Diagramme du projet

Légende : s : statique
f : final
p : private

Trame

- s ListeTrame > lsTrames
- f int id
- f String (entete
- f String lastProtocol
- f boolean refuse

constructeur Trame(String trame)

Protocoles

- s Map<id, Infos> ethernet
- s Map<id, Infos> ipv4
- s Map<id, Infos> tcp
- s Map<id, Infos> http

Infos

- f String type (∈ {Ethernet, ...})
- Map<String, String> hashInfos

- - - - - -

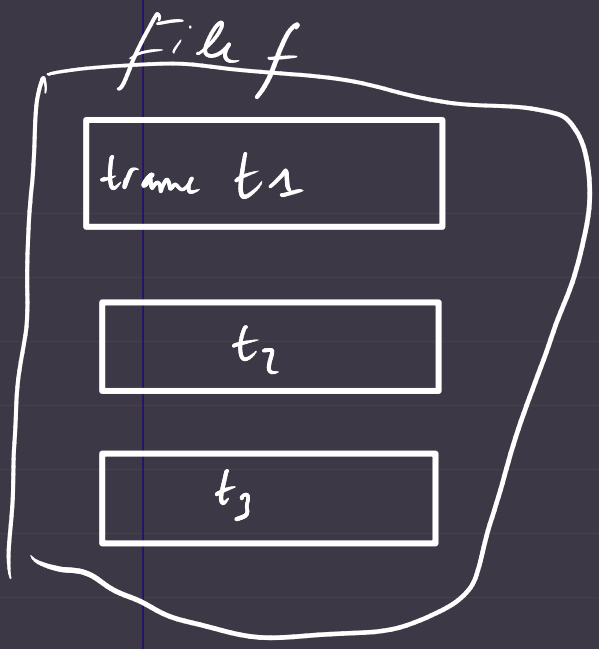
- void hashEthernet()
- void hashIpv4()
- void hashHttp()
- void hashTcp()

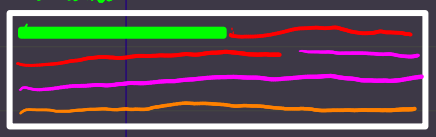
- p void setInfos(String field, String data)
- void parseHeaderIpv4(String header)
- void parseHeaderHttp(String header)
- void parseHeaderTcp(String header)
- void parseHeaderEthernet(String header)
- String getInfo(String field)
- String getType();

Parser

void splitInput(File f)
↳ split l'input et crée toutes les trames grâce au constructeur de Trames

void observateurProtocolX
⋮
void parserProtocolX





- observateurProtocoleX
  - Input : { - texte restant
    - index du début de l'entête à analyser
  - return { nextProtocol (peut être nulle)
    taille de l'entête

- traitementProtocol X
  - Input : .....
  - Ajoute les infos extraites de l'entête en cours à la classe Protocols
  - set le dernier protocole de la trame à X

Traitement de trame t :

→ récupérer Id

→ récupérer content

→ observateur Ethernet

↳ récupérer next et length

↳ si next ∉ {null, ipv4} → - Refuser la trame
- s'arrêter

→ traitement Ethernet

→ si next = ipv4 : appeller observateur IPv4

etc ...