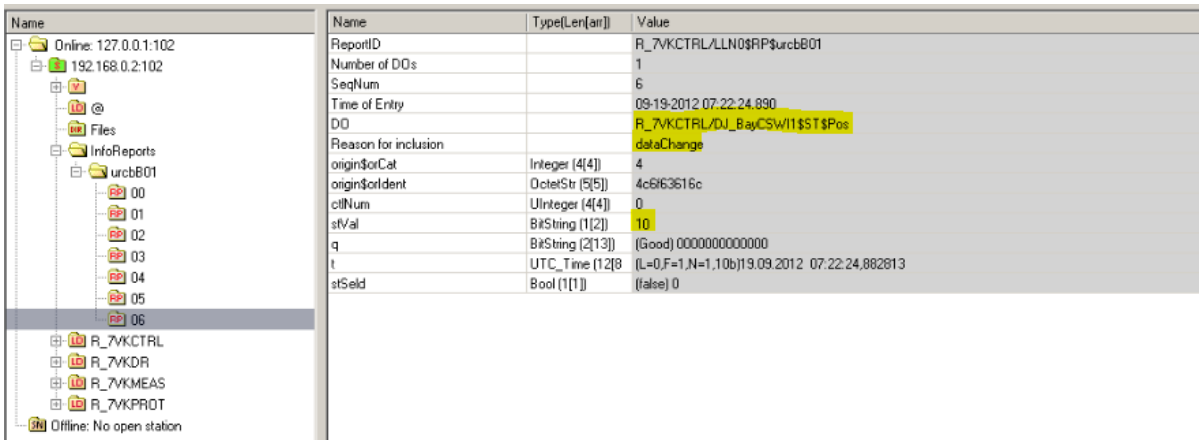

Figura 91 - Relatório espontâneo – dataChange – mudança de posição do disjuntor

Capítulo 7 – Considerações finais

A partir das abordagens teóricas e práticas realizadas ao decorrer deste trabalho, percebem-se pontos favoráveis e pontos fracos na utilização de cada um dos protocolos apresentados. Pensando nos aspectos práticos da utilização de cada um deles em sistemas de energia, cabe ao engenheiro eletricista decidir qual protocolo melhor se encaixa em determinada aplicação da rede. Dentre os principais pontos que devem ser levados em consideração ao se planejar um sistema de comunicação, destacam-se aspectos de manutenção, disponibilidade da rede, confiabilidade do tráfego de dados e custos de implementação.

Analisando a esfera de manutenção de redes de energia, valorizam-se as que possuem informações claras e de fácil acesso ao operador remoto e principalmente ao operador de campo, que trabalha em ambiente de risco e que precisa de informações claras e rápidas. Enquanto os protocolos Modbus e DNP3 apresentam estruturas de dados rígidas e com poucas informações qualitativas e próximas a linguagem de alto nível, o protocolo IEC61850 ganha destaque pela sua organização em nós lógicos personalizáveis, que atrela cada mensagem captada pelo *software* visualizador de tráfego de dados ao dispositivo de origem apenas com uma análise superficial, pois são denominados de forma customizável no momento da parametrização dos dispositivos. Em uma análise inicial, esta característica do protocolo parece ser um atrativo meramente estético; entretanto, a possibilidade de visualizar nomes comuns de dispositivos ao invés de endereços em hexadecimal reduz extremamente as horas de manutenção de sistemas de energia, refletindo positivamente nos custos.

Além da facilidade em identificar e mapear os dispositivos da rede através da visualização das mensagens, pode-se citar o diferencial do protocolo IEC61850 de, no momento de parametrização da rede, criar diferentes *datasets* em um mesmo dispositivo, como visto na parte prática do protocolo. Dessa forma, cada dispositivo supervisório recebe apenas os dados que lhe interessam: economiza-se tanto em largura de banda (informação é gerada e transmitida apenas na quantidade necessária) quanto em tempo de leitura das informações (o operador sabe quais variáveis ele espera receber em cada dispositivo).

Focando agora na implementação física das redes de comunicação, enquanto os protocolos MODBUS e DNP3 exigem que exista um cabo elétrico conectando I/O’s entre dispositivos pertencentes à rede para que uns informem aos outros seus níveis lógicos, o protocolo IEC61850 consegue realizar toda esta sinalização de *status* através da distribuição das mensagens GOOSE. Como visto na parte prática, tais mensagens (de pequeno tamanho e rápida repetição perto de eventos importantes), possuem garantia de entrega aos dispositivos-alvo baseados em estatística matemática. Esta solução reduz drasticamente custos de cabeamento em SE’s de energia, além de tornar as topologias físicas das redes bem menos complexas (pelo fato da diminuição do número de fios). O grande crescimento da utilização do protocolo em novas redes de energia comprova o sucesso e a garantia de confiabilidade da solução. Obviamente, uma queda de um meio de

comunicação físico IEC61850 acarretará em consequências muito mais graves do que a queda de um meio físico que transmite apenas o sinal de um componente, visto que o primeiro carrega informação de diversos equipamentos. Tal fato é consequência de concentrar cada vez mais informações em um único meio de comunicação. Entretanto, opta-se pela escolha de soluções cada vez mais enxutas e de menor custo, valorizando o padrão IEC61850.

O grande ponto que merece destaque ao se planejar uma rede de comunicação é como se deseja ajustar a balança “simplicidade *versus* funcionalidade”: o protocolo Modbus é perfeitamente indicado para situações de comunicação de poucos dispositivos que requerem um tempo de atualização rápido (como o medidor utilizado para simulação na parte prática), mas sua utilização se tornaria impraticável para redes com muitos dispositivos conectados. Além disso, o aumento da versatilidade dos dispositivos presentes no mercado, que podem se comunicar e realizar diferentes serviços em mais de um protocolo, torna muito atrativa a opção de escolha de um protocolo que aceite compartilhar seu meio físico de tráfego com mensagens de outros serviços, como é o caso do IEC61850.

Os testes em bancada descritos nesta monografia foram planejados, em um primeiro momento, buscando testar a comunicação de um mesmo conjunto de aparelhos sob a ação dos protocolos aqui abordados. Entretanto, as condições práticas de disponibilidade de dispositivos para testes prejudicaram esta premissa, forçando a realização dos testes com diferentes dispositivos para posterior análise qualitativa dos dados. Dentre novas possibilidades de testes, pode-se destacar a comunicação através dos protocolos IEC 60870-5-101 e IEC 60870-5-104, padrões do tipo *serial* também bastante utilizados na área de automação de sistemas de energia, e que poderiam ser analisados sob aspecto qualitativo.

A análise aprofundada das organizações lógicas e da construção dos *frames* de mensagens de cada protocolo mostrou-se bastante complexa num primeiro momento, principalmente na abordagem do primeiro padrão aqui apresentado; entretanto, por partilharem semelhanças nos fluxogramas de comunicação e em alguns aspectos de controle de tráfego, a análise tornou-se gradativamente mais simples ao decorrer da pesquisa.

Tendo em vista que este último consegue englobar com facilidade áreas de redes que se comunicam através de outros protocolos, é de se esperar que o padrão IEC61850 continuará a ser a escolha mais pedida no desenvolvimento de novas redes. Os estudos de novas tecnologias que aprimoram os meios físicos de comunicação de dados não são desencorajados pelo padrão, visto que sua organização distribuída entre as sete camadas do modelo OSI isola a escolha do meio físico da estrutura lógica. Por aliar de maneira satisfatória os pilares da engenharia funcional (desempenho, custo e segurança), IEC61850 estará com presença garantida no mercado de comunicação de sistemas de energia pelo menos na próxima década.

Bibliografia

1. **Silva, Ivan Nunes da.** Redes de Computadores – Notas de Aula.

2. **Modbus Organization.** *Modbus Application Protocol Specificartion - V1.1b* – Dezembro/2006.

3. **Modbus Organization.** *Modbus Messaging On TCP/IP Implementation Guide - V1.1b* – Outubro/2006.

4. **M&M Assessoria e Projetos de Sistemas Ltda.** Protocolo de Comunicação DNP Versão 3.0 – revisão 02.

5. **M&M Assessoria e Projetos de Sistemas Ltda.** Protocolo de Comunicação DNP Versão 3.0 – Detalhes de Implementação – revisão 01.

6. **VA TECH SAT GmbH & CO.** *Firmware Description DNPMxx – Distributed Network Protocol* – version A1 – revision 00 – 2002.

7. **Ralph Mackiewcz.** *Overview of IEC61850 and Benefits* – SISCO.

8. **Siemens.** *Efficient Energy Automation with the IEC 61850 Standard Application Siemens AG* – Energy Sector - 2010

9. **Dos Santos, Luis F. & Pereira, Maurício.** Uma Abordagem Prática do IEC61850 para Automação, Proteção e Controle de Subestações – ABB Ltda.

10. **Siemens.** IEC61850 – Treinamento – IC Sector – 2010.

11. **Siemens.** *Ethernet & IEC61850 – Start Up* – Manual – 2011.

12. **Siemens.** *Ethernet & IEC61850 – Concepts, Implementation , Comissioning* – Manual – 2011.

13. **Guerreiro, César.** Desempenhos via cabo e mensagem GOOSE Conforme NORMA IEC61850 – Siemens IC SG EA – 2008.