

## Trabalho 1 – Lab. Redes

**Integrantes: Luca Manfroi e Lucas Weiss**

### DeviceUDP.java

#### Descrição:

Classe principal que representa o dispositivo na rede. É responsável por gerenciar a comunicação UDP, escutando pacotes, identificando seus tipos (HEARTBEAT, TALK, FILE, CHUNK, END, ACK, etc.) e respondendo de acordo. Ela funciona como o “core” do sistema, lidando com a recepção e roteamento das mensagens para as classes auxiliares.

#### Objetivo principal:

Gerenciar toda a lógica de comunicação de rede, garantindo que os comandos e transferências sejam corretamente processados e encaminhados.

#### Principais métodos:

- `processarPacote(DatagramPacket packet)`: Método que faz o parsing das mensagens recebidas e decide a ação (ex: TALK, FILE, ACK...).
  - `enviarAck(String id)`: Envia um ACK para confirmar o recebimento de pacotes importantes.
  - **Outros**: Trechos que lidam com a atualização de lista de dispositivos, controle de pacotes desconhecidos e logs.
- 

### ComandoCLI.java

#### Descrição:

Classe que trata a interface de linha de comando (CLI). Monitora os comandos digitados pelo usuário e executa ações como listar dispositivos ativos, enviar mensagens (talk) ou iniciar envio de arquivos (sendfile).

#### Objetivo principal:

Permitir a interação do usuário com o sistema através de comandos simples, integrando com as demais classes.

#### Principais métodos:

- `run()`: Loop principal que lê e interpreta os comandos digitados.
  - `listarDispositivos()`: Mostra a lista atualizada de dispositivos ativos.
  - `talk(String destino, String mensagem)`: Envia uma mensagem de texto para o dispositivo especificado.
  - `sendfile(String destino, String caminhoArquivo)`: Dispara a thread de envio de arquivo usando a classe `FileSender`.
-

## HeartbeatSender.java

### Descrição:

Thread responsável por enviar periodicamente pacotes do tipo HEARTBEAT em broadcast, permitindo que outros dispositivos saibam que este está ativo.

### Objetivo principal:

Manter a presença do dispositivo na rede, garantindo que ele apareça como ativo para os outros participantes.

### Principais métodos:

- `run()`: Envia a mensagem "HEARTBEAT [nome]" a cada 5 segundos continuamente enquanto a aplicação estiver rodando.

---

## LimpezaDispositivos.java

### Descrição:

Thread que verifica periodicamente a lista de dispositivos ativos e remove aqueles que não enviaram HEARTBEAT nos últimos 10 segundos.

### Objetivo principal:

Garantir que a lista de dispositivos seja sempre atualizada e contenha apenas dispositivos realmente ativos.

### Principais métodos:

- `run()`: Loop que verifica timestamps dos dispositivos e remove os que excederam o limite de inatividade.

---

## FileSender.java

### Descrição:

Classe que implementa a lógica de envio confiável de arquivos, dividindo o arquivo em blocos (CHUNKs), enviando cada um deles e aguardando ACK para confirmar a recepção. Implementa lógica de reenvio em caso de timeout/falha.

### Objetivo principal:

Realizar a transmissão confiável de arquivos inteiros por UDP, garantindo que todos os blocos sejam entregues e confirmados.

### Principais métodos:

- `run()`: Gerencia todo o fluxo de envio do arquivo (FILE, CHUNKs e END).
- `enviarEConfirmar(String mensagem, String id)`: Envia um pacote e aguarda o ACK correspondente, com timeout e reenvio automático.
- `bytesToHex(byte[] bytes)`: Converte o hash SHA-256 em string hexadecimal para envio na mensagem END.

---

## FileReceiver.java

**Descrição:**

Classe que trata a recepção dos arquivos enviados. Monta o arquivo a partir dos CHUNKs recebidos, envia os ACKs e, no final, verifica a integridade comparando o hash SHA-256 recebido com o hash calculado.

**Objetivo principal:**

Reconstruir corretamente os arquivos recebidos, lidar com duplicação e fora de ordem (armazenando via TreeMap) e garantir a integridade final do arquivo.

**Principais métodos:**

- receberFile(String id, String nomeArquivo, long tamanho): Inicializa a recepção do arquivo.
- receberChunk(String id, int seq, String dadosBase64): Armazena cada CHUNK, garantindo que duplicatas sejam descartadas.
- receberEnd(String id, String hashEsperado): Finaliza o arquivo e verifica o hash de integridade, imprimindo sucesso ou falha.

---

**Dispositivo.java****Descrição:**

Classe modelo que representa um dispositivo ativo na rede, armazenando informações como IP, nome e o timestamp da última vez que enviou HEARTBEAT.

**Objetivo principal:**

Modelar um dispositivo para ser usado nas listas gerenciadas pelo sistema (ex: lista de dispositivos ativos).

**Principais atributos/métodos:**

- String nome: Nome amigável do dispositivo.
- String ip: Endereço IP do dispositivo.
- long ultimoHeartbeat: Timestamp do último HEARTBEAT recebido.
- Getters e setters simples.

**Funções:****- Devices.**

Exibe a lista de dispositivos atualmente ativos na rede.

Para cada dispositivo, devem ser mostradas as seguintes informações:

```
devices
Dispositivos ativos:
LucaWin - IP: 192.168.0.151 (último heartbeat há 2s)
LucaMac - IP: 192.168.0.224 (último heartbeat há 0s)
```

### - Talk .

Envia uma mensagem de texto para o dispositivo com o nome especificado.

Device1:

```
> talk LucaMac teste talk
Mensagem enviada para LucaMac
> Recebido: ACK TALK
```

Device1:

```
Recebido TALK: TALK teste talk
```

### - Sendfile .

Inicia a transferência de um arquivo para o dispositivo com o nome indicado.

```
Uso: sendfile <nome> <nome-arquivo>
> sendfile LucaMac arquivoteste.txt
> Enviando: FILE e3fa622d-1fab-405c-b93b-098a111ff129 arquivoteste.txt 27
ACK recebido para ID: e3fa622d-1fab-405c-b93b-098a111ff129
Enviando: CHUNK e3fa622d-1fab-405c-b93b-098a111ff129 0 VGVzdGFuZG8gZW52YW8gZGUgYXJxdWl2b3Mu
ACK recebido para ID: e3fa622d-1fab-405c-b93b-098a111ff129
Enviando: END e3fa622d-1fab-405c-b93b-098a111ff129 23d47b3af467cfd78d8979d9d88948d602b37eae98d7c12907fab016182416f7
ACK recebido para ID: e3fa622d-1fab-405c-b93b-098a111ff129
Arquivo enviado com sucesso!
```

```
Recebido FILE: FILE e3fa622d-1fab-405c-b93b-098a111ff1
29 arquivoteste.txt 27
Recebido CHUNK seq=0
Chunk 0 recebido (27 bytes)
Hash esperado: 23d47b3af467cfd78d8979d9d88948d602b37eae98d7c12907fab016182416f7
Hash calculado: 23d47b3af467cfd78d8979d9d88948d602b37eae98d7c12907fab016182416f7
Arquivo arquivoteste.txt recebido com sucesso!
```

### Testes de Integridade:

#### Pacote TALK sem interferência:

Captura do pacote UDP referente ao comando talk enviado do Windows (192.168.0.151) para o Mac (192.168.0.224) na porta 5000. O campo Data mostra claramente o conteúdo da mensagem enviada ("mensagem teste"). Esta captura serve como **base de referência** para comparações posteriores com rede instável simulada (Clumsy).

118283	382.009027	192.168.0.151	192.168.0.224	UDP	61 5000 → 5000 Len=19
118290	382.026798	192.168.0.224	192.168.0.151	UDP	60 5000 → 5000 Len=8[Malformed Packet]

> Frame 118281: 61 bytes on wire (488 bits), 61 bytes captured (488 bits) on interface \Device\NPF\_{1E3A18E8-400F-4517-96A7-7CF5EF2D3701}, id 0

> Ethernet II, Src: ASUSTekCOMPU\_2e:8c:5b (d8:6e:bf:2e:8c:5b), Dst: Apple\_05:ba:cd (fc:e2:6c:05:ba:cd)

> Internet Protocol Version 4, Src: 192.168.0.151, Dst: 192.168.0.224

> 0100 .... = Version: 4

> .... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

> Total Length: 47

> Identification: 0x0bba (40382)

> 0000 .... = Flags: 0x0

> ...0 0000 0000 0000 = Fragment Offset: 0

> Time to Live: 128

> Protocol: UDP (17)

> Header Checksum: 0x0000 [validation disabled]

> [Header checksum status: Unverified]

> Source Address: 192.168.0.151

> Destination Address: 192.168.0.224

> [Stream index: 20]

> User Datagram Protocol, Src Port: 5000, Dst Port: 5000

> Source Port: 5000

> Destination Port: 5000

> Length: 27

> Checksum: 0xd2f4 [unverified]

> [Checksum Status: Unverified]

> [Stream index: 194]

> [Stream Packet Number: 1]

> [Timestamps]

> UDP payload (19 bytes)

> Data (19 bytes)

> Data: 34424c4b20656e730167056d207465737465

> [Length: 19]

0000 fc e2 6c 05 ba cd b0 6e bf 2e 8c 5b 00 00 45 00 ..1.j.n..[.E

0010 00 2f 50 ba 00 00 11 00 00 c0 a8 00 97 c0 a8 ..Y.....

0020 00 00 13 88 13 88 00 1b 82 f4 54 41 4c 4b 20 6d ..E FILE e

0030 65 6e 73 67 67 65 6d 20 74 65 73 74 65 ..e0c0b0-362c-4d6

0040 30 20 62 34 64 30 20 62 34 63 61 33 35 65 36 35 0-b4d0-b 4ca3ee65

0050 34 65 37 20 63 72 71 75 69 76 6f 74 65 73 74 65 4e7 0rqu ivoteste

0060 2e 74 78 74 20 32 37 ..txt 27

Pacotes CHUNK de arquivo sem interferência:

Captura de pacotes UDP durante o envio de arquivo entre Windows (192.168.0.151) para o Mac (192.168.0.224), porta 5000. Mostra os pacotes CHUNK contendo fragmentos do arquivo e o pacote de finalização (END). Essa referência permite comparar a transmissão normal com as condições de rede instável nos testes seguintes.

342970	1345.132390	192.168.0.151	192.168.0.224	UDP	103 53985 → 5000 Len=61
342976	1345.243274	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
342977	1345.250454	192.168.0.151	192.168.0.224	UDP	123 53985 → 5000 Len=81
342983	1345.292905	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
342984	1345.296642	192.168.0.151	192.168.0.224	UDP	147 53985 → 5000 Len=105
342987	1345.345470	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40

> Frame 342970: 103 bytes on wire (824 bits), 103 bytes captured (824 bits) on interface \Device\NPF\_{1E3A18E8-400F-4517-96A7-7CF5EF2D3701}, id 0

> Ethernet II, Src: ASUSTekCOMPU\_2e:8c:5b (d8:6e:bf:2e:8c:5b), Dst: Apple\_05:ba:cd (fc:e2:6c:05:ba:cd)

> Internet Protocol Version 4, Src: 192.168.0.151, Dst: 192.168.0.224

> 0100 .... = Version: 4

> .... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

> Total Length: 89

> Identification: 0xd0dd1 (40401)

> 0000 .... = Flags: 0x0

> ...0 0000 0000 0000 = Fragment Offset: 0

> Time to Live: 128

> Protocol: UDP (17)

> Header Checksum: 0x0000 [validation disabled]

> [Header checksum status: Unverified]

> Source Address: 192.168.0.151

> Destination Address: 192.168.0.224

> [Stream index: 20]

> User Datagram Protocol, Src Port: 53985, Dst Port: 5000

> Source Port: 53985

> Destination Port: 5000

> Length: 69

> Checksum: 0xb31e [unverified]

> [Checksum Status: Unverified]

> [Stream index: 631]

> [Stream Packet Number: 1]

> [Timestamps]

> UDP payload (61 bytes)

> Data (61 bytes)

> Data: 46494c45206565530633030622d333632632d3464363020623463613335653635346537206172717569766f74657374652e747874203237

> [Length: 61]

0000 fc e2 6c 05 ba cd b0 6e bf 2e 8c 5b 00 00 45 00 ..1.j.n..[.E

0010 00 59 90 d1 00 00 00 11 00 00 c0 a8 00 97 c0 a8 ..Y.....

0020 00 00 02 41 13 00 00 45 83 1e 46 49 4c 45 20 65 ..E FILE e

0030 65 65 30 63 30 30 62 2d 33 36 32 63 2d 34 64 36 e0c0b0-362c-4d6

0040 30 20 62 34 64 30 20 62 34 63 61 33 35 65 36 35 0-b4d0-b 4ca3ee65

0050 34 65 37 20 63 72 71 75 69 76 6f 74 65 73 74 65 4e7 0rqu ivoteste

0060 2e 74 78 74 20 32 37 ..txt 27

entrega fora de ordem,

atrasos e pacotes corrompidos 192.168.0.224

Teste Drop 20%:

Configuração do Clumsy aplicando 20% de perda de pacotes no tráfego UDP (porta 5000). No Wireshark, o pacote CHUNK Seq=2 foi perdido e reenviado após timeout, evidenciado por duas capturas do mesmo pacote. O console confirma o recebimento

757975	2844.836390	192.168.0.151	192.168.0.224	UDP	185	53985 → 5000	Len=63
757978	2844.888930	192.168.0.224	192.168.0.151	UDP	82	5000 → 53985	Len=40
757979	2844.893994	192.168.0.151	192.168.0.224	UDP	1455	53985 → 5000	Len=1413
757980	2844.901802	192.168.0.224	192.168.0.151	UDP	82	5000 → 53985	Len=40
757981	2844.903552	192.168.0.151	192.168.0.224	UDP	1455	53985 → 5000	Len=1413
757987	2844.909797	192.168.0.224	192.168.0.151	UDP	82	5000 → 53985	Len=40
758200	2846.928172	192.168.0.151	192.168.0.224	UDP	1455	53985 → 5000	Len=1413
758204	2846.939177	192.168.0.224	192.168.0.151	UDP	82	5000 → 53985	Len=40
758291	2848.268213	192.168.0.151	192.168.0.255	UDP	59	5000 → 5000	Len=17
758292	2848.268303	192.168.0.151	192.168.0.255	UDP	59	5000 → 5000	Len=17
758348	2848.786054	192.168.0.224	192.168.0.255	UDP	60	5000 → 5000	Len=17
758366	2848.939485	192.168.0.151	192.168.0.224	UDP	1455	53985 → 5000	Len=1413
758538	2850.953024	192.168.0.151	192.168.0.224	UDP	1455	53985 → 5000	Len=1413

[illegible]

Configuração do Clumsy aplicando duplicação de pacotes (10% de chance) no tráfego UDP (porta 5000). No Wireshark, o pacote CHUNK Seq=4 foi duplicado, aparecendo duas vezes na captura. O receptor identificou a duplicação utilizando a estrutura TreeMap, que armazena cada CHUNK com base no número de sequência. Antes de gravar, o código verifica se o número já foi recebido (`chunks.containsKey(seq)`) e descarta automaticamente duplicatas. O arquivo foi reconstruído com sucesso e a integridade foi confirmada.

10796_3837.549823	192.168.0.151	192.168.0.224	UDP	104 53985 → 5000 Len=62
10796_3837.560945	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
10796_3837.563459	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
10796_3837.569202	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
10796_3837.571360	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
10796_3837.578416	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
10796_3837.580409	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
10796_3837.588103	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
10796_3837.589350	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
10796_3837.590625	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
10796_3837.590659	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
10796_3837.595202	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
10796_3837.596713	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
10796_3837.598901	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
10796_3837.600162	192.168.0.151	192.168.0.224	UDP	1019 53985 → 5000 Len=977
10796_3837.603690	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
10796_3837.605515	192.168.0.151	192.168.0.224	UDP	147 53985 → 5000 Len=105
10796_3837.605777	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
10796_3837.617111	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40





sejam recebidos, a integridade do arquivo é verificada no final por meio do hash SHA-256, garantindo que arquivos corrompidos sejam detectados e descartados.

14860_	5706.848441	192.168.0.151	192.168.0.224	UDP	104 53985 → 5000 Len=62
14860_	5706.936475	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
14860_	5706.937660	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
14860_	5706.944807	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
14860_	5706.946479	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
14860_	5706.954072	192.168.0.224	192.168.0.151	UDP	82 5000 → 53985 Len=40
14860_	5707.958822	192.168.0.151	192.168.0.255	UDP	59 5000 → 5000 Len=17
14860_	5707.958938	192.168.0.151	192.168.0.255	UDP	59 5000 → 5000 Len=17
14861_	5708.791819	192.168.0.224	192.168.0.255	UDP	60 5000 → 5000 Len=17
14861_	5708.957519	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
14862_	5710.968240	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
14863_	5712.962765	192.168.0.151	192.168.0.255	UDP	59 5000 → 5000 Len=17
14863_	5712.962859	192.168.0.151	192.168.0.255	UDP	59 5000 → 5000 Len=17
14863_	5712.978887	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
14864_	5713.810839	192.168.0.224	192.168.0.255	UDP	60 5000 → 5000 Len=17
14865_	5714.980792	192.168.0.151	192.168.0.224	UDP	1455 53985 → 5000 Len=1413
14867_	5717.971049	192.168.0.151	192.168.0.255	UDP	59 5000 → 5000 Len=17
Internet Protocol Version 4, Src: 192.168.0.151, Dst: 192.168.0.224					0020 00 49 42 41 13 88 05 91 37 04 43 48 55 4e 4b 38 85838f88 -c611-48
0100 .... = Version: 4					0030 38 35 38 33 30 66 38 38 2d 63 36 31 31 2d 34 38 05-85da-71beebdd
.... 0101 = Header Length: 20 bytes (5)					0040 36 35 2d 38 35 64 61 2d 39 31 62 61 05 62 64 64 65-85da-71beebdd
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)					0050 36 37 33 36 08 31 08 61 57 38 67 63 38 4f 6a 62 6736 1 a WgCRDp
Total Length: 1441					0060 79 42 6c 63 33 4e 6c 62 6d 4e 70 59 57 6c 7a 49 y81cNlD mpyMzI
Identification: 0x0ee2 (40674)					0070 48 42 68 63 6d 45 67 64 6d 4e 7a 61 57 52 68 63 h81cNzD mpyMzI
> 000. .... = Flags: 0x0					0080 69 42 77 63 6d 39 38 62 32 4e 7a 62 47 39 7a 48 lbucN8D 2Vp85zI
...0 0000 0000 0000 = Fragment Offset: 0					0090 47 52 6c 49 47 4e 76 62 58 56 75 61 57 4e 68 77 g81cNzD XvM8Mw
Time to Live: 120					00a0 36 66 44 67 32 38 73 49 47 4e 76 62 6d 5a 70 63 6f00d5I 0Vb8MzC
Protocol: UDP (17)					00b0 60 31 68 63 69 42 68 49 48 4a 6c 63 32 6c 73 61 m81cNlD Xj1c1z8
Header Checksum: 0x0000 [validation disabled]					00c0 63 4f 71 62 6d 4e 70 59 53 42 68 49 48 42 6c 63 c0d8Mpy 58f1z81c
[header checksum status: Unverified]					00d0 6d 52 68 63 78 42 6c 49 47 4e 7a 63 32 56 6a 64 m81cNlD 0F1c2z8d
Source Address: 192.168.0.151					00e0 58 4a 68 63 69 42 78 64 57 55 67 62 33 4d 67 58 X81cNzD M8p8MzG
Destination Address: 192.168.0.224					00f0 47 46 6d 62 33 4d 67 63 6d 56 6a 5a 57 4a 70 58 0F0z8MzG mpyMzDpZ
[Stream Index: 20]					0100 47 39 7a 49 47 4e 76 63 6e 4a 6c 63 33 42 76 62 0F1cNzD 0F1c2z8d
User Datagram Protocol, Src Port: 53985, Dst Port: 5000					0110 6d 52 6c 62 53 42 77 58 58 4a 6d 5a 57 6c 30 59 m81058zC X8Mz8Mz
Source Port: 53985					0120 57 31 6c 62 5a 6c 49 47 4e 7a 63 79 42 6d 59 W810MzI 0F1c2z8d
Destination Port: 5000					0130 57 52 76 63 79 42 6c 62 6e 5a 79 59 57 52 76 63 m81cNlD 0F1c2z8d
Length: 1421					0140 79 34 67 51 53 42 77 63 6d 56 6a 61 58 50 44 6f y8058zC mpyMzDpZ
Checksum: 0x2fba [unverified]					0150 32 38 67 62 6d 56 7a 64 47 55 67 6a 47 6c 77 62 28p8MzD 0F1c2z8d
[checksum status: Unverified]					0160 79 42 6d 5a 53 42 30 58 58 4e 30 5a 53 44 44 71 y8K2580Z X8Mz500q
[Stream Index: 631]					0170 53 42 6d 64 57 35 6b 59 57 31 6c 62 66 52 68 62 58Mz8zC W810MzD
[Timestamps]					0180 45 42 77 59 58 4a 68 49 47 52 6c 6a 47 56 6a 64 C8V81z8I 0F1c2z8d
[Time since first frame: 4363.825129000 seconds]					0190 47 46 79 49 47 5a 68 62 47 68 68 63 79 42 76 64 0F1c2z8d 0F1c2z8d
[Time since previous frame: 2.003447000 seconds]					01a0 53 42 70 62 6d 4e 76 62 6e 4e 70 63 33 54 44 71 58p8MzD mpyMzDpZ
UDP payload (1413 bytes)					01b0 6d 35 6a 61 57 46 7a 49 47 52 31 63 6d 46 75 64 m81058zC 0F1c2z8d
Data [..]: 4348554e4b2038353833306638382d633631312d343836352d3931626165626464363733362031206157386763384f6a6279426c63334e6c626d4e7059576c7a49					01c0 47 55 67 59 53 42 30 63 6d 46 75 63 32 31 70 63 0F1c2z8d 0F1c2z8d
Data [..]: 4348554e4b2038353833306638382d633631312d343836352d3931626165626464363733362031206157386763384f6a6279426c63334e6c626d4e7059576c7a49					01d0 33 58 44 6f 32 38 73 49 47 4e 7a 63 32 56 6e 64 3F00d5I 0F1c2z8d
Data [..]: 4348554e4b2038353833306638382d633631312d343836352d3931626165626464363733362031206157386763384f6a6279426c63334e6c626d4e7059576c7a49					01e0 58 4a 68 62 6d 52 76 49 47 45 67 59 32 39 75 58 X810MzI 0F1c2z8d
Data [..]: 4348554e4b2038353833306638382d633631312d343836352d3931626165626464363733362031206157386763384f6a6279426c63334e6c626d4e7059576c7a49					01f0 6d 6c 68 59 6d 6c 73 61 57 52 68 5a 47 55 67 5a m810MzI mpyMzDpZ
Data [..]: 4348554e4b2038353833306638382d633631312d343836352d3931626165626464363733362031206157386763384f6a6279426c63334e6c626d4e7059576c7a49					0200 47 38 67 63 32 6c 7a 64 47 56 7a 59 53 42 70 62 0F1c2z8d 0F1c2z8d
Data [..]: 4348554e4b2038353833306638382d633631312d343836352d3931626165626464363733362031206157386763384f6a6279426c63334e6c626d4e7059576c7a49					0210 58 42 73 5a 57 31 6c 62 6e 52 68 5a 47 38 75 44 X810MzI mpyMzDpZ
Data [..]: 4348554e4b2038353833306638382d633631312d343836352d3931626165626464363733362031206157386763384f6a6279426c63334e6c626d4e7059576c7a49					0220 51 67 7a 49 45 56 7a 64 47 55 67 77 36 6d 67 64 0F1c2z8d 0F1c2z8d
Data [..]: 4348554e4b2038353833306638382d633631312d343836352d3931626165626464363733362031206157386763384f6a6279426c63334e6c626d4e7059576c7a49					0230 57 38 67 64 47 56 7a 64 47 55 67 5a 47 55 67 5a W810MzI 0F1c2z8d
Data [..]: 4348554e4b2038353833306638382d633631312d343836352d3931626165626464363733362031206157386763384f6a6279426c63334e6c626d4e7059576c7a49					0240 32 56 79 59 63 4f 6e 77 36 4e 76 49 47 52 6c 49 2VYCYOm 6VY810I