Alunos: Adriano Rafael Meira Tosetto, Giulio Guilherme de Souza Simão, Gustavo Borges França.

1. Precision Time Protocol

PTP é um é um protocolo utilizado para sincronizar *clocks* em uma rede, alcançando precisões em nanosegundos e, por vezes, picosegundos.

O PTP foi originalmente definido no padrão IEEE 1588-2002, publicado em 2002, e em 2008 foi definido o padrão 1588-2008, conhecido como PTP versão 2, nele foi melhorado aspectos de precisão e robustez do protocolo.

O IEEE1588 descreve uma hierarquia mestre-escravo para a distribuição dos *clocks*, e dentro dessa hierarquia, existe(m) segmento(s) de rede que são o meio de comunicação dos relógios, e um ou mais relógios.

2. Protocolo

Existem quatro tipos de relógio: *grandmaster*, *ordinary*, v2(*transparent*), *boundary*:

- *Grandmaster* é o *clock* “raíz”, ele que vai transmitir informações de sincronização para os *clocks* no seu segmento de rede.
- *Ordinary clocks* são *clocks* que ou são escravos ou são mestres em um segmento de rede, estes têm apenas uma porta conexão.
- *Boundary clocks* são *clocks* que tem várias conexões e estes podem sincronizar outros segmentos de rede.
- *Transparent clock* é um dispositivo que mede o tempo que uma mensagem PTP levou para transitar e disponibiliza essa informação para *clocks* recebendo a mensagem.

Além disso, suas características são:

- PTP é baseado no TAI (*International atomic time*)
- *Unix time* é baseado em UTC
- O PTP geralmente usa o mesmo *Epoch* que o Unix
    - Por conta dos padrões de tempo serem diferentes, o *grandmaster* do PTP comunica o *offset* entre o UTC e o TAI, dessa forma UTC pode ser computado do tempo recebido pelo PTP.

Tipos de mensagens:

- Mensagens de evento (*Timestamped*)
    - *SYNC*
    - *DELAY_REQ*
    - *PDELAY_REQ* (V2)
    - *PDELAY_RES* (V2)
- Mensagens gerais (*Not timestamped*)
    - *FOLLOW_UP*
    - *DELAY_RESP*
    - *PDELAY_RESP_FOLLOW_UP* (V2)
    - *ANNOUNCE* (V2)

    - *SIGNALING*(V2)
    - *MANAGEMENT*

As diferentes versões do PTP e tipos de mestre possuem mensagens características:

- *SYNC*, *FOLLOW_UP*, *DELAY_REQ* e *DELAY_RESP* são utilizadas pelos *boundary* e *ordinary clocks* para comunicar informações relacionadas ao tempo e sincronizar os *clocks* da rede.
    - *One step clock*: O *timestamp* do mestre vai na mensagem de SYN;
    - *Two step clock*: O *timestamp* do mestre é enviado na mensagem *FOLLOW_UP*;
- *PDELAY_REQ*, *PDELAY_RES* e *PDELAY_RES_FOLLOW_UP* são usados por *transparent clocks* (Não existem na versão 1)
- *SIGNALING* e *ANNOUNCE* estão presentes também na versão 2 do PTP
    - *SIGNALING* para comunicações na rede que não são dependentes da precisão do tempo.
    - *ANNOUNCE* serve auxiliar na construção da hierarquia e seleção do *Grandmaster clock*
- *MANAGEMENT* é utilizada para monitorar e configurar um sistema PTP

O tempo origem do PTP é 1º de janeiro de 1970, 00h00min00seg TAI, que corresponde a 31 de dezembro de 1969, 23h59min51,999918seg UTC. É possível calcular o tempo UTC usando os valores *timePropertiesDS.currentUtcOffset*, que corresponde a TAI - UTC. Dentro de um domínio o tempo será medido como o tempo decorrido desde o tempo origem.

O timestamp representa um tempo positivo relativo ao epoch,

```
struct Timestamp {
        UInteger48 secondsField;
         UInteger32 nanosecondsField;
};
```

O atributo *secondsField* é a parte de segundos do timestamp, e o *nanosecondsField* é a parte fracionária do timestamp, que é a unidade em nanosegundos. O *nanosecondsField* é sempre menor que 10^-9

2.1 Hierarquia mestre-escravo

Cada *clock* pode possuir uma ou mais portas. Uma porta pode se ligar com uma de outro *clock*, em que uma assume o papel de mestre e a outra de escravo. As portas do relógio *grandmaster* são sempre mestre, e aquelas às quais se ligam serão sempre escravo. As outras portas dos *clocks* que se ligam ao *grandmaster* serão mestres e se ligarão a portas escravo e assim por diante. Se uma porta se ligaria a outra de um *clock* já pertencente à rede, elas ficarão em estado passivo.

O *grandmaster* estabelece comunicação com seus vizinhos para sincronizá-los de acordo com seu tempo. Seus vizinhos, então, fazem o mesmo com os *clocks* que estão ligados às suas portas mestre, até que a rede esteja sincronizada com o *grandmaster*.

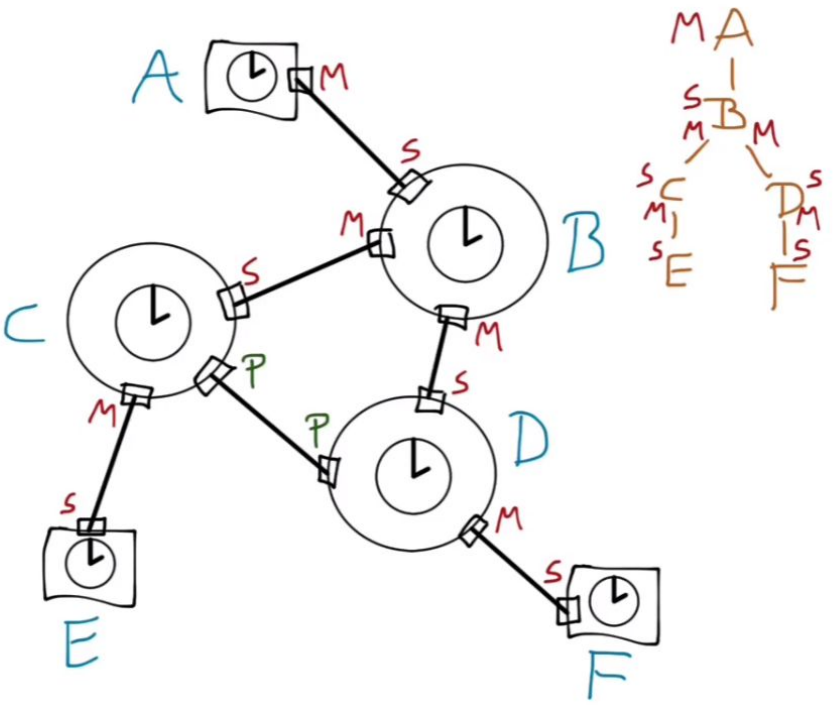

Fig 1. Hierarquia mestre escravo.

2.2 Diferenças em relação ao NTP

O principal na hora de se comparar é a precisão. O NTP é preciso a casa do milisegundo, já o PTP consegue atingir precisão na casa dos nanosegundos. O motivo disso é que o NTP depende primariamente de algoritmos em *software* para processar o tempo. Já o PTP apesar de ainda necessitar de algoritmos em *software*, ele utiliza de *hardware* para realizar o *timestamping*.

Outro ponto de diferença, é que o NTP pode ser implementado na rede *Ethernet* padrão, já o PTP necessita de uma infraestrutura que siga o padrão IEEE 1588

3. Formato das Mensagens

Mensagens PTP podem ser transportadas em vários protocolos, eles são:

3.1 Formato sobre IPV4/IPV6

Quando transportando sobre UDP, o primeiro *byte* da mensagem PTP segue imediatamente o *byte* final do *header* UDP. O campo destinado para a porta identifica o datagrama UDP como uma mensagem PTP.

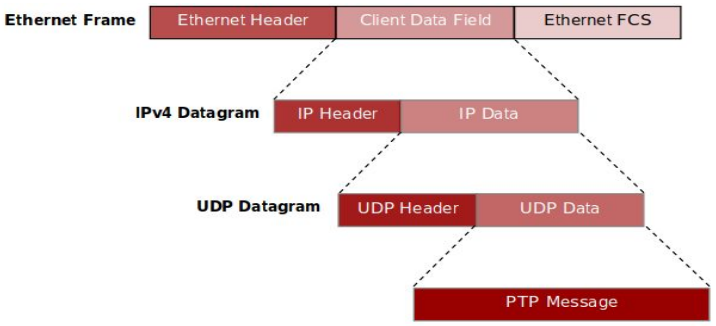

Fig2. Campos do formato sobre IPV4

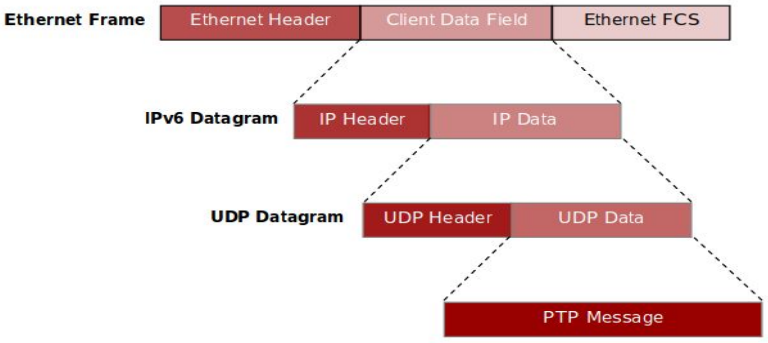

Fig3. Campos do formato sobre IPV6

3.2 IEEE 802.3/Ethernet

Quando carregado sobre a *ethernet*, o primeiro *byte* da mensagem PTP ocupa o primeiro *byte* do campo de dados do cliente do *frame ethernet*. O campo de tipo *Ethernet* é setado para 0x88F7 e identifica o campo do cliente como uma mensagem PTP.

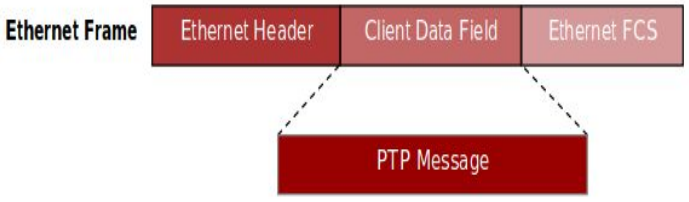

Fig 4. PTP sob a Ethernet

3.3 Formato das Mensagens PTP

Todas as mensagens PTP consistem de um *header*, *body* e um sufixo (opcional).

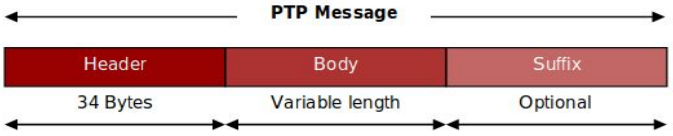

Fig 5. Formato das mensagens PTP

3.3.1 Formato do *Header*

O *header* é parte comum a todas mensagens PTP e possui 34 bytes. Seus campos são:

- *messageType*: define qual tipo de mensagem está contida no campo *Body*. Por exemplo: *Sync*, *Delay_Req* e etc;
- *messageLength*: determina o tamanho da mensagem toda em *byte* incluindo o *header*, *body* e sufixo, mas não inclui *padding;*
- *domainNumber*: identifica o domínio o qual a mensagem PTP pertence;
- *flags*: contém várias *flags* que indicam *status;*
- *correctionField*: valor de correção em nanosegundos (*transparent clock*);
- *sourcePortIdentity*: campo que identifica a porta originária da mensagem;
- *sequenceId*: um número sequencial para tipos de mensagens individuais;
- *controlField*: campo legado. *messageType* com menos opções;
- *logMessageInterval*: esse campo é determinado pelo tipo da mensagem;

PTP Message Header Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
Flags								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceID								2	30
controlField								1	32
logMessageInterval								1	33

Fig 6. Formato do cabeçalho das mensagens PTP

3.3.2 Formato do *Body*

3.3.2.1 Mensagem *Announce*

- *originTimestamp*
- *currentUtcOffset*
- *reserved*
- *grandmasterPriority1*
- *grandmasterQuality*
- *grandmasterPriority2*
- *grandMasterIdentity*
- *stepsRemoved*
- *timeSource*

Announce Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
originTimestamp								10	34
currentUtcOffset								2	44
Reserved								1	46
grandmasterPriority1								1	47
grandmasterClockQuality								4	48
grandmasterPriority2								1	52
grandmasterIdentity								8	53
stepsRemoved								2	61
timeSource								1	63

Fig 7. Formato da mensagem *Announce*

3.3.2.2 Mensagem *Sync*, *Delay_Req*

- *originTimestamp*
  - tempo do mestre quando a mensagem foi enviada ao escravo (*Sync*)
  - tempo do escravo quando a mensagem foi enviada ao mestre (*Delay_Req*)

Sync Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
originTimestamp								10	34

Fig 8. Formato da mensagem *Format*

3.3.2.3 Mensagem *Follow_Up*

- *preciseOriginTimestamp* - tempo do mestre quando o *Sync* foi enviado

Follow_Up Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
preciseOriginTimestamp								10	34

Fig 9. Formato da mensagem *Follow_Up*

3.3.2.4 Mensagem *Delay_Resp*, *Pdelay_Resp*

- *receiveTimestamp* - o tempo do mestre quando o *Delay_Req* foi recebido
- *requestingPortIdentity* - a porta de quem enviou o *Delay_Req*

Delay_Resp Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
receiveTimestamp								10	34
requestingPortIdentity								10	44

Fig10. Formato da mensagem *Delay_Resp*

3.3.2.5 *Pdelay_Req*

- *originTimestamp* - tempo do requester quando a mensagem é enviada
- *reserved* - serve para deixar a mensagem do mesmo tamanho que *Pdelay_Resp*

Pdelay_Req Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
originTimestamp								10	34
reserved								10	44

Fig11. Formato da mensagem *Pdelay_Req*

3.3.2.6 *Pdelay_Resp*, *Pdelay_Resp_Follow_Up*

- *responseOriginTimestamp*
    - tempo do responder quando o *Pdelay_Req* é recebido (*Resp*)

    - tempo do responder quando o *Pdelay_Resp* é enviado (*Follow_Up*)
- *requestingPortIdentity* - porta do requester

Pdelay_Resp_Follow_Up Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
responseOriginTimestamp								10	34
requestingPortIdentity								10	44

Fig 12. Formato da mensagem *Pdelay_Resp_Follow_Up*

3.3.2.7 *Signalling*

- *targetPortIdentity* - endereço da(s) porta(s) destino da mensagem;
- *TLV*s - identificadores *Type*, *Length* e *Value*;

Signalling Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
targetPortIdentity								10	34
One or more TLVs								N	44

Fig 13. Formato da mensagem *Signalling*

3.3.2.7 *Management*

- *targetPortIdentity* - endereço da(s) porta(s) destino da mensagem;
- *startingBoundaryHops* - número de relógios *boundary* que podem retransmitir a mensagem;
- *boundaryHops* - número de retransmissões restantes;
- *actionField* - tipo de ação que a mensagem deve tomar;
- *managementTLV* - contém outros quatro campos:
    - *tlvType* - será setado para MANAGEMENT (0x0001);
    - *lengthField* - tamanho do TLV
    - *managementID* - tipo da mensagem de *management*;
    - *dataField* - dependente do *managementID*;

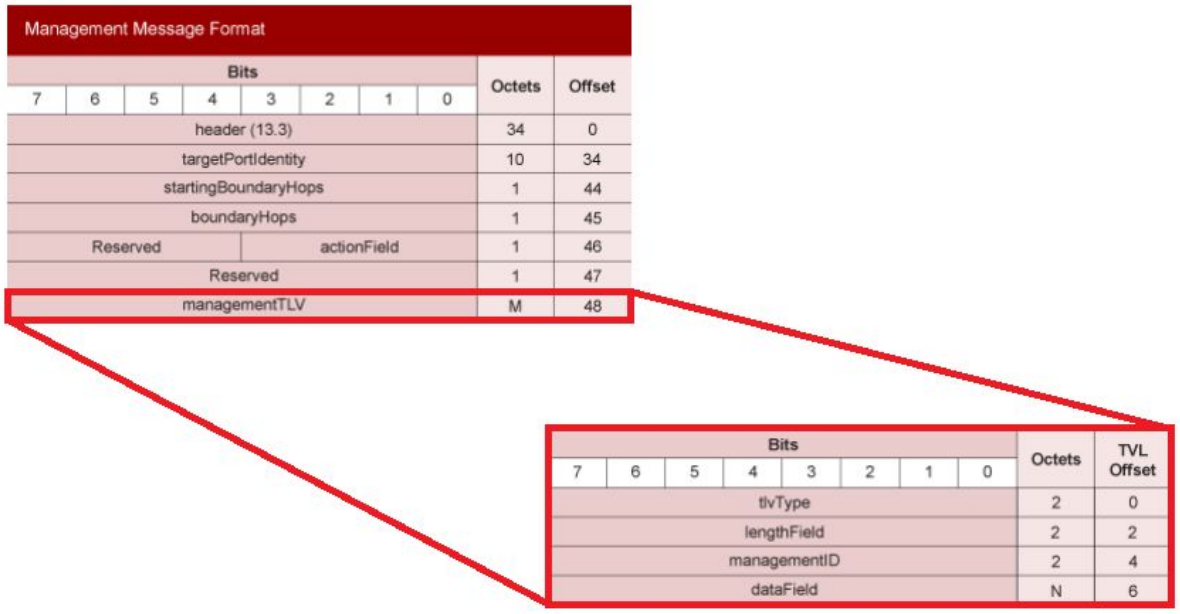

Fig 14. Formato da mensagem *Management*

4. Funcionamento

Para sincronizar o escravo, o mestre manda uma mensagem de *sync*, e ao enviar essa mensagem, ele guarda localmente o *timestamp* no qual a mensagem foi enviada. Ao receber a mensagem de sincronização, o escravo então guarda localmente o seu próprio *timestamp*. Ao receber a mensagem de sincronização, o escravo então guarda localmente o seu próprio *timestamp*.

O mestre então prossegue enviando ao escravo uma mensagem *FOLLOW_UP* contendo o *timestamp* t1. Ao receber a mensagem de *FOLLOW_UP* o escravo agora conhece o t1 e o t2 (que é o seu próprio *timestamp*).

Agora o escravo manda ao mestre uma mensagem *DELAY_REQ*, e ao enviar, ele guarda seu *timestamp* t3. O mestre ao receber guarda em si o seu próprio *timestamp* t4. Depois mestre envia ao escravo uma mensagem de *DELAY_RES* com o t4.

Calculando o atraso (considerando que o atraso entre o mestre e o escravo é o mesmo do escravo para o mestre):

$$PD = ((T2 - T1) + (T4 - T3)) \div 2$$

$$OFFSET = (T2 - T1) - PD$$

$$Clock_{novo} = Clock_{velho} - OFFSET$$

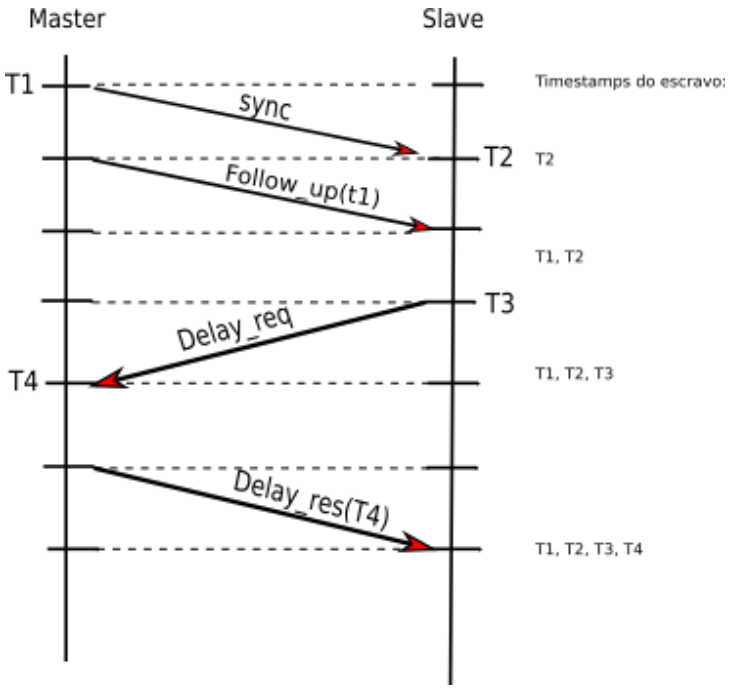

Fig 15. Troca de mensagens entre o mestre e o escravo.

5. Problemas do Mundo Real

- Variação no atraso de rede
    - A variação da rede (transmissão mestre-escravo escravo-mestre) pode causar problemas na precisão do PTP
- Influência da arquitetura sob a qual o PTP está rodando.
    - Algo que é ignorado mas pode ser impactante no PTP é as operações da arquitetura na qual o PTP roda.
    - Exemplo: Não encontramos PTP em SOs comuns pois, além de outras coisas, a precisão que seria obtida seria algo próximo ao NTP, devido ao não determinismo de operações do SO.
- Variação dos clocks
    - Outro problema também é os clocks terem frequências diferentes.

A figura 16 mostra o resultado do teste realizado(código disponível no github[1]), após a troca de mensagens para calcular o offset, foram testados 200 timestamps para verificar a discrepância entre os timestamps. Por essa imagem é possível ver que como o código estava sendo executado em um computador com um SO linux, pelo não determinismo existente de saber quanto tempo o SO vai realmente “lançar” o timestamp, é possível ver uma variação do erro do *timestamp* do mestre para o *timestamp* do escravo

[1] http://github.com/gstvob/SO2

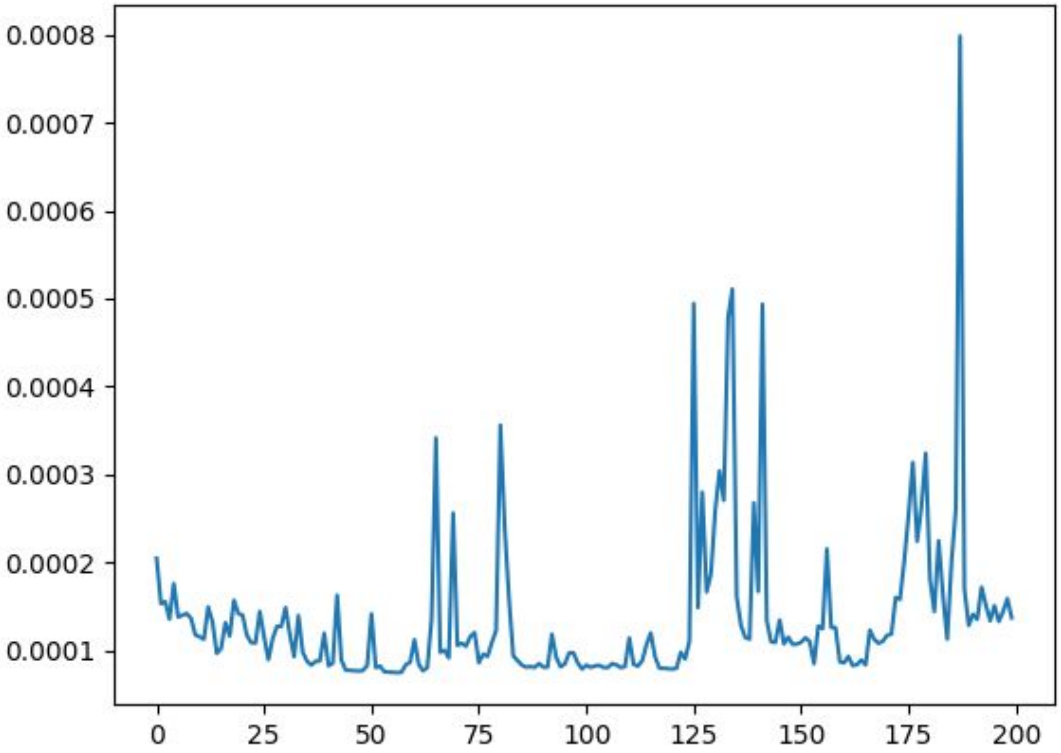

Fig16. Gráfico dos erros.(X=Número do erro, Y=Quantidade)