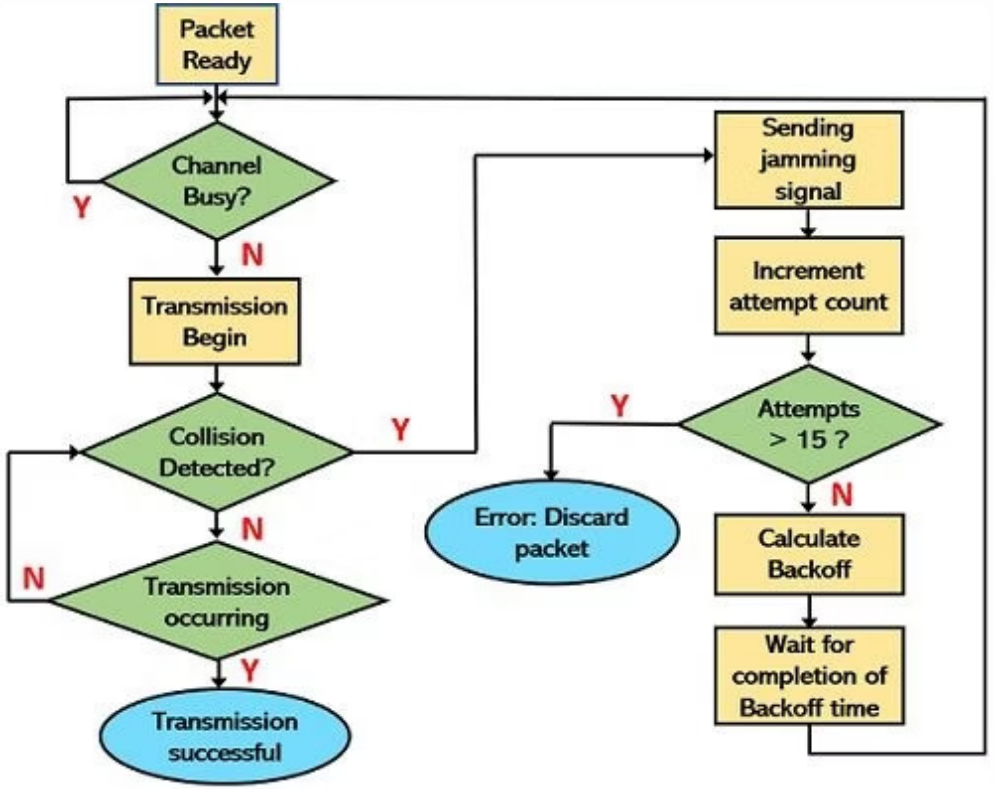
Simulação CSMA/CD

por Lucas Tempass Cerveira

Implementação

- **Simulação de eventos discretos (SED).**
- Baseado nos princípios do livro *Modeling and Tools for Network Simulation*.
- Eventos de envio de pacotes.
- Gerados previamente conforme taxa de transmissão.
- Padrão **10 BASE-T com 2 hosts.**



```
List<Host> hosts = gerarHosts(taxaDePacotes, duracao);

// tempo necessário para transmitir o frame por completo
double TEMPO_TRANSMISSAO = BITS_POR_PACOTE / VAZAO;
double TEMPO_INTERFRAME = 96 / VAZAO; // 8 bytes de interframe

while (true) {
    Host hostProximoPacote = getHostProximoPacote(hosts);
    if (hostProximoPacote == null) break;
    Pacote proximoPacote = hostProximoPacote.getPacotes().peek();
    if (proximoPacote == null) break;

    // adiciona o tempo de espera mínimo especificado para Ethernet
    double tempoProximoPacote = proximoPacote.getTempo() +
TEMPO_INTERFRAME;

    double tempoColisao = validarColisao(getOutroHost(hosts,
hostProximoPacote), tempoProximoPacote);

    if (tempoColisao == null) {
        hostProximoPacote.onSucesso();
        // limita sobreposição de pacotes do mesmo host
        double tempoMinimoProximosPacotes = tempoProximoPacote +
TEMPO_TRANSMISSAO;
        atualizarPacote(hostProximoPacote, tempoMinimoProximosPacotes);
    } else {
        hostProximoPacote.onColisao(VAZAO, tempoColisao);
    }
}
```

```
Pacote pacoteHost = host.getPacotes().peek();

double TEMPO_PROPAGACAO = COMPRIMENTO_BARRAMENTO /
VELOCIDADE_DE_PROPAGACAO_DO_MEIO;

// tempo de chegada do primeiro símbolo de informação
double tempoDeteccao = tempoProximoPacote + TEMPO_PROPAGACAO;
double tempoConclusaoPacote = tempoDeteccao + TEMPO_TRANSMISSAO;

double tempoPacoteHost = pacoteHost.getTempo() + TEMPO_INTERFRAME;

// host vai ser capaz de identificar que meio está ocupado
if (isDetectavelAndHasInterseccao(tempoPacoteHost, tempoConclusaoPacote,
tempoDeteccao)) {
    // atrasa os pacotes para a conclusão do próximo pacote (omitido)
}

// host não será capaz de detectar a portadora
if (tempoPacoteHost <= tempoDeteccao) {
    // considera tempo de transmissão do JAM, pois é necessário ser
concluído
    host.onColisao(VAZAO, tempoDeteccao + TEMPO_JAM);
    return tempoPacoteHost + TEMPO_PROPAGACAO + TEMPO_JAM;
}

return null;
```

```
if (++colisoes > MAX_COLISOES) {
    return removerPacote();
}

Pacote pacote = pacotes.peek();
if (pacote == null) return;

double tempoBackoff = tempo + getTempoBackoffExponencial(larguraDeBanda,
colisoes);

// atrasa envio dos pacotes previstos, imitando um comportamento de buffer
for (Pacote p : pacotes) {
    if (tempoBackoff < p.getTempo()) break;
    p.setTempo(tempoBackoff);
}
```

Algoritmo de *backoff* exponencial

Seleção de um *timeslot* aleatório no intervalo de de 0 a `N` (exclusivo).

- O valor de `N` é baseado em `N = 2^Q`, onde `Q` representa quantidade de colisões do quadro.
- O *timeslot* é definido por `T`, onde `T = M / V`, onde `V` representa a vazão em bits/s e `M` representa a menor quantidade de bits por quadro.
  - `M = 512` e `V = 10.000.000`
- `R` representa um valor de 0 a 1 (exclusivo) gerado por uma função pseudo-randômica.

```
B = R * N * T
```

```
double quantidadeSlots = Math.pow(2, colisoes);
// intervalo de [0, N[
double slot = Math.random() * quantidadeSlots;
// tamanho mínimo do frame de 512 bits (64 bytes)
return slot * 512 / larguraDeBanda;
```

Parâmetros de simulação

Padrão IEEE 802.3

- **10 BASE-T**
- Vazão de 10Mbit/s
- Segmento de 100m

- **Tamanho fixo de quadros com 64 bytes**
- **Quantidade fixa de 2 hosts.**
- Duração de 5 segundos

Constantes

- Velocidade da luz aproximada
  - **C ~= 300.000.000 m/s**
- Velocidade de transmissão no par trançado
  - **V ~= 0.66 * C**

Execução

Cenário 1

2.000 quadros por segundo por hosts ou 2,048Mbits/s teóricos

Resultados

- Colisões: **44**
- Média de colisões/pacote: **0,02**
- Taxa de falhas: **0%**
- Média de *delay* dos quadros: **5µs**

Cenário 2

4.000 quadros por segundo por hosts ou 4.096Mbits/s teóricos

Resultados

- Colisões: **389**
- Média de colisões/pacote: **0,10**
- Taxa de falhas: **0%**
- Média de *delay* dos quadros: **22µs**

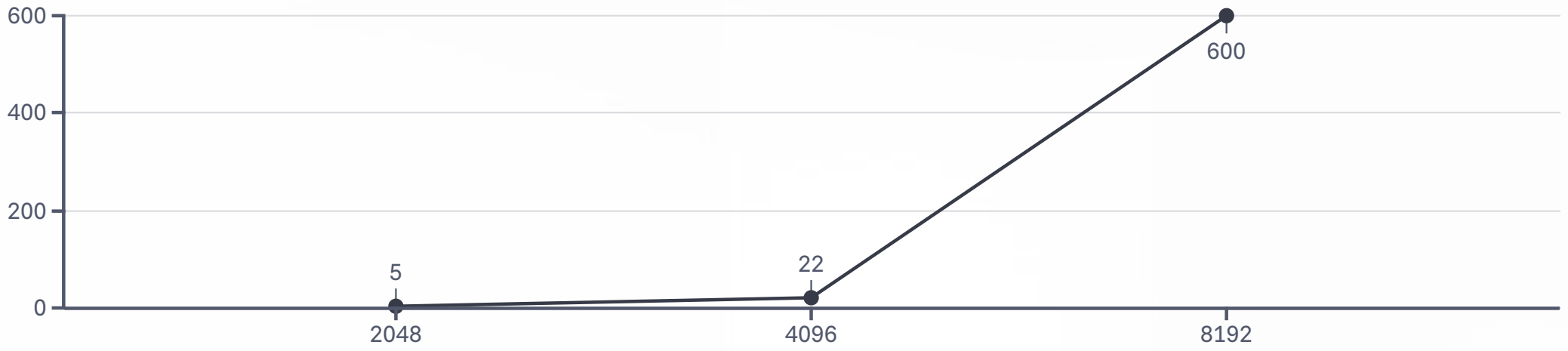
Cenário 3

8.000 quadros por segundo por hosts ou 8.192Mbits/s teóricos

Resultados

- Colisões: **2163**
- Média de colisões/pacote: **0,30**
- Taxa de falhas: **0%**
- Média de *delay* dos quadros: **600µs**

Delay



Colisões