VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

XGPON MODUL PRO WIRESHARK

XGPON MODULE FOR WIRESHARK

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Patrik Šuba

AUTHOR

VEDOUCÍ PRÁCE — Ing. Adrián Tomašov

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Patrik Šuba ***ID:*** 213135

Ročník: 2 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

XGPON modul pro Wireshark

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit modul/plugin pro aplikaci Wireshark, který dokáže zpracovat data z pasivních optických sítí XG-PON. V práci student navrhne formát PCAPNG souboru pro podporu XG-PON a prozkoumá způsoby tvorby modulu pro aplikaci Wireshark. Vytvoří prototyp modulu pro XG-PON síť v sestupném směru. V práci bude také implementován i vzestupný směr a finalizován XG-PON modul. Funkčnost modulu bude ověřena na reálném vzorku dat ze sítě XG-PON.

DOPORUČENÁ LITERATURA:

[1] LAMPING, Ulf; ONTANON, Luis E.; BLOICE, Graham. Wireshark developer's guide. 2014-11-09)[2016-04-25]. https://www. wireshark. org/docs/wsug_html_chunked, 2004.

[2] MEMON, Kamran A., et al. Dynamic bandwidth allocation algorithm with demand forecasting mechanism for bandwidth allocations in 10-gigabit-capable passive optical network. Optik, 2019, 183: 1032-1042.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 21.5.2024

Vedoucí práce: Ing. Adrián Tomašov

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Táto diplomová práca sa zaoberá návrhom súboru s formátom PcapNG pre podporu XG-PON dát a preskúmava možnosti tvorby modulu. Cieľom je vytvoriť modul ako technický nástroj pre detailnú analýzu a diagnostiku XG-PON sietí. Návrh súboru pozostáva zo spracovania paketu sieťového analyzátora a vytvorenia blokov PcapNG formátu. Tento návrh súboru je implementovaný v programe, ktorý je napísaný v jazyku Python. Pre tvorbu Wireshark modulu je použitý programovací jazyk LUA s natívnou podporou využitia novej linkovej vrstvy. Použitie modulu pozostáva z potrebného nastavenia aplikácie pre podporu protokolu XG-PON a využití samotného modulu, ktorý dokáže spracovať XG-PON rámce v oboch smeroch komunikácie v navrhnutom PcapNG súbore. Dôležitou súčasťou bola verifikácia funkčnosti modulu na dátach z dostupnej XG-PON siete, čo demonštrovalo schopnosť správne spracovávať dáta z reálneho prostredia. Výsledkom práce je nástroj pre analýzu a diagnostiku XG-PON sietí, ktorý dokáže prispieť k lepšiemu riadeniu a správe týchto moderných optických sietí.

KĽÚČOVÉ SLOVÁ

HEC, LUA, PLOAM, PcapNG, Plugin, Python, SDU, Wireshark, XGEM, XGTC, XG-PON

ABSTRACT

The master's thesis deals with the design of a PcapNG file format for supporting XG-PON data and explores the possibilities of creating a module. The aim is to create a module as a technical tool for detailed analysis and diagnostics of XG-PON networks. The file design proposal consists of packet processing by a network analyzer and the creation of PcapNG format blocks. This file design proposal is implemented in a program written in the Python language. The Wireshark module is implemented in the LUA programming language which have native support for utilizing the new link layer. The use of the module involves configuring the application to support the XG-PON protocol and utilizing the module itself, which can process XG-PON frames in both directions of communication in the proposed PcapNG file. An important part was the verification of the module's functionality on data from an available XG-PON network, demonstrating its ability to correctly process data from a real environment. The result of the work is a tool for the analysis and diagnosis of XG-PON networks, which can contribute to better management of these modern optical networks.

KEYWORDS

Dissector, HEC, LUA, PLOAM, PcapNG, Python, SDU, Wireshark, XGEM, XGTC, XG-PON

ŠUBA, Patrik. *XGPON modul pro Wireshark*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024, 68 s. Diplomová práca. Vedúci práce: Ing. Adrián Tomašov

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Bc. Patrik Šuba
VUT ID autora:	213135
Typ práce:	Diplomová práca
Akademický rok:	2023/24
Téma závěrečnej práce:	XGPON modul pro Wireshark

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi Ing. Adriánovi Tomašovovi, za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci. Zároveň by som chcel poďakovať celej mojej rodine za trpezlivosť a podporu počas štúdia.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

V rámci pasívnych optických sietí sa stále viac presadzuje štandard 10Gb Pasívna Optická Sieť[1], ktorý umožňuje ešte väčšiu priepustnosť a efektivitu siete. Pre správnu analýzu a diagnostiku XG-PON sietí je nevyhnutné mať k dispozícií nástroje, ktoré umožňujú efektívne spracovanie a interpretáciu dát.[2]

Cieľom tejto práce je navrhnúť a implementovať modul/plugin pre aplikáciu Wireshark[3], ktorý umožní spracovanie dát z pasívnych optických sietí XG-PON. Práca sa zameriava na návrh formátu PcapNG[4] súboru pre podporu XG-PON a skúmanie metód tvorby modulu pre aplikáciu Wireshark. Vytvorenie prototypu modulu pre XG-PON sieť v zostupnom smere predstavuje prvý krok v procese implementácie. Následne bola prevedená implementácia vzostupného smeru a finálne vyladenie modulu.

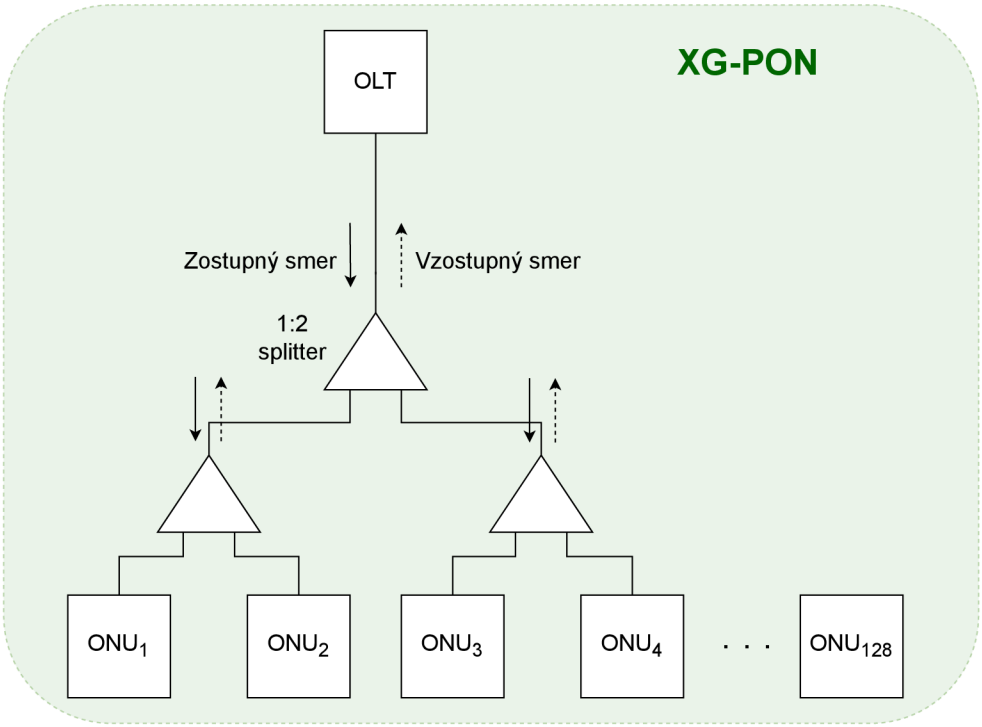
Význam práce spočíva v poskytnutí nástroja, ktorý umožní inžinierom vykonávať detailnú analýzu a diagnostiku XG-PON sietí pomocou široko využívaného nástroja Wireshark. Funkčnosť vytvoreného modulu bude overená na vzorku dát z reálnej sieti, čo poskytne dôkaz jeho schopnosti korektne spracovať dáta v reálnej prevádzke.

V práci je detailne spracovaná štruktúra XG-PON rámcov v zostupnom aj vzostupnom smere. Zároveň obsahuje spracovanie paketu dát, ktorý je vytvorený pomocou použitého sieťového analyzátoru pre zachytávanie prevádzky. Bol spracovaný obsah potrebných blok PcapNG pre správnu identifikáciu formátu aplikáciou Wireshark. Práca obsahuje použitie hybridnej chybovej korekcie, ktorá je prítomná vo viacerých štruktúrach XG-PON protokolu.

Táto práca sa ďalej skladá z nasledujúcich častí: v prvej kapitole sú predstavené základné princípy XG-PON sietí, vrátane štruktúry rámcov, štruktúry paketu a ich použitia. Kapitola 2 sa venuje štruktúre blokov PcapNG formátu a verejne dostupnej knižnici pre vytváranie a prácu s PcapNG súbormi. Formát PcapNG je možné využiť pre zobrazenie sieťového dátového toku v aplikácií Wireshark. V druhej časti preskúmava možnosti vytvorenia Wireshark modulu, ich dostupnosť a porovnanie. Kapitola 3 je zameraná na riešenie problematiky vytvorenia súboru s formátom PcapNG. Popisuje implementáciu riešenia v programovacom jazyku Python a riešenie problémov, ktoré vznikli pri jeho tvorbe. Taktiež popisuje potrebné spracovanie binárnych dát pre získanie čistých XG-PON dát. Kapitola 4 obsahuje popis návrhu a implementácie modulu pre program Wireshark. Obsahuje samotnú problematiku vytvorenia modulu a taktiež možnosť využitia modulu pre vývoj protokolu linkovej vrstvy. V kapitole 5 je popísaná verifikácia a validácia implementovaného modulu na reálnych dátach. Zároveň je konkrétne popísaná použitá XG-PON sieť a jej zapojenie. Nakoniec sú v kapitole 6 zhrnuté dosiahnuté výsledky práce s názornou ukážkou zobrazených dát v aplikácií Wireshark.

1 10Gb Pasívna Optická Sieť (XG-PON)

Táto kapitola popisuje prenosovú konvergenčnú vrstvu pre *10Gb Pasívna Optická Sieť* (XG-PON) systémy[5], ktoré operujú nad point-to-multipoint optickou prístupovou infraštruktúrou s nominálnou rýchlosťou prenosu dát 9,95328 Gbit/s pre zostupný smer, a štvrtinovou nominálnou rýchlosťou 2,48832 Gbit/s pre vzostupný smer.[1] Point-to-multipoint infraštruktúra pozostáva z jedného optického sieťového terminálu *Optický sieťový terminál* (OLT) a až 128 optických sieťových jednotiek *Optické sieťové jednotky* (ONUs). Infraštruktúra poskytuje široký rozsah širokopásmových a úzko-pásmových služieb pre koncového zákazníka. Smer prenosu dát od OLT (poskytovateľa) k ONU (užívateľovi) je nazývaný zostupný smer. Opačný smer od ONU k OLT je nazývaný vzostupný smer. Signál v zostupnom smere prichádza vždy ku všetkým ONU v sieti. Vzostupný smer prechádza cez splittery a prechádza len smerom k OLT a k ostatným ONU sa nedostane. Zostupný smer využíva technológiu vlnového multiplexovania *Dense wavelength-division multiplexing* (DWDM) a vzostupný smer využíva technológiu multiplexovania s časovým delením *Time-division multiplexing* (TDM). [6]



Obr. 1.1: XG-PON sieť.

[1]Existuje aj symetrická varianta, nazvaná XGS-PON, ktorá má rýchlosť prenosu dát 9,95328 Gbit/s pre oba smery.

1.1 XGTC vrstva

XG-PON transmission convergence (XGTC) vrstva špecifikuje formáty a procedúry mapovania medzi vrchnou vrstvou, *Servisné Dátové Jednotky* (SDUs) a spodnou vrstvou, bitovými tokmi vhodnými pre moduláciu na optický nosič. XGTC vrstva pozostáva z troch vrstiev a to adaptačnou servisnou pod-vrstvou, pod-vrstvou rámcovania a adaptačnou *Physical interface* (PHY) pod-vrstvou. XGTC vrstva je prítomná v oboch smeroch komunikácie. Pre zostupný smer je rozhranie medzi XGTC vrstvou a vrstvou fyzického média reprezentovaná bitovým tokom pri nominálnej rýchlosti rozhrania, ktorý je rozdelený do rámcov o veľkosti 125 μs. Rýchlosť prenosu zostupného smeru je 9,95328 Gbit/s čo odpovedá veľkosti 155520 bajtov pre jeden rámec. Pre vzostupný smer je rozhranie medzi XGTC vrstvou a vrstvou fyzického média reprezentované sekvenciou presne načasovaných XGTC zhlukov. [7] [5]

Servisné Dátové Jednotky (SDUs)

Servisné dátové jednotky obsahujú enkapsulované dáta. Najčastejšie sú použité pre enkapsulovanie ethernet rámcov ale v teórií môžu byť enkapsulované dáta akékoľvek.

1.1.1 XGTC adaptačná servisná pod-vrstva

XGTC adaptačná servisná pod-vrstva je zodpovedná za enkapsuláciu vrchnej vrstvy SDU, multiplexovanie a vymedzenie počas prenosu v pasívnej optickej sieti.

Na strane vysielača pod-vrstva prijíma SDUs reprezentované rámcami užívateľských dát a ONU manažmentom a riadi prevádzku rozhrania, vykonáva fragmentáciu SDU podľa potreby, priradí *XG-PON encapsulation method* (XGEM) Port-ID k SDU alebo fragmentu a zapuzdri ich do XGEM rámca. Payload XGEM rámca môže byť voliteľne šifrovaný. Séria rámcov XGEM tvorí payload XGTC rámca v zostupnom smere alebo XGTC zhluky vo vzostupnom smere.

Na strane prijímača pod-vrstva prijíma payload tvorený z XGTC rámcov alebo zhlukov, vykonáva vymedzovanie XGEM rámcov, filtruje XGEM rámce podľa Port-ID, dešifruje XGEM payload pri použití šifrovania, zostaví fragmentované SDUs a doručí ich príslušným klientom. [7]

1.1.2 XGTC pod-vrstva rámcovania

XGTC vrstva rámcovania je zodpovedná za vytvorenie a spracovanie hlavičiek, ktoré podporujú potrebný manažment funkčnosti PON. Formát rámcov je navrhnutý aby rámce a ich prvky boli zarovnané na medzu slov o dĺžke 4 bajty, kedykoľvek je to možné.

Na strane vysielača pod-vrstva prijíma sériu XGEM rámcov, ktoré tvoria XGTC payload z XGTC adaptačnej servisnej pod-vrstvy a konštruuje zostupný XGTC rámec alebo XGTC zhluk poskytovaním vložených *Operations Administration and Management* (OAM) a fyzickou vrstvou *Fyzická vrstva operácií, administratívy a údržby* (PLOAM) kanálu pre odosielanie správ hlavičkami. Veľkosť každého XGTC payloadu v zostupnom smere je možné získať odčítaním nákladov premenlivej veľkosti šírky pásma hlavičky údržby vzostupného smeru a záťaže kanálu PLOAM od fixnej veľkosti XGTC rámca zostupného smeru. Pre vzostupný smer XGTC zhluky multiplexujú XGTC payload spojený s viacerými Alloc-IDs, pričom veľkosť každých nákladov je stanovená na základe prichádzajúcich informácií o šírke pásma.

Na strane prijímača vrstva prijíma XGTC rámce alebo XGTC zhluky, parsuje XGTC hlavičky, extrakciou prichádzajúcej vstavanej údržby a tokoch PLOAM správ a doručuje XGTC payload adaptačnej servisnej vrstve. Prichádzajúci tok správ kanálom PLOAM je doručený do motora spracovania PLOAM. Vložené informácie OAM, ktoré sa vzťahujú k správe šírke pásma vzostupného smeru (parsovanie BWmap) a signalizácií dynamickej alokácie šírky pásma, sú spracované pod-vrstvou rámcovania čo poskytuje čiastočnú kontrolu nad adaptačnou pod-vrstvou PHY (načasovanie PHY zhluku a kontrola profilu pre vzostupný smer) a adaptačnou servisnou pod-vrstvou (indikovanie šifrovacieho kľúča). Ostatok vložených OAM informácií je doručených kontrolným prvkom mimo pod-vrstvy rámcovania, ako sú bloky správy a sledovania výkonu ONU. [7]

1.1.3 XGTC adaptačná PHY pod-vrstva

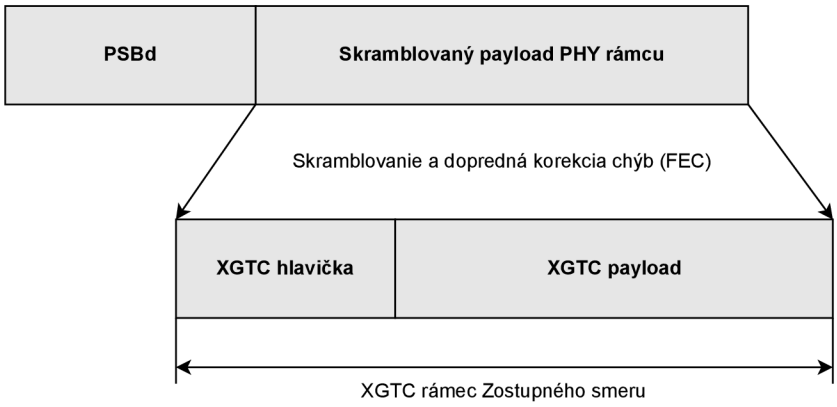
XGTC adaptačná PHY pod-vrstva zahrňuje funkcie, ktoré menia bitový tok modulovaný optickým vysielačom s cieľom zlepšiť vlastnosti detekcie, prijímania a vymedzenia signálu prenášaného cez optické médium.

Na strane vysielača pod-vrstva prijíma XGTC rámce alebo XGTC zhluky z pod-vrstvy rámcovania, rozdeľuje ich na *Popredná chybová korekcia* (FEC) dátové bloky, vypočítava a pridáva k nim FEC paritné polia, prevádza scramblovanie obsahu chráneného algoritmom FEC, pridáva fyzický blok synchronizácie vhodný pre zostupný smer (PSBd) alebo vzostupný smer (PSBu) prenosu a poskytuje časovú synchronizáciu výsledného bitového toku.

Na strane prijímača pod-vrstva prevádza fyzickú synchronizáciu a vymedzenie prichádzajúceho bitového toku, odstraňuje scramblovanie obsahu PHY rámca alebo zhluku, prevádza FEC a extrahuje paritné polia FEC, doručuje výsledné XGTC rámce alebo zhluky pod-vrstve rámcovania. [7]

1.2 Štruktúra XG-PON rámca pre zostupný smer

Táto sekcia vychádza z dokumentácie prenosovej vrstvy XG-PON [7]. XG-PON rámec v zostupnom smere začína s hlavičkou PHY rámca (PSBd), za ktorou nasleduje payload PHY rámca. Špecifikácia pod-vrstvy PHY je definovaná v sekcií 1.1.3. Payload PHY rámca sa po dekódovaní poprednej korekcie a scramblovania stáva XGTC rámcom, ktorý pozostáva z XGTC hlavičky a XGTC payloadu. Celková dĺžka XGTC rámca je 135432 bajtov.



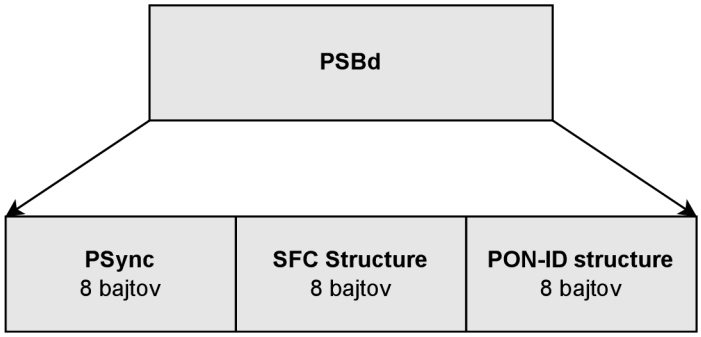
Obr. 1.2: Rámec PHY zostupného smeru.

1.2.1 PSBd

PSBd má veľkosť 24 bajtov a skladá sa z troch 8 bajtových hodnôt. Prvá hodnota je Psync, ktorá slúži ako magické číslo, ktoré určuje začiatok XG-PON rámca a má konštantnú hodnotu `0xC5E5 1840 FD59 BB49` hexadecimálne. Druhá hodnota je štruktúra super rámcového počítadla ktorá má veľkosť 8 bajtov kde posledných 13 bitov je paritné pole ochranného kódu HEC pre overenie správnosti dát. Tretia hodnota je štruktúra PON-ID o veľkosti 8 bajtov kde posledných 13 bitov je paritné pole HEC.

Štruktúra Superframe počítadla

Hodnota počítadla sa v každom PHY rámci zostupného smeru inkrementuje o jedna s následnosťou na predošlí PHY rámec. Pokiaľ počítadlo dosiahne maximálnu hodnotu (samé jedničky), nasledujúcemu PHY rámca sa počítadlo znovu nastaví na počiatočnú hodnotu, ktorá je 0.



Obr. 1.3: Hlavička PHY rámca v zostupnom smere.

Štruktúra PON-ID

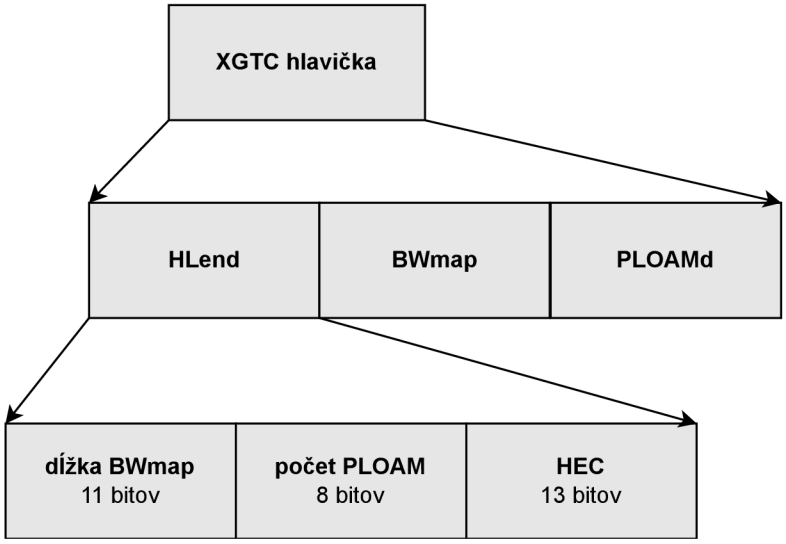
PON-ID je nastavené podľa uváženia OLT a jeho východzia hodnota sú samé nuly. Prvých 8 bitov štruktúry definuje typ PON-ID, kde prvý bit indikuje prislúchajúci obsah pola TOL pre štartovací výkon OLT alebo predlžovača dosahu, označený ako RE flag. Nasledujúce 3 bity typu PON-ID identifikujú nominálne optické parametre vysielača pre ODN triedu, reprezentácia kódu sa nachádza v tabuľke 1.1. Posledné štyri bity typu PON-ID sú rezervované. Za typom PON-ID nasleduje samotného PON-ID o veľkosti 32 bitov s východzou hodnotou samé nuly. Za PON-ID nasleduje pole TOL o veľkosti 11 bitov, ktoré určuje aktuálny štartovací výkon vysielača, kde jeho hodnota v dátovom type integer je relatívna ku 1 μW so šumom o veľkosti 0,1 dB. Nulová hodnota reprezentuje -30 dBm. Posledných 13 bitov štruktúry je paritné pole ochranného kódu HEC.

Tab. 1.1: Kódová reprezentácia triedy ODN.

Hodnota kódu	**ODN rozpočtová trieda**
000	N1
001	N2a
010	N2b
011	E1
100	E2a
101	E2b
110	Rezervované
111	Rezervované

1.2.2 XGTC hlavička

XGTC hlavička pozostáva z troch polí a to pole HLend, BWmap a PLOAMd. Hlavička má premenlivú dĺžku pričom pole HLend má konštantnú dĺžku 4 bajty, pole BWmap sa skladá z N 8 bajtových alokačných štruktúr a pole PLOAMd pozostáva z M 48 bajtových PLOAM správ, ktorých môže byť 0 až niekoľko.



Obr. 1.4: XGTC hlavička zostupného smeru.

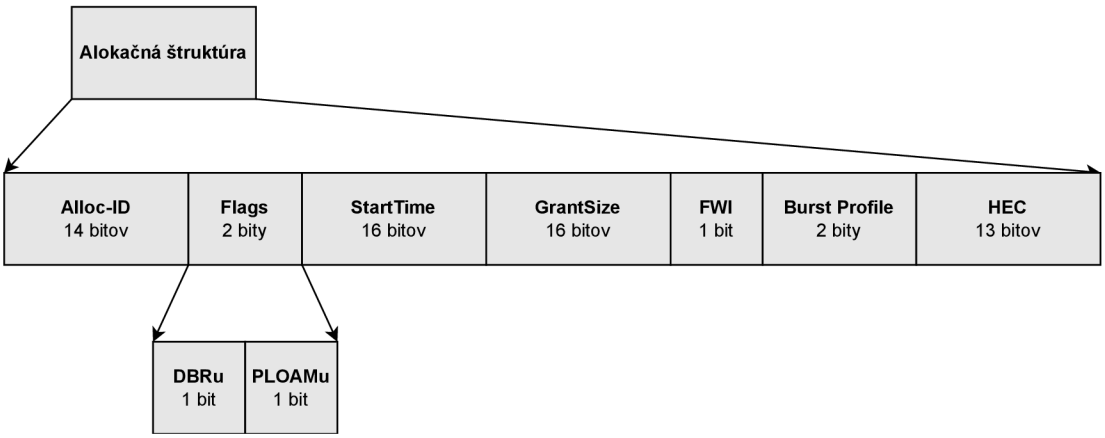
Štruktúra HLend

HLend pole obsahuje dĺžku premenlivých polí v rámci XGTC hlavičky. Pozostáva z troch polí:

- dĺžky BWmap s veľkosťou 11 bitov, ktoré obsahuje bez znamienkové číslo N,
- počtu PLOAM s veľkosťou 8 bitov, ktoré obsahuje bez znamienkové číslo M,
- HEC s veľkosťou 13 bitov, ktoré slúži pre detekciu a opravu chýb v rámci HLend štruktúry.

BWmap partícia

BWmap[8] je séria 8 bajtových alokačných štruktúr (Allocation structure). Nemá konštantnú dĺžku a jej aktuálna veľkosť je $N \times 8$ bajtov. Každá alokačná štruktúra špecifikuje alokáciu šírky pásma konkrétneho Alloc-ID. Sekvencia jednej alebo viacerých alokačných štruktúr, ktoré prislúchajú viacerým Alloc-ID, priradených jednému ONU, sú určené pre súvislí vzostupný prenos a sériu alokačných zhlukov. Alokačná štruktúra sa skladá z nasledujúcich polí.



Obr. 1.5: Alokačná štruktúra.

Pole Alloc-ID obsahuje číslo o veľkosti 14 bitov, ktoré indikuje príjemcu alokácie šírky pásma, konkrétny *Transmission container* (T-CONT) alebo pre vzostupný smer *ONU management and control interface (OMCI) channel* (OMCC) v rámci ONU.

Pole Flags o veľkosti 2 bity. Pokiaľ je nastavený prvý bit znamená to, že ONU by malo poslať DBRu správu pre dané Alloc-ID. Ak bit nie je nastavený, DBRu správa sa neprenáša. Pokiaľ je nastavený druhý bit v prvej alokačnej štruktúre sérií alokačných zhlukov, dĺžka XGTC hlavičky zhluku vo vzostupnom smere by mala byť 52 bajtov a ONU by malo vyslať PLOAM správu ako časť XGTC hlavičky zhluku. Pokiaľ nie je nastavený bit nachádzajúci sa v prvej alokačnej štruktúre zhluku vo vzostupnom smere, dĺžka XGTC hlavičky zhluku by mala byť 4 bajty a PLOAM správa by nemala byť prenášaná. V ďalších nasledujúcich alokačných štruktúrach by mal byť druhý bit pola Flags nastavený na nulu.

Pole StartTime obsahuje 16 bitové číslo, ktoré označuje polohu prvého bajtu XGTC zhluku vzostupného smeru v rámci PHY rámca vzostupného smeru. Štartovací čas je meraný od začiatku PHY rámca vzostupného smeru a má nepresnosť štyri bajty. Hodnota pola 0 odpovedá prvému slovu a hodnota 9719 odpovedá poslednému slovu PHY rámca vzostupného smeru. Iba prvá alokačná štruktúra nesie špecifickú hodnotu pola StartTime, ostatné štruktúry nesú hodnotu pola `0xFFFF`.

Pole GrandSize obsahuje 16 bitovú číselnú hodnotu, ktorá indikuje kombinovanú dĺžku XGTC payloadu s DBRu hlavičkami prenášanú v rámci danej alokácie. GrandSize má rozsah veľkosti 1 slovo (4 bajty). Hodnota pola je rovná nule pre granty, ktoré slúžia iba k prenosu PLOAM správy, vrátane sériového čísla grantov a rozsahu grantov použitých v procese aktivácie ONU. Najmenšia možná nenulová hodnota je 1, ktorá odpovedá jednému slovu alokácie pre prenos správy iba s DBRu hlavičkou. Minimálna alokácia pre samotný XGTC Payload je 4 slová, čo odpovedá hodnote 4

pola GrandSize.

Ďalej nasleduje bit Forced wake-up indication (FWI), ktorý slúži pre označenie ONU, ktoré podporuje protokolovo založený manažment výkonu. OLT nastavuje FWI bit pre zrýchlené prebudenie ONU, ktorá bola v nízko-výkonovom režime. FWI bit sa nastaví v prvej alokačnej štruktúre každej série alokácií pre danú ONU, keď je to vyžadované stavovým strojom riadenia výkonu OLT.

BurstProfile je pole o veľkosti dvoch bitov, ktoré obsahuje index profilu zhluku používaný v Adaptačnej pod-vrstve PHY aby ONU vytvorila formu zhluku. Index odkazuje na skupinu vytvorených profilov pri broadcast alebo unicast komunikácií medzi OLT a ONU pomocou kanálu PLOAM správ. Index je definovaný v Profil správe PLOAM.

Na koniec obsahuje pole HEC, o veľkosti 13 bitov, pre detekciu a opravenie chýb v rámci alokačnej štruktúry. Využitie pola je špecifikované v sekcií 1.7.

PLOAMd partícia

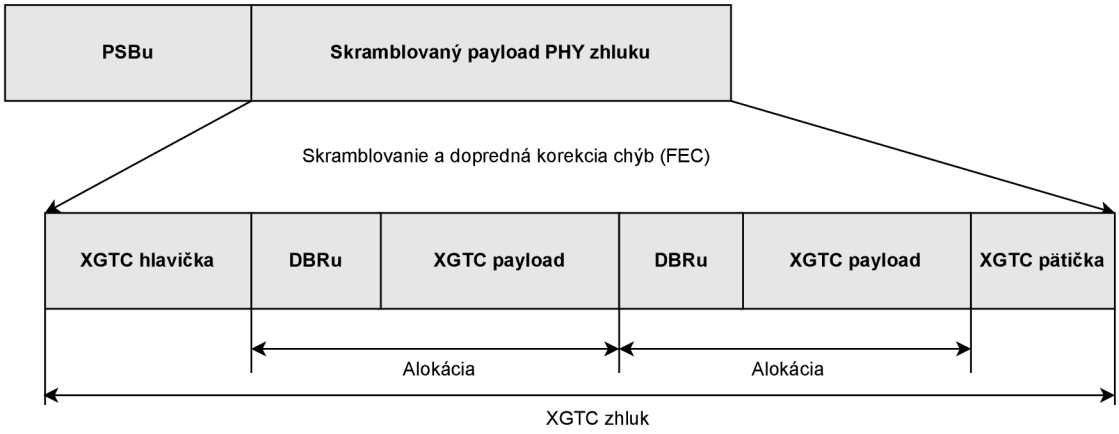
Partícia môže obsahovať nula, jedna alebo viacero PLOAM správ. Dĺžka každej PLOAM správy je 48 bajtov. Počet PLOAM správ v partícií je daný polom počet PLOAM v štruktúre HLend. Celkovú dĺžku PLOAMd partície je možné vypočítať pomocou vzťahu $M * 48$ bajtov. Štruktúra PLOAM správy je definovaná v sekcií 1.5

1.2.3 XGTC payload

Za XGTC hlavičkou nasleduje jej payload, ktorý tvorí zvyšok rámca zostupného smeru. XGTC payload pozostáva z XGEM rámcov, ktoré majú svoju hlavičku a payload. XGEM rámec má rovnakú štruktúru aj pre vzostupný smer. Štruktúra XGEM rámca je definovaná v sekcií 1.4.

1.3 Štruktúra XG-PON rámca pre vzostupný smer

Táto sekcia vychádza z dokumentácie prenosovej vrstvy XG-PON [7]. XG-PON rámec pre vzostupný smer pozostáva z PHY zhluku, ktorý obsahuje hlavičku PSBu, za ktorou nasleduje PHY payload. Po dekódovaní PHY payloadu sa payload stáva XGTC zhlukom. XGTX zhluk pozostáva z XGTC hlavičky, jedného alebo viacero alokačných intervalov šírky pásma a XGTC pätičky. Alokačný interval obsahuje XGTC payload a môže obsahovať alokačnú hlavičku DBRu.



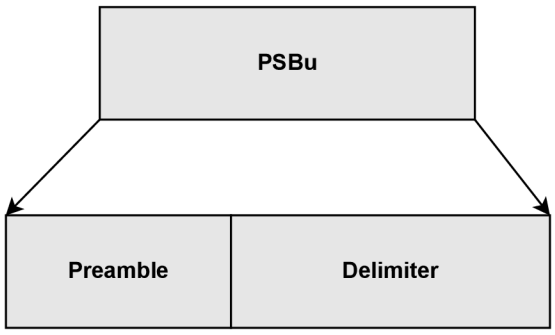
Obr. 1.6: Rámec PHY vzostupného smeru.

1.3.1 PSBu

Hlavička PSBu sa skladá z dvoch polí a to preamble a delimiter. Preamble má veľkosť 4 bajty a delimiter môže mať veľkosť 4 bajty alebo 8 bajtov. Preamble môže nadobúdať dve hodnoty, ktoré ovplyvňujú následné hodnoty delimiteru. Základná hodnota preamble je 0xAAAA AAAA, ktorá poskytuje maximálnu prenosovú hustotu a priemerný výkon. Všetky odporúčané hodnoty na nachádzajú v tabuľke 1.2.

Tab. 1.2: Hodnoty hlavičky PSBu.

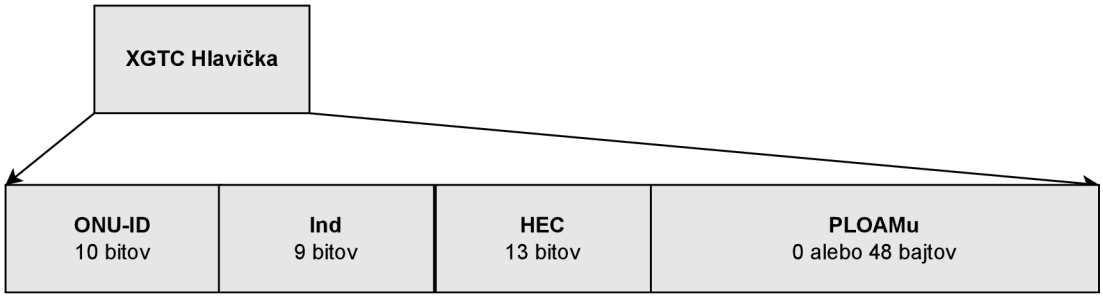
Preamble	**4 bajty Delimiter**	**8 bajtov Delimiter**
0xBB52 1E26	0xA376 70C9 0x4BDE 1B90 (FEC zapnuté) 0xA376 70C9 (FEC vypnuté)	0xB9D4 3E68 462B C197 0xB9D4 3E68 462B C197 (FEC zapnuté) 0xB752 1F06 48AD E879 (FEC vypnuté)
0xAAAA AAAA	0xAD4C C30F 0xA566 79E0 (FEC zapnuté) 0xAD4C C30F (FEC vypnuté)	0xB3BD D310 B2C5 0FA1 0xB3BD D310 B2C5 0FA1 (FEC zapnuté) 0xCE99 CE5E 5028 B41F (FEC vypnuté)



Obr. 1.7: Hlavička PHY rámca vo vzostupnom smere.

1.3.2 XGTC hlavička

Za hlavičkou PSBu nasleduje hlavička XGTC. XGTC hlavička pozostáva z troch pevných polí o veľkosti 4 bajty a voliteľného pola PLOAMu, ktoré obsahuje PLOAM správu o veľkosti 48 bajtov.



Obr. 1.8: XGTC hlavička vzostupného smeru.

Pole ONU-ID

Prvé pole je ONU-ID o veľkosti 10 bitov. Toto poje obsahuje unikátne ONU-ID pridelené ONU, ktoré poslalo vzostupnú správu. ONU-ID je pridelené behom aktivačného procesu. Pokiaľ ONU, ktoré ešte nemá pridelené ONU-ID odpovedá na broadcast žiadosť pridelenia za účelom oznámenia svojej prítomnosti v PON, mala by použiť hodnotu 0x03FF (1023).

Indikačné pole

Ďalej nasleduje Indikačné pole o veľkosti 9 bitov, ktoré je používané pre rýchle nevyžiadané signalizovanie stavu ONU. Význam bitov je následovný:

- Bit 8 (MSB) – stav PLOAM fronty. Pokiaľ je tento bit nastavený na hodnotu 1, znamená to že ONU fronta s PLOAM správami nie je prázdna a po aktuálne zaslanom zhluku je príslušná PLOAM správa poslaná v ďalšom zhluku. Ak bit nie je nastavený znamená to že je fronta prázdna.
- Bit 7 až 1 – Rezervované bity.
- Bit 0 (LSB) – Posledný signál (Dying Gasp). Pokiaľ je tento bit nastavený znamená to že ONU detekovalo podmienky, ktoré jej môžu zabrániť odpovedať na vzostupnú alokáciu šírky pásma. Táto indikácia môže pomôcť OLT rozlíšiť problémy umiestnenia vlákna od priestorových problémov. Zaslanie indikácie neznamená nevyhnutne záväzok alebo zámer pre vypnutie ONU vysielača. Ak sťažené podmienky už nepretrvávajú, ONU odvolá indikáciu a pokračuje v pravidelnej prevádzke. OLT by nemalo interpretovať Posledný signál ako podloženie odstúpenia od alokácie šírky pásma pridelenej ONU.

Pole HEC

Posledné pevné pole je pole HEC o veľkosti 13 bitov, ktoré slúži pre detekciu a opravenie chýb v rámci XGTC hlavičky, vzniknutých pri prenose médiom. Využitie pola je špecifikované v sekcií 1.7.

PLOAMu

Pole PLOAMu obsahuje presne jednu správu PLOAM, pokiaľ je pole prítomné. Prítomnosť PLOAM správy je riadená zariadením OLT pomocou príznakového bitu v prvej alokačnej štruktúre, prislúchajúcej zhluku alokačnej série v zostupnom smere. Formát PLOAM správ pre vzostupný smer je definovaný v sekcií 1.5

1.3.3 Hlavička alokácie DBRu

Hlavička alokácie nemusí byť prítomná v rámci danej alokácie. Prítomnosť hlavičky je riadená zariadením OLT pomocou DBRu príznakového bitu v zodpovedajúcej alokačnej štruktúre v rámci BWmap partície v zostupnom smere. DBRu štruktúra má veľkosť 4 bajty a nesie správu o stave vyrovnávacej pamäti, ktorá je priradená k špecifickému Alloc-ID.

Pole BufOcc

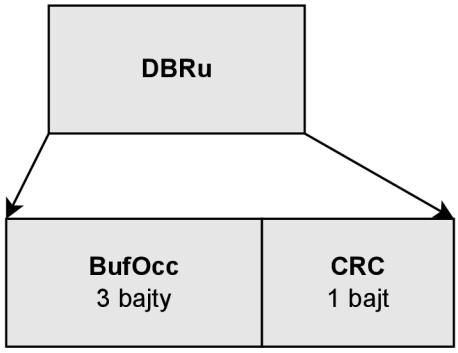
Pole má veľkosť 3 bajty a nesie v sebe obsadenosť vyrovnávacej pamäti, čiže celkové množstvo prenesených SDU vyjadrené v jednotke 4 bajtových slov v rámci danej alokácie. Pokiaľ individuálne SDU má dĺžku L bajtov, jej príspevok W do obsadenosti vyrovnávacej pamäti je vypočítaný následovne:

$$W = \begin{cases} \left\lceil \frac{L}{4} \right\rceil, & \text{if } L > 8 \\ 2, & \text{if } 0 < L \leq 8 \end{cases} \tag{1.1}$$

Hodnota pola udáva najlepší možný odhad, ktorý odpovedá časovému okamihu vyslaniu rámca, čiže začiatku alokačného intervalu vo vzostupnom smere. Vrátane ohlásenej hodnoty by mala byť akákoľvek premávka, ktorá mohla byť naplánovaná pre vzostupný prenos v rámci alokačného intervalu. Pole BufOcc zahrňuje 2 špeciálne hodnoty a to 0x000000, ktorá označuje prázdnu vyrovnávaciu pamäť a 0xFFFFFF, ktorá označuje neplatné meranie.

Pole CRC

Pole CRC má veľkosť 1 bajt a chráni DBRu štruktúru pomocou ochranného kódu Cyclic redundancy check CRC-8.[9] Použitý polynóm je $g(x) = x^8 + x^2 + x + 1$.



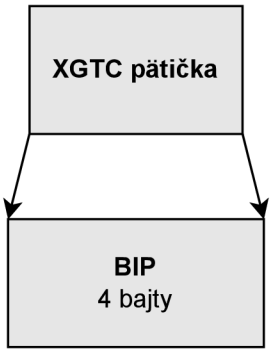
Obr. 1.9: Alokačná hlavička.

1.3.4 XGTC payload

Veľkosť každej XGTC payload sekcie v zhluku sa rovná veľkosti alokácie, od ktorej je potrebné odčítať veľkosť alokačnej hlavičky v prípade jej prítomnosti. V rámci jedného XGTC zhluku sa môže nachádzať jeden alebo viac XGTC payload sekcií, v prípade že jednému ONU (ONU-ID) bolo pridelených viacero alokačných intervalov, ktoré sú asociované pomocou špecifického Alloc-ID. XGTC payload pozostáva z XGEM rámcov, ktoré majú rovnakú štruktúru ako pri zostupnom smere. Štruktúra XGEM rámca je definovaná v sekcií 1.4.

1.3.5 XGTC pätička

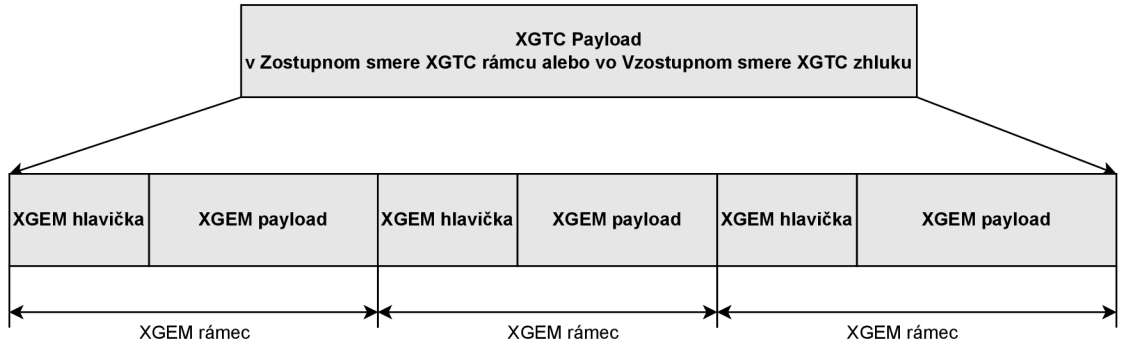
Pätička XGTC zhluku obsahuje párnu bitovo prekladanú paritu, BIP pole, o veľkosti 4 bajty. BIP je vypočítané cez celý XGTC zhluk. OLT prijímač overí BIP pre *Bit Error Rate* (BER) na optickej linke pre vzostupný smer. Odhad bitovej chybovosti pomocou BIP je možné použiť len v prípade vypnutého FEC v PHY pod-vrstve. V prípade zapnutého FEC je potrebné chybovosť vypočítať z výsledkov korekcie FEC.



Obr. 1.10: Pätička XGTC rámca.

1.4 XGEM rámcovanie

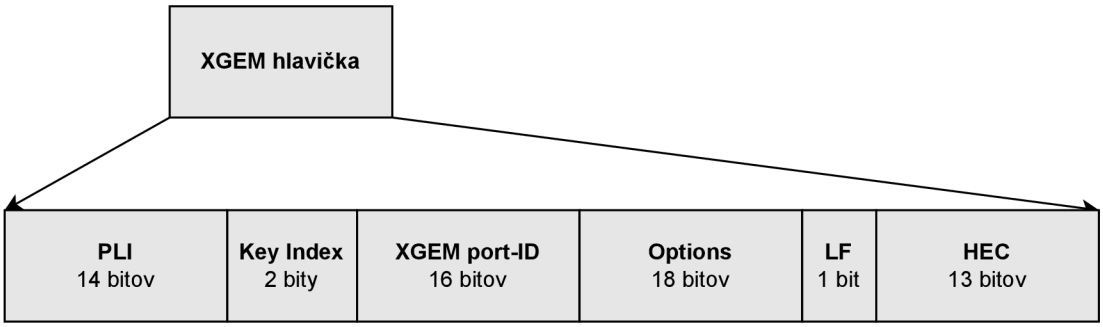
V XG-PON systéme sú SDUs, ktoré obsahujú používateľské dátové rámce a vysoko úrovňové PON riadiace rámce *Management and Control Interface* (OMCI), prenášané v XGTC payload sekciách XGTC rámca v zostupnom smere a v XGTC zhluku vo vzostupnom smere za použitia XGEM. XGEM podporuje fragmentáciu, enkapsuláciu a vymedzenie SDU, ktoré sa využívajú v oboch smeroch komunikácie. XGTC payload môže obsahovať niekoľko XGEM rámcov, ktoré obsahujú XGEM hlavičku a XGEM payload.



Obr. 1.11: XGEM rámec.

1.4.1 XGEM hlavička

Veľkosť XGEM hlavičky je 8 bajtov a pozostáva z polí v nasledujúcom poradí.



Obr. 1.12: Hlavička XGEM rámca.

Indikácia dĺžky payloadu (PLI)

PLI pole má veľkosť 14 bitov a obsahuje dĺžku payloadu L v bajtoch. Dĺžka L je dĺžka SDU alebo SDU fragmentu v poli XGEM payload, ktorý nasleduje za XGEM

hlavičkou. 14 bitové pole umožňuje maximálnu číselnú hodnotu 16383 a preto je možné zakódovať dĺžku rozšíreného Ethernet rámca (do 2000 bajtov) a taktiež jumbo Ethernet rámca (do 9000 bajtov). Hodnota pola PLI je presná na jeden bajt a nie je nevyhnutne rovná dĺžke XGEM payloadu, pretože je potrebné aby bol zarovnaný na 4 bajtové slová.

Index kľúču (Key Index)

Index kľúču je indikátor šifrovacieho kľúču použitého pre šifrovanie XGEM payloadu o veľkosti dvoch bitov. V závislosti na XGEM Port-ID, index kľúču odkazuje na unicast alebo broadcast typ kľúču. Až dva kľúče môžu byť aktívne naraz pre oba typy kľúčov. Hodnota indexu kľúču 01 odkazuje na prvý kľúč a hodnota 10 odkazuje na druhý kľúč. Hodnota 00 indikuje payload bez šifrovania a hodnota 11 je rezervovaná pre budúce použitie. Pokiaľ XGEM rámec obsahuje tieto dve hodnoty tak je vyradený. [10]

XGEM Port-ID

Pole má veľkosť 16 bitov a obsahuje identifikátor XGEM Portu, do ktorého daný rámec patrí.

Pole Options

Použitie pola options zostáva na ďalšie štúdium. Pole je nastavené na hodnotu nula vysielačom a je ignorované na strane prijímača.

Posledný Fragment (LF)

Pole LF je príznakovým bitom. Pokiaľ enkapsulovaný fragment do XGEM rámca je posledným fragmentom SDU alebo je enkapsulovaná kompletná SDU tak LF bit je nastavený na hodnotu 1, inak je LF bit nastavený na hodnotu 0.

HEC

Pole HEC má veľkosť 13 bitov a slúži k detekcií a oprave chýb v rámci XGEM hlavičky. Výpočet opravného kódu HEC je špecifikovaný v sekcií 1.7.

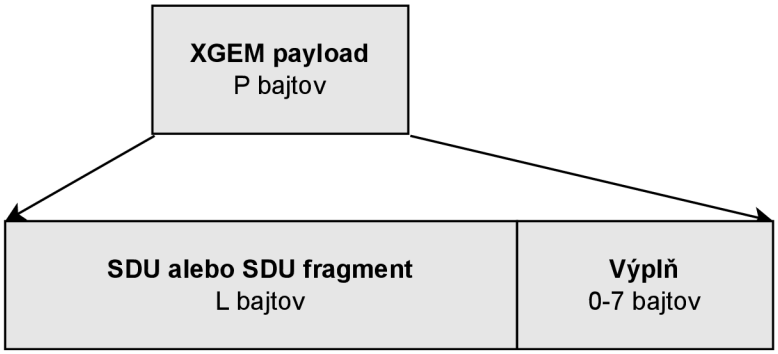
1.4.2 XGEM payload

XGEM payload má premenlivú dĺžku, ktorú je možné zistiť pomocou pola PLI v XGEM hlavičke. Pre XGEM rámec s užitočnými dátami, dĺžka XGEM payloadu

P súvisí s hodnotou L, ktorá je prenášaná v PLI poli. Podľa nasledujúceho vzťahu je možné vypočítať aktuálnu dĺžku payloadu v bajtoch:

$$P = \begin{cases} 4 * \left\lceil \frac{L}{4} \right\rceil, & \text{if } L \geq 8 \\ 8, & \text{if } 0 < L < 8 \\ 0, & \text{if } L = 0 \end{cases} \tag{1.2}$$

Payload môže obsahovať jeden až sedem bajtov výplne na pozícií najmenej významných bajtov. Vysielač použije pre výplň bajtov hodnotu 0x55. Po prijatí je výplň zahodená.



Obr. 1.13: Payload XGEM rámca.

XGEM payload obsahuje SDU alebo SDU fragment. Fragmentácia SDU môže nastať v oboch smeroch komunikácie a podlieha nasledujúcim pravidlám. Pre zostupný smer platí, že pokiaľ je dostupná dĺžka XGTC payload v aktuálnom XGTC rámci aspoň 16 bajtov a dĺžka SDU dostupná pre prenos, ktorá zahrňuje 8 bajtovú XGEM hlavičku, presiahne dostupný payload, SDU by mala byť rozdelená do dvoch fragmentov a to tak, že prvý fragment kompletne zaplní dostupný payload aktuálneho XGTC rámca, zatiaľ čo druhý fragment je prenesený v XGTC payloade ďalšieho rámca. Ak dĺžka druhého SDU fragmentu je menej ako 8 bajtov, potom by mala byť vyplnená na 8 bajtov pre splnenie minimálnej veľkosti XGEM rámca 16 bajtov. Akonáhle začala SDU fragmentácia, druhý fragment SDU musí byť odoslaný pred akoukoľvek inou SDU.

Pre vzostupný smer platí, že pokiaľ dostupný XGTC payload v aktuálnej alokácií je aspoň 16 bajtov a dĺžka SDU alebo SDU fragmentu naplánovaná k prenosu, ktorá zahrňuje 8 bajtovú XGEM hlavičku prekročí dostupný payload, SDU by mala byť rozdelená do dvoch fragmentov a to tak, že prvý SDU fragment plne obsadí dostupný XGTC payload v aktuálnej alokácií, zatiaľ čo zvyšok SDU je prenesený v XGTC payloade nasledujúcej alokácií, spojenej s rovnakým Alloc-ID, ktorá podlieha rovnakým pravidlám fragmentácie. Akonáhle začala SDU fragmentácia, všetky

fragmenty SDU musia byť odoslané pred akoukoľvek inou SDU spojenou s rovnakým Alloc-ID.

Pokiaľ veľkosť dostupného XGTC payloadu je menej ako 16 bajtov, mal by byť vyplnený Nečinným XGEM rámcom.

1.4.3 Idle XGEM rámec

Pokiaľ vysielač nemá žiadne SDU alebo SDU fragmenty k vyslaniu alebo ich dĺžka prekročí dostupnú XGTC payload sekciu, ale fragmentovať ich by porušilo pravidlá, vysielač musí vygenerovať Nečinné (Idle) XGEM rámce pre vyplnenie dostupnej XGTC payload sekcie.

Nečinný XGEM rámec je každý XGEM rámec s hodnotou XGEM port-ID rovnej 0xFFFF. Pole PLI obsahuje aktuálnu veľkosť payload rámca, ktorá môže byť rovná akémukoľvek násobku 4, vrátane 0 do maximálnej podporovanej SDU dĺžky. Nečinné rámce sú odoslané nešifrovane, kde pole Key Index indikuje vypnuté šifrovanie a hodnota pola LF je 1. Prijímač ignoruje rámce s týmito hodnotami.

Obsah XGEM payloadu Nečinného XGEM rámca je zostavený vysielačom podľa vlastného uváženia s potrebným ohľadom na kontrolu vzoru linky a prevenciu degradácie nosnej vlny. Pokiaľ dostupné miesto na konci XGTC payload sekcie je menej ako veľkosť XGEM hlavičky (miesto je rovné štyri bajty), vysielač musí vygenerovať krátky Nečinný XGEM rámec, ktorý je definovaný ako štyri bajty s hodnotami nula.

1.5 PLOAM správy

PLOAM kanál odosielania správ v XG-PON systéme je prevádzkové a riadiace zariadenie medzi OLT a ONUs, ktoré je založené na fixnej sade prenášaných správ v rámci vyhradeného pola v hlavičke XGTC rámca a hlavičke XGTC zhluku. OLT a ONU PLOAM spracovanie sa javí ako klient príslušnej pod-vrstvy rámcovania XGTC. Kanál PLOAM poskytuje viac flexibilnú funkcionalitu ako vstavané riadiace kanály a je vo všeobecnosti rýchlejší ako OMCI kanál.

PLOAM kanál podporuje riadiace funkcie XGTC vrstvy. Je založený na výmene 48 bajtových správ, ktoré sú prenášané v PLOAM partícií v oboch smeroch komunikácie. PLOAM kánál podporuje nasledujúcu funkcionalitu:

- komunikáciu profilu zhluku,
- aktiváciu ONU,
- registráciu ONU,
- výmenu a aktualizáciu šifrovacieho kľúču,
- signalizáciu spínania ochrany,
- správu napájania.

1.5.1 Formát PLOAM správ

Všeobecná štruktúra správy PLOAM je zobrazená v tabuľke 1.3. Typy správy sú definovaný v nasledujúcej podsekcií.

Tab. 1.3: Všeobecná štruktúra PLOAM správy.

Oktet	Pole	Popis
1-2	ONU-ID	Veľkosť 10 bitov, zarovnaných na koniec 2 bajtového pola (LSB). Šesť najviac významových bitov je rezervovaných a mali by byť nastavené na hodnotu 0.
3	ID typu správy	Tento bajt označuje typ správy. Číselný kód pre každú správu je definovaný v ďalšej podsekcií.
4	SeqNo	Sekvenčné číslo.
5-40	Obsah správy	Obsah správy je špecifický pre každý typ správy.
41-48	MIC	Pole pre kontrolu integrity správy.

ONU-ID

Pole ONU-ID obsahuje 10 bitový ONU identifikátor, ktorý je rozšírený o 6 bitov s hodnotami nula. Toto ID špecifikuje príjemcu PLOAM správy v zostupnom smere alebo odosielateľa správy vo vzostupnom smere. Počas aktivácií ONU je mu priradené ONU-ID v rozsahu od 0 do 1022. Hodnota ONU-ID 1023 (0x03FF) je využívaná ako identifikátor broadcast správy v zostupnom smere alebo označuje ONU, ktoré ešte nemá pridelené ONU-ID vo vzostupnom smere.

ID typu správy

ID typu správy je pole o veľkosti 8 bitov, ktoré špecifikuje typ správy a definuje sémantiku obsahu správy. Validné hodnoty tohoto pola sú definované v nasledujúcej podsekcií a všetky ostatné hodnoty sú rezervované a nemali by byť poslané.

SeqNo

Pole SeqNo má veľkosť 8 bitov a obsahuje sekvenčné číslo počítadla, ktoré zvyšuje robustnosť kanálu PLOAM. Pre zostupný smer OLT udržiava oddelené číslovanie pre každé ONU zvlášť a tiež pre broadcast správy. Počítadlo má počiatočnú hodnotu 1. Počítadlo pre broadcast je nastavená pri reštarte OLT a pre každú ONU je nastavené pri pridelení ONU-ID. Pri pretečení maximálnej hodnoty 255 je počítadlo nastavené na počiatočnú hodnotu 1. Pre vzostupný smer je hodnota počítadla rovná počítadlu z PLOAM správy zostupného smeru. Jedna hodnota pola SeqNo sa môže vyskytnúť vo viac ako len jednej správe zostupného smeru. Taktiež môže byť použitá hodnota 0 v prípade odpovedi na pridelenie sériového čísla.

Obsah správy

Oktety 5 až 40 PLOAM správy sú využité ako payload správy. Nevyužité oktety v rámci PLOAM správy sú doplnené vysielačom hodnotou 0x00. Obsah jednotlivých typov správ je definovaný v dokumente [7] v sekcií 11.3.

MIC

MIC (Message integrity check) je pole o veľkosti 8 bajtov, ktoré je použité k overeniu identity odosielateľa a predchádza útoku vytvorenia PLOAM správy. Za použitia obsahu PLOAM správy a PLOAM zdieľaného kľúču, odosielateľ vypočíta MIC a pošle ho v rámci PLOAM správy. Za použitia prijatej PLOAM správy a zdieľaného kľúču príjemca vypočíta jeho verziu MIC a porovná ju s prijatou hodnotou MIC. Pokiaľ sú obe hodnoty rovné, PLOAM správa je validná a pracuje sa s ňou ďalej.

1.5.2 Typy PLOAM správ

Pre zostupný a vzostupný smer komunikácie sú definované protichodné správy. Všetky definované typy PLOAM správ zostupného smeru sa nachádzajú v tabuľke 1.5 a typy PLOAM správ vzostupného smeru v 1.4. Identifikátor typu správy, ktorý sa nenachádza v týchto tabuľkách by sa nemal vyskytnúť v rámci PLOAM správy.

Tab. 1.4: Typy PLOAM správ vzostupného smeru.

ID typu správy	Názov správy	Funkcia
0x01	Serial_Number_ONU	Ohlásenie sériového čísla ONU.
0x02	Registrácia	Ohlásenie registračného ID ONU.
0x05	Key_Report	Poslanie fragmentu nového šifrovacieho kľúču alebo hash existujúceho kľúču.
0x09	Acknowledgement	Indikácia prijatia špecifickej správy zostupného smeru. Tiež používaná pre prázdne, chybové a busy odpovede.
0x10	Sleep_Request	Oznámenie od ONU so zámerom začať alebo ukončiť režim úspornej energie.

Tab. 1.5: Typy PLOAM správ zostupného smeru.

ID Typu správy	**Názov správy**	**Funkcia**
0x01	Profil	Broadcast alebo unicast správa, ktorá poskytuje informácie o hlavičke zhluku vzostupného smeru.
0x03	Assign_ONU-ID	Pripojenie voľnej ONU-ID hodnoty s ONU sériovým číslom
0x04	Ranging_Time	Indikácia vyrovnania oneskorenia round-trip. Pri broadcast správe môže byť použitá pre vyrovnanie oneskorenia všetkých ONUs.
0x05	Deactivate_ONU-ID	Rozkázať špecifickému ONU aby zastavilo vysielanie a resetovalo sa. Môže byť použitá ako broadcast.
0x06	Disable_Serial_Number	Broadcast správe pre vypnutie alebo zapnutie ONU s aktuálnym sériovým číslom.
0x09	Request_Registration	Vyžiadanie registračného ID od ONU.
0x0A	Assign_Alloc-ID	Pridelenie špecifického Alloc-ID pre špecifické ONU.
0x0D	Key_Control	OLT prikáže ONU vygenerovať nový šifrovací kľúč alebo žiada ONU o potvrdenie existujúceho kľúču.
0x12	Sleep_Allow	Zapnutie alebo vypnutie režimu úspornej energie ONU v reálnom čase.

1.6 Aktivácia ONU

Aktivačný proces je sada distribuovaných procedúr, ktoré umožňujú pripojiť alebo obnoviť operácie neaktívnej ONU v PON. Aktivačný proces ONU je ovládaný pomocou OLT, ktoré vydáva funkcie pridelenia grantu šírky pásma a výmenu PLOAM správ vzostupného a zostupného smeru. Udalosti aktivačného procesu v obvyklom poradí sú následovné.

1. ONU vstúpi do aktivačného procesu, zachytením vysielania zostupného smeru, z ktorej si vyberie pole PSync a štruktúru Superframe počítadla.
2. ONU následne poslúcha na Profil PLOAM správu, periodicky vysielanú od zariadenia OLT, aby začalo učenie profilov zhluku, ktoré špecifikujú vysielanie vzostupného smeru.
3. Keď ONU príjme grant sériového čísla so známym profilom, ohlási svoju prítomnosť v PON s PLOAM správou Serial_Number_ONU.
4. OLT príjme sériové číslo novo pripojenej ONU a pridelí mu ONU-ID za pomoci PLOAM správy Assign_ONU-ID.
5. Voliteľne OLT vyšle riadené meranie grantu novo objavenému ONU a začne precízne merať čas odozvy.
6. Následne ONU posiela Registračnú PLOAM správu.
7. OLT vykoná počiatočnú autentizáciu ONU, založenú na registračnom ID, vypočíta individuálne vyrovnanie oneskorenia a prenesie vypočítanú hodnotu pre ONU za použitia PLOAM správy Ranging_Time.
8. ONU upraví začiatok vysielania jeho PHY rámca vzostupného smeru pomocou časovaču a hodnoty vyrovnania oneskorenia.
9. ONU dokončí aktiváciu a začne regulárnu prevádzku.

Pre ONUs v stave regulárnej prevádzky, OLT monitoruje fázu a BER prichádzajúcej komunikácie. Zo zachytených informáciach o fáze komunikácie, OLT môže prepočítať a dynamicky zmeniť vyrovnanie oneskorenia pre akékoľvek ONU.

1.7 Hybrid error correction (HEC)

Hybrid Error Correction (HEC)[11] je použité v XG-PON na niekoľkých miestach. V XGTC hlavičke je použitý pre chránené pole o veľkosti 19 bitov, čo vytvára celkovú veľkosť štruktúry 32 bitov. V BW-map a XGEM poliach je použitý HEC pre chránené pole o veľkosti 51 bitov, čo vytvára veľkosť štruktúry 64 bitov. Pri výpočte HEC je 19 bitové chránené pole rozšírené o 32 bitov núl zo začiatku pola. HEC dokáže opraviť 2 chybové bity a detekovať 3 chybové bity. Pozostáva z dvoch častí.

Prvá časť je BCH(63,12,2) kód. Generačný polynóm pre tento kód je následovný:

$$x^{12} + x^{10} + x^8 + x^5 + x^4 + x^3 + 1. \tag{1.3}$$

Tento kód je použitý na chránené pole (51 bitov), aby 63 bitový výsledok bol deliteľný generačným polynómom. Vlastnosť kódu spočíva v unikátnom 12 bitovom syndróme pre každú jedno-bitovú a dvoj-bitovú chybu. Troj-bitové chyby môžu vytvoriť rovnaký syndróm ako pri dvoj-bitovej chybe alebo neplatný kód, ale neexistuje syndróm troj-bitovej chyby, ktorý by bol rovnaký ako syndróm jedno-bitovej chyby. Táto vlastnosť povoľuje detekovať a vylúčiť troj-bitové chyby za použitia jedného paritného bitu. Existuje 63 unikátnych syndrómov jedno-bitových chýb a 1953 unikátnych syndrómov dvoj-bitových chýb. Celková hodnota 12 bitového priestoru tvorí 4095 možných syndrómov z ktorých je 2079 syndrómov neplatným kódom. Neplatný kód môže vzniknúť pri troch-bitových chybách a viac-bitových chybách.

Druhá časť je jeden paritný bit na konci. Paritný bit je nastavený v prípade že celkový počet jednotiek v chránenom poli a HEC poli je párne číslo. Parita indikuje nepárny počet chýb, ktoré nastali v hlavičke. BCH kód nezahrňuje paritný bit do výpočtu ale paritný bit zahrňuje BCH kód do výpočtu.

Bitová pozícia 63 odkazuje na prvý bit chráneného 51 bitového pola a bitová pozícia 1 odkazuje na predposledný bit pola HEC. Pozícia 0 je rezervovaná pre použitie paritného bitu.

1.8 Sieťový analyzátor pre 10Gb Pasívnu Optickú sieť

Sieťový analyzátor je založený na hardware platforme Xavier POS a obsahuje firmware pre XG-PON[2]. Firmware pozostáva z dvoch hlavných programovateľných komponentov a to FPGA a NVIDIA Jetson Xavier AGX modul. Tieto komponenty umožňujú zachytávať sieťovú komunikáciu a predávať ju ďalšiemu softwaru, ktorý ju dokáže spracovať. [6] Text tejto podkapitoly vychádza z dokumentu [12].

1.8.1 Spracovanie dát prenášaných po Pasívnej Optickej Sieti (PON)

Dáta v zostupnom smere sú scramblované. Pričom jediná nescramblovaná časť dát je PSYNC nachádzajúca sa v hlavičke PSBd (viz. sekcia 1.2.1), ktorá sa vyskytuje na začiatku každého rámca. Analyzátor sa zarovná na začiatok rámca a spracuje ostatok bloku PSBd. Vyčíta SFC a PON-ID, oba údaje overí pomocou opravného kódu HEC. Tieto polia sú scramblované inak ako ostatok dát. Ďalej začne spracovávať rámec PHY, ktorý je potrebné descramblovať. Po descramblovaní dát sú kontrolované a opravované jednotkou FEC, ktorá v prípade zlého signálu výrazne zníži chybovosť na linke. Dáta sú v tomto bode obalené potrebnými hlavičkami a pošlú sa cez PCIe

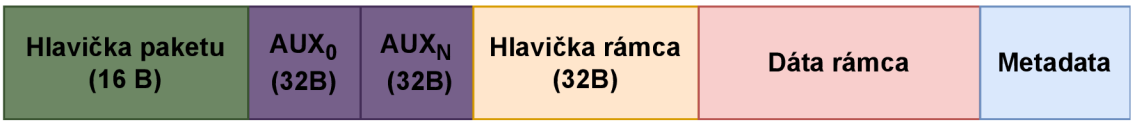
[2] `https://www.dfcdesign.cz/cz/cecilie-xpon-module`

nadriadenému systému. Z dát je extrahované PLOAMd pole a pošle sa procesoru pre načasovanie prijatia časovo multiplexovaných dát vzostupného smeru.

Dáta vo vzostupnom smere sú taktiež najprv zarovnané podľa delimiteru v hlavičke PSBu. Narozdiel od zostupného smeru, ktorý si vystačí bez nadbytočných informácií, sú potrebné údaje získané pri spracovaní zostupného smeru. Je potrebné aby analyzátor poznal hodnotu delimiteru, ktorá môže byť pre každý balík dát iná a pozná ju na základe zoznamu profilov, ktorý udržuje riadiaci procesor podľa PLOAMd správ zachytených v zostupnom smere. Pre descramblovanie je potrebná informácia o stave SFC, v ktorej bol prenos balíku naplánovaný. Pre FEC je potrebná informácia o povolení FEC, teda prítomnosti paritných polí. Dáta sú v tomto bode obalené hlavičkami a pošlú sa cez PCIe nadriadenému systému. Z dát je extrahované PLOAMu pole a pošle sa procesoru pre odvodenie vzťahu zostupného a vzostupného smeru.

1.8.2 Protokol zapuzdrenia paketov XG-PON

Dáta sú prenášané v paketoch. Každý paket analyzátora pozostáva z hlavičky paketu, ktorá má konštantnú dĺžku. Za hlavičkou paketu môžu nasledovať voliteľné pomocné správy (AUX). Pokiaľ paket obsahuje XG-PON dáta, predchádza ho hlavička rámca, ktorá má konštantnú dĺžku. Pri absencií AUX správ bude za hlavičkou paketu nasledovať hlavička rámca a rámec s dátami. Dĺžka dát nie je predom známa ale je možné ju odvodiť. Na koniec paketu sú pridané metadáta, ktoré obsahujú svoju dĺžku. Dátový rámec obsahuje samotný XG-PON rámec pre zostupný alebo vzostupný smer. Dáta pre oba smery začínajú za hlavičkou PSB.



Obr. 1.14: Paket sieťového analyzátora.

Hlavička paketu

Hlavička sa skladá z verzii protokolu, magického čísla, sekvenčného čísla, dĺžky paketu (voliteľné) a rezervovaného priestoru. Hlavička má konštantnú veľkosť 16 bajtov.

	0	1	2	3	4	5	6	7
0	Verzia protokolu (1.0)		Magické číslo		Sekvenčné číslo			
	0x10	0x00	0xC0	0xDF	0xXX	0xXX	0xXX	0xXX
1	Dĺžka paketu		Rezervované					
	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Obr. 1.15: Hlavička paketu.

Pomocné správy (Auxiliary messages)

Pomocné správy sú súčasťou dátového toku XG-PON siete. Správy môžu obsahovať napríklad nečakané udalosti, stavy, hranice rámcov a ďalšie. Základná veľkosť je 32 bajtov ale môže byť aj dlhšia. Pomocná správa sa líši od hlavičky rámca v siedmom bite. Pokiaľ je siedmy bit 1 je to pomocná správa a ak 0 je to hlavička rámca. Pomocné správy obsahujú kontrolný blok o veľkosti 8 bitov kde prvé tri bity sú rezervovaný priestor, ďalšie štyri bity sú typ správy a posledný bit rozlišuje pomocnú správu a hlavičku rámca.

	0	1	2	3	4	5	6	7
0	Časové razítko				Rezervované			Kontrolné
	0xXX	0xXX	0xXX	0xXX	0x00	0x00	0x00	0x80
1 - 3	Rezervované alebo podla typu správy							
	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Obr. 1.16: Pomocná správa.

Hlavička rámca

Hlavička rámca sa líši od pomocnej správy v poslednom bite siedmeho bajtu. Pokiaľ je posledný bit 0 je to hlavička rámca a ak 1 je to pomocná správa. Hlavička má konštantnú dĺžku 32 bajtov. Na začiatku má hlavička časové razítko o veľkosti 63 bitov. Ďalší bit je C flag, ktorý určuje či sa jedná o hlavičku rámca alebo pomocnú správu. Ďalej nasleduje S flag, ktorý má veľkosť 8 bitov a prvý bit indikuje prvý paket pri potrebe rozdeliť paket do fragmentov, ostatné bity sú rezervované. Ďalších 56 bitov je rezervovaných. Za nimi nasleduje lineárny čas, ktorý má veľkosť 48 bitov a využíva sa najmä pre plánovanie RX resetov. Zostávajúce bity sú rezervované.

Metadáta

Keďže dĺžka celého paketu nie je známa, je potrebné dekódovať metadáta od konca, kde je uložená dĺžka metadát. Je to dĺžka celého bloku, je udávaná v 32 bitových slovách a jej minimálna hodnota je 1, pretože sa započítava aj samotný ukazatel

	0	1	2	3	4	5	6	7
0	Časové razítko 63 bitov							
	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xX0
1	S	Rezervované						
	0xX0	0x00	0x00	0x00	0x00	0x00	0x00	0x00
2	Lineárny čas						Rezervované	
	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0x00	0x00
3	Rezervované							
	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Obr. 1.17: Hlavička rámca.

dĺžky. Blok metadát môže obsahovať ľubovolné množstvo podblokov metadát. Každý blok obsahuje Metadata ID, ktoré identifikuje jeho obsah.

N0 slov	N1 slov	Nn slov	1 slovo
Podblok 0	Podblok 1	Podblok N	Pätička bloku metadát

Obr. 1.18: Štruktúra metadát.

Pätička bloku metadát

Pätička musí byť dekódovaná ako prvá. Obsahuje počet slov metadát, ktoré sa nachádzajú pred pätičkou, vďaka čomu je možné odlíšiť metadáta od predchádzajúcich dát. Pätička sa skladá z magického čísla, hodnoty platnosti dát a počet slov metadát.

Bajt 3	Bajt 2	Bajt 1	Bajt 0
Magické číslo 0xEA	Flagy	Počet slov podbloku	

Obr. 1.19: Pätička bloku metadát.

Podblok metadát

Podblok metadát predstavuje konkrétny typ informácií. Jeho štruktúra a typ informácie je daný hodnotou Metadata ID, ktorá sa nachádza v hlavičke podbloku. Pätička podbloku nesie informáciu o dĺžke celého podbloku a teda musí byť dekódovaná ako prvá.

Hlavička podbloku obsahuje magické číslo a Metadata ID. Pätička podbloku obsahuje magické číslo, príznakový bajt a počet slov podbloku.

1 slovo	N slov	1 slovo
Hlavička podbloku	Metadáta	Pätička podbloku

Obr. 1.20: Štruktúra podbloku.

Bajt 3	Bajt 2	Bajt 1	Bajt 0
Magické číslo 0xEA	Rezervované	ID Metadát	

Obr. 1.21: Hlavička podbloku.

Bajt 3	Bajt 2		Bajt 1	Bajt 0
Magické číslo 0xEB	Hlavný tkeep	Tkeep metadát	Počet slov metadát	

Obr. 1.22: Pätička podbloku.

FEC Metadáta podblok

FEC podblok môže obsahovať dva rôzne metadáta identifikátory a to ID 0x0011 pre dáta, ktoré neobsahujú FEC a ID 0x0010, ktoré je generované keď sa FEC nachádza v dátach. Samotné metadáta podbloku obsahujú informácie o dekódovaní FEC *Code Words* (CW).

	Bajt 3	Bajt 2	Bajt 1	Bajt 0
Hlavička	Magické číslo 0xEA	Rezervované	ID Metadát	
Chybová bitová maska	Chybová maska neopraviteľných CWs 0 do 31			
	..... Chybová maska neopraviteľných CW (N-1)*32 do N*32-1			
Počítadlá0	Všetky CWs		Rezervované	
Počítadlá1	Neopraviteľné CW		Opraviteľné CW	
Počítadlá2	Rezervované			
Počítadlá3	Počítadlo opraviteľných chýb			
Pätička	Magické číslo 0xEA	Flagy	Počet slov podbloku	

Obr. 1.23: FEC podblok.

2 Wireshark

Wireshark, známy aj pod svojím pôvodným názvom Ethereal, je open-source sieťový analyzátor, ktorý umožňuje zachytávať a analyzovať sieťový dátový tok v reálnom čase. Jeho schopnosť zachytiť a zobraziť sieťovú prevádzku na úrovni jednotlivých paketov umožňuje používateľom detailne sledovať komunikáciu medzi jednotlivými zariadeniami a identifikovať potenciálne problémy v sieti. Bez ohľadu na to, či sa jedná o diagnostiku problémov s pripojením, monitorovanie bezpečnostných hrozieb alebo analýzu výkonnosti aplikácií, Wireshark poskytuje bohatú súčinnosť a informácie na úrovni, ktorú môžu používatelia využiť na optimalizáciu svojich sietí a procesov.[13]

V tejto kapitole sa nachádza popis *PCAP Next Generation* (PcapNG)[4] formátu a potrebných blokov k jeho vytvoreniu. Formát PcapNG je možné zobraziť pomocou aplikácie Wireshark, avšak existuje viacero formátov pre uloženie sieťovej prevádzky, ktoré aplikácia Wireshark podporuje. Pre vytvorenie PcapNG súboru bola spracovaná knižnica Python-pcapng, ktorá poskytuje podporu vytvorenia PcapNG súboru v programovacom jazyku Python. Nakoniec obsahuje spracované informácie o vytvorení pluginu v programovacích jazykoch Python, LUA a C.

2.1 PcapNG formát

Tento text je založený na informáciách z neoficiálnej online dokumentácie štandardu [14] a z Wireshark dokumentácie [15]. PcapNG formát je súborový formát pre zachytávanie sieťovej komunikácie navrhnutý pre prekonanie limitov originálneho Pcap formátu ako napríklad možnosť uschovania paketov s rozličným typom linkovej vrstvy. PcapNG formát vznikol okolo roku 2004. Ďalšie užitočné funkcie PcapNG formátu sú :

- viacero PcapNG súborov môže byť zlúčených dohromady,
- umožňuje pripojenie komentárov k paketu,
- obsahuje navrhnuté dátové štruktúry pre uloženie metadát ako sú napríklad hostname, šifrovacie kľúče, zachytávacie filtre a informácie o použitom rozhraní.

2.1.1 Section Header Block (SHB)

Každý pcapng súbor musí začínať s SHB. SHB sa skladá z typu bloku, ktorý má konštantnú hodnotu `0x0A0D0D0A` hexadecimálne. Táto hodnota je palindróm. Za ním sa nachádza dĺžka bloku čiže SHB. Ďalej nasleduje bytové poradie, Endianness a má hodnotu `1A2B3C4D`. Ďalej nasledujú majoritná a minoritná verzia, aktuálna a

jediná verzia je 1.0. Ďalej nasleduje dĺžka sekcie, je to 64 bitové číslo. A ako posledné sa v SHB môžu nachádzať voliteľné parametre.

2.1.2 Interface Description Block (IDB)

Najčastejšie sa za SHB nachádza blok IDB. IDB obsahuje metadáta o použitom sieťovom rozhraní k zachytávaniu paketov. PcapNg súbor môže obsahovať niekoľko IDB pre každé použité rozhranie. Nové IDB bloky sa môžu nachádzať kdekoľvek v PcapNg súbore. IDB začína s typom bloku, ktorý má hodnotu `0x00000001` hexadecimálne. Za ním sa nachádza typ linkovej vrstvy. Ďalej nasledujú rezervované 2 bajty a za nimi maximálna veľkosť paketu. Hodnota je najčastejšie 65535 bajtov alebo 256 kilobajtov. Ako posledné sa môžu nachádzať voliteľné parametre. Typicky obsahujú informácie o sieťovom rozhraní ako názov rozhrania, IP adresu, MAC adresu rozhrania, operačný systém atď.

Typ linkovej vrstvy

Hodnota definuje typ paketov, ktorý bol zachytený sieťovým rozhraním, napríklad známy ethernet alebo WiFi[16].

2.1.3 Paketové bloky

PcapNG formát definuje 3 paketové bloky, v ktorých sú uschované pakety a sú nimi paketový blok, jednoduchý paketový blok a rozšírený paketový blok. Paketový blok nie je odporúčané používať je označený ako "zastaralý". V jednoduchom paketovom bloku nie je možné uschovať metadáta ako časové razítko paketu čo nie je ideálne a preto je najlepšie použiť rozšírený paketový blok, ktorý to dokáže.

Rozšírený paketový blok (EPB)

EPB obsahuje ako prvé typ bloku, ktorý má konštantnú hodnotu `0x00000006` hexadecimálne. Ďalej obsahuje celkovú dĺžku bloku, ID rozhrania v systéme (počítané od nuly), hornú a dolnú hranicu časového razítka, dĺžku zachyteného paketu, naozajstná dĺžka paketu. Naozajstná dĺžka paketu sa môže líšiť od dĺžky zachyteného paketu v prípade, že bola použitá dlhšia veľkosť maximálneho paketu v IDB. Ďalej nasledujú samotné paketové dáta a za nimi môžu nasledovať voliteľné parametre. Obvykle poznámku, ktorá má kód `opt_comment`.

2.2 Python-pcapng

Python-pcapng[1] je knižnica pre parsovanie a vytváranie Pcap-NG formátu, používaného v nových verziách dumpcap a podobných nástrojov ako Wireshark, winpcap a dalšie. Knižnica je verejne dostupná a môže byť nainštalovaná pomocou príkazu `pip` pod názvom python-pcapng. Knižnica obsahuje čítač PcapNG súborov, ktorý rozpozná PcapNG formát a následne je možné pracovať s jednotlivými blokmi súboru. PcapNG súbor je možné vytvoriť z pred pripravených štruktúr blokov, ktoré odpovedajú formátu PcapNG špecifikovaného v sekcií 2.1 a nachádzajú sa v zdrojom súbore blocks. [17]

2.3 Vytvorenie pluginu (dissector) pre wireshark

Dissector môže byť built-in pre Wireshrak alebo napísaný ako samo registrujúci plugin. Výhoda pluginu je v rýchlej znovu kompilácií pri jeho vytváraní. Pri build-in dissectore by bolo potrebné znovu zostaviť balíček Wireshark čo je oproti pluginu značne dlhšie. [18]

2.3.1 Jazyk Python

Oficiálna podpora vytvorenia pluginu v jazyku Python pre Wireshark bola v roku 2014 zrušená. Existuje alternatíva knižnica pyreshark, ktorá by mala byť schopná túto podporu vytvoriť. Knižnica nie je oficiálne podporovaná spoločnosťou Wireshark a je možné ju nájsť vo verejnom repozitári[2] GitHub pod účtom ashdnazg. [19]

2.3.2 Jazyk Lua

Wireshark obsahuje natívnu podporu drop-in pluginu napísaného v jazyku LUA. V jednoduchosti sa vytvorený LUA súbor vloží do príslušného adresára a pri zapnutí programu Wireshark je vložený LUA skript skompilovaný a následne, pri správnej implementácií, načítaný ako plugin. Príslušný adresár je možné nájsť v menu Help -> About Wireshark -> Folders pod názvami Personal Lua Plugins, Global Lua Plugins. Pri zmene LUA súboru je potrebné reštartovať program alebo znovu načítať všetky LUA scripty pomocou skratky `CTRL+SHIFT+L`. [20]

[1] `https://pypi.org/project/python-pcapng/`
[2] `https://github.com/ashdnazg/pyreshark`

2.3.3 Jazyk C

Wireshark je naprogramovaný v jazyku C a dissectory sú zvyčajne taktiež napísané v jazyku C. Dissector napísaný v jazyku C je rýchlejší oproti dissectoru v jazyku LUA. Pre kompilovanie vytvoreného skriptu je potrebné stiahnuť zdrojový kód Wiresharku, vytvoriť adresár s názvom dissectoru v príslušnej zložke, umiestniť skript do vytvoreného adresára, vytvoriť v adresári `CMakeLists.txt` súbor pre špecifikáciu kompilácie a nakoniec zostaviť balíček Wireshark pomocou programu CMake. Výsledný DLL súbor sa bude nachádzať v adresári `plugins`. [21]

3 Prevod binárnych dát na PcapNG

Pre prevod zachytených dát sieťovým analyzátorom, špecifikovaným v sekcií 1.8, je potrebné dekódovať hlavičky a pätičku vytvorené pri zachytávaní komunikácie. Skramblovanie a poprednú korekciu chýb nie je potrebné vykonávať softwérovo, pretože pri spracovaní dát analyzátorom sa príslušný hardvér postará o dekódovanie dát. Pre spracovanie paketu sieťového analyzátora bol vytvorený nástroj pre príkazovú riadku v jazyku Python s využitím knižnice Python-pcapng (viz. sekcia 2.2), pre vytvorenie PcapNG súboru s dátami XG-PON rámcov.

3.1 Type Hints

V kóde sú využité Python Type Hints pre argumenty a návratové hodnoty funkcií. Tento prístup poskytuje viacero výhod vrátane zlepšenej čitateľnosti kódu a kontroly jeho správnosti. Typovanie návratových hodnôt umožňuje jednoduchú identifikáciu očakávaného typu výstupu. Typovanie argumentov funkcií uľahčuje pochopenie očakávaných vstupov.

Výpis 3.1: Príklad použitia Type Hints v definícií funkcie.

```
def start(file_paths: list,out_file: str,ploam_only: bool) -> None:
```

Výhody použitia Type Hints

- **Zlepšená Čitateľnosť**: Type Hints umožňujú ľahšiu identifikáciu typov vstupov a výstupov funkcií, čo zlepšuje zrozumiteľnosť kódu.
- **Kontrola Kódu**: Type Hints pomáhajú zachytiť chyby typov počas vývoja, čo vedie k robustnejšiemu a spoľahlivejšiemu kódu.
- **Dokumentácia**: Anotácie typov poskytujú užitočnú dokumentáciu priamo v kóde, čo uľahčuje jeho pochopenie a používanie pre ostatných vývojárov.

[22] [23]

3.2 Nástroj pre príkazovú riadku

Nástroj prijíma ako potrebný argument relatívnu cestu ku binárnemu súboru obsahujúci XG-PON rámce zachytené sieťovým analyzátorom. Program dokáže spracovať jeden alebo viac argumentov relatívnej cesty. Program očakáva reťazec `ds` v názve súboru, pre súbor obsahujúci zachytenú komunikáciu v zostupnom smere a reťazec `us`, pre súbor so zachytenou komunikáciou vo vzostupnom smere. Nástroj prijíma

nepovinné parametre pomocou prepínačov. Dostupné prepínače sú *-p*, *-t*, *-o*. Pomocou prepínača *-p* program vytvorí výsledný PcapNG súbor s rámcami v zostupnom smere, ktoré obsahujú aspoň jednu PLOAM správu. Použitie prepínača *-o* očakáva následný argument názov súboru, ktorý bude obsahovať spracované výstupné dáta vo formáte PcapNG. Súbor bude vytvorený v aktuálnom adresári. Ak prepínač nie je použitý, výsledný súbor bude mať názov `xgpon.pcapng`. Pri použití prepínača *-t*, program na začiatku behu vykoná test endianness, ktorý je vypísaný na štandardný výstup. [24]

Výpis 3.2: Výpis použitia nástroju.

```
usage: PcapNGPaketParser [-h] [-o OUT] [-t] [-p] filepaths [filepaths ...]

Program decodes data packet from optical network analyzer and store them in PcapNG
    file.

positional arguments:
  filepaths           Files names have to contain 'ds' for downstream and 'us' for
    upstream frames

optional arguments:
  -h, --help          show this help message and exit
  -o OUT, --out OUT   Specifies name of output file. File name should contain
    extension .pcapng
  -t, --test          Test for better understanding of endianess
  -p, --ploam-only    Works only for downstream frames
```

3.2.1 Štart programu

Program otvorí výstupný súbor s príslušným názvom a vytvorí SHB blok, definovaný v podsekcií 2.1.1, do ktorého vloží možnosti `shb_os` s hodnotou *python* a `shb_userappl` s hodnotou *python-pcapng*. K SHB bloku je ďalej pripojený blok IDB, definovaný v podsekcií 2.1.2, ktorý obsahuje parameter `link_type` s hodnotou 147. Hodnota 147 je definovaná ako rezervovaný DLT_USER0, na ktorú bude pripojený vytvorený plugin pre XG-PON dáta. Program využíva jediný typ linkovej vrstvy pre oba smery komunikácie. Ďalej sú do IDB bloku pridané možnosti `if_description` s hodnotou *Hand-rolled* a `if_os` s hodnotou *Python 3.9*.

Otvorený výstupný súbor je predaný do konštruktoru novo vytvorenej inštancií triedy `FileWriter` z knižnice python-pcapng. Zároveň je predaný blok SHB, ktorý je zapísaný do súboru. Následne je pre každý definovaný vstupný súbor vytvorená inštancia triedy `ParserToPcapNg`. Z triedy je zavolaná prvá metóda *parse_file*, ktorá začne parsovanie súboru.

3.2.2 Trieda Packets

Trieda `Packets` slúži ako iterátor medzi jednotlivými paketmi v rámci jedného súboru. Trieda prijíma do konštruktoru názov súboru, z ktorého následne načíta dáta a uloží ich do premennej triedy s názvom *data*. Dáta sú uložené do dátového typu *array* z knižnice Numpy[1]. V rámci *Numpy array* je definovaný dátový typ hodnoty v poli ako *uint8*, čo reprezentuje 8 bitové číslo resp. 1 bajt. Pre použitie inicializovanej triedy ako iterátor je potrebné použiť funkciu `iter()`. Pri volaní nasledujúcej hodnoty iterátoru z celých dát sú extrahované prvé štyri bajty, v ktorých sa nachádza dĺžka aktuálneho paketu. Následne je odstránených prvých 16 bajtov a funkcia vracia `Numpy array` s dátami jedného paketu. Po spracovaní všetkých paketov v rámci súboru, funkcia vyvolá výnimku *StopIteration*. Pri použití iterátoru v cykluse, výnimka *StopIteration* ukončí cyklus.

Výpis 3.3: Trieda Packets.

```
class Packets:
    def __init__(self, filepath: str):
         with open(filepath, "rb") as fd:
            self.data = numpy.frombuffer(fd.read(), dtype=numpy.
    uint8)

    def __iter__(self):
        return self

    def __next__(self):
        if not len(self.data): raise StopIteration
        packet_size = int.from_bytes(self.data[:4], 'little')
        self.data = self.data[16:]
        packet = self.data[:packet_size]
        self.data = self.data[packet_size:]
        return packet
```

3.2.3 Trieda ParserToPcapNg

Trieda `ParserToPcapng` obsahuje metódy pre výber, odstránenie a pretypovanie načítaných binárnych dát uložených v bajtoch. Pre výber a odstránenie určitého počtu bajtov je využitý slicing pola. Ďalej trieda obsahuje metódy pre spracovanie hlavičky paketu, pomocných správ, hlavičky rámca a metadát, ktoré sú definované v podsekcií 1.8.2. Taktiež trieda obsahuje metódy pre spracovanie súboru a paketu, overenie počtu PLOAM správ v rámci zostupného smeru a využitie HEC pre korekciu chýb XGTC hlavičky v oboch smeroch komunikácie. Dáta XG-PON rámca sú zapísané

[1] `https://numpy.org/`

v notácií little-endian po štyroch bajtoch a preto trieda obsahuje ešte jednu metódu pre prevod dát do notácie big-endian.

Trieda prijíma do konštruktoru štyri parametre a to cestu k súboru, už vytvorený blok SHB, inicializovanú triedu `FileWriter` a boolean hodnotu pre výber len PLOAM správ z XG-PON dát. V inicializácií sú do premenných triedy uložené prijaté parametre a následne využívané jednotlivými metódami triedy. V rámci inicializácie je vytvorená súkromná premenná *data* s hodnotou *None*, do ktorej sa ukladajú dáta jednotlivých paketov pre spracovanie. Dátový typ dát je očakávaný ako `Numpy Array` s hodnotami *uint8*.

Ako dátový typ dát bol pôvodne použitý typ `bytearray`. Keďže zachytené dáta môžu mať veľkosť až niekoľko jednotiek GB, spracovanie dát pomocou dátové typu `bytearray` trvá viac ako 10 hodín. Preto je v implementácií využitý dátový typ `Numpy Array`[25], ktorý dokáže pracovať s pamäťou oveľa efektívnejšie. Po prevedení dátového typu, dĺžka behu programu sa skrátila na niekoľko minút. Krátky beh programu je dôležitý pre efektívne využívanie nástroju. Nakoľko zápis veľkých dát na disk zaberie programu najväčšiu časť behu, tak samotné spracovanie dát trvá niekoľko sekúnd.

Metódy pre spracovanie paketu sieťového analyzátoru

Metóda pre spracovanie paketovej hlavičky sa postará o overenie magického čísla, ktoré má pevnú hodnotu `0xDFC0`. Po overení magického čísla je hlavička odstránená z dát.

Pre pomocné správy a hlavičku rámca bola vytvorená spoločná metóda zobrazená vo výpise 3.4. Funkčnosť metódy spočíva v overovaní kontrolného bitu (flag), pomocou ktorého je nájdená hlavička rámca. Z hlavičky rámca je následovne odoberané časové razítko, ktoré bude neskôr použité v metadátach jednotlivých rozšírených paketových blokov PcapNG súboru. Pomocné správy a hlavička rámca sú odstránené z dát.

Pri spracovaní metadát sú overované magické čísla v pätičke bloku, pätičke podbloku a hlavičke podbloku metadát. Paket musí na svojom konci obsahovať magické číslo `0xEB`. V prípade že metadáta obsahujú podblok, je overené magické číslo `0xEA`, ktoré sa nachádza v prvom a poslednom slove podbloku. Metadáta sú odstránené z dát.

Výpis 3.4: Metóda spracovania pomocných správ a hlavičky rámca.

```
def decode_aux_frame_header(self) -> numpy.array:
        timestamp = self.__get_bytes_start(8)
        control_byte = timestamp[-1:]
        flag = self.retype_byte_to_int(control_byte) & 0x80
        while flag != 0:
```

```
            self.__delete_bytes_start(32)
            timestamp = self.__get_bytes_start(8)
            control_byte = timestamp[-1:]
            flag = self.retype_byte_to_int(control_byte) & 0x80
        self.__delete_bytes_start(32)
        return timestamp
```

Metódy pre spracovanie paketu a súboru

Metóda pre spracovanie súboru je hlavnou metódou triedy, ktorá je zavolaná pri štarte programu. Metóda začína vytvorením inštancie triedy `Packets` s argumentom cesty k súboru. Následne začne iterovanie medzi jednotlivými paketmi v súbore. Pre každý paket je ako prvá zavolaná metóda pre spracovanie paketu sieťového analyzátoru. Pokiaľ pri spracovaní nastane chyba, paket je zahodený. Pokiaľ dáta rámca nemajú dostatočnú dĺžku pre ďalšie spracovanie, tak je paket zahodený. Pred ďalším spracovaním sú XG-PON dáta rámca prevedené na notáciu big-endian po štyroch bajtoch. Pokiaľ súbor obsahuje rámce zostupného smeru a pri spustení programu bol použitý prepínač *-p* tak je overený počet PLOAM správ v štruktúre Hlend. Pokiaľ aktuálny paket neobsahuje žiadne PLOAM správy tak je zahodený. Podľa typu smeru komunikácie je zavolaná príslušná metóda korekcie chýb, ktorá odstráni opraviteľné chyby. Pred zápisom dát je na začiatok každého paketu pripojený jeden bajt, ktorý slúži pre rozlíšenie smeru komunikácie vo Wireshark plugine. Pre rámce zostupného smeru je použitá hodnota `0x01` a pre rámce vzostupného smeru hodnota `0x02`. Na koniec je vytvorený blok EPB, do ktorého je pridaná časová značka, odobraná z hlavičky rámca, dĺžka paketu a samotné dáta paketu. Vytvorený blok je zapísaný do výstupného súboru pomocou inštancie triedy `FileWriter` do adresára z kadiaľ bol nástroj spustený.

Metóda pre spracovanie paketu uloží do súkromnej premennej triedy dáta paketu, ktoré prijíma cez argument. Následne volá jednotlivé metódy pre spracovanie paketu sieťového analyzátora, ktoré pri akomkoľvek zlyhaní vypíšu príslušnú chybovú hlášku na štandardný chybový výstup. Pokiaľ zlyhá metóda spracovania paketovej hlavičky alebo pomocných správ a hlavičky rámca tak je paket zahodený. Ak zlyhá metóda spracovania metadát paket nie je zahodený. V návratovej hodnote sa môže vyskytnúť časová značka alebo hodnota `None` pri zlyhaní.

Metódy korekcie chýb v XGTC hlavičke

Metódy v oboch smeroch komunikácie sú zamerané na opravu XGTC hlavičky. V prípade zostupného smeru je opravená štruktúra Hlend a každá jedna alokačná štruktúra v rámci BWmap partície. Pri vzostupnom smere je opravená XGTC hlavička

bez PLOAM správy, nakoľko PLOAM správy neobsahujú pole HEC. Opravené polia sú zapísané naspäť k dátam. V prípade, že pole neobsahuje žiadne chyby alebo práve naopak obsahuje tri alebo viac chýb tak dáta nebudú prepísané. Pokiaľ sa v poli nachádzajú chyby, tak príslušné chyby sú vypísané na štandardný chybový výstup. Zároveň pri dvoch chybových bitoch je vykonané overenie parity celého pola a výsledok je taktiež vypísaný na štandardný chybový výstup.

3.3 Hybrid Error Correction (HEC)

Na základe dokumentu [26], je možné využiť binárnu skupinu kódov s N miestami, v ktorých je K informačných miest k vytvoreniu kódu (N,K), ktorý umožňuje opraviť T chýb. V prípade XG-PON HEC je použitý kód (63,12), ktorý dokáže opraviť 2 chyby.

3.3.1 Galois polia (Galois field)

Galois pole je matematický koncept[27] v abstraktnej algebre, ktorý sa zaoberá s konečnými matematickými štruktúrami. Je to množina čísel, ktorá pozostáva z konečného počtu elementov a povoľuje dve operácie a to sčítanie a násobenie, ktoré dodržiavajú určité pravidlá. Pravidlá pre operácie zaručujú, že Galois pole ostane zatvorené, výsledok ktorejkoľvek vykonanej operácie v rámci množiny čísel bude element z tej istej množiny. Veľkosť Galois pola je reprezentovaná prvočíslom p a označenie pola sa zapisuje ako $GF(p)$. Operácia sčítania v Galois poli je ekvivalentná operácií XOR a operácia násobenia je ekvivalentná operácií AND.

Keďže správa pozostáva z binárnych čísel tak je možné využiť binárne Galois pole, ktoré sa zapisuje ako $GF(2^M)$, kde M je možné dostať zo vzťahu $N = 2^M - 1$.[26] V prípade použitia kódu BCH(63,12), bude použité pole $GF(2^6)$, ktoré je dostatočne veľké pre veľkosť N. Prijatá správa o veľkosti 8 bajtov je prevedená na binárne pole, z ktorého je odstránený posledný bit. Po odstránení paritného bitu je vytvorené Galois pole s 63 elementmi.

3.3.2 H matica (Parity Check Matrix)

Na základe generačnej matice (G matica), ktorá obsahuje všetky možné kódové slová, je vytvorená H matica. G matica je definovaná ako $G = [I, P]$ alebo $G = [P, I]$, kde I má veľkosť KxK a P má veľkosť $KxN - K$. Matica I je jednotkovou maticou a matica P je paritná pod matica, ktorá obsahuje nezávislé nenulové vektory z použitého pola $GF(2^M)$. Matica G má veľkosť KxN. Z G matice je možné odvodiť H maticu, podľa vzťahu $G = [I, P] => H = [P^T, I]$. Pod matica P^T má veľkosť

$N-KxK$ a matica I má veľkosť $N-KxN-K$, tak výsledná matica H má veľkosť $N-KxN$. Po vynásobení matíc, kde jedna bude transponovaná je výsledok 0, za použitia vektorového sčítania (XOR). A preto $H * G^T = 0$ alebo $G * H^T = 0$. Keďže poslaný kód je vytvorený pomocou vzťahu $c = m * G$ tak pre jeho overenie je potrebné použiť vzťah $c * H^T = 0$. Pokiaľ je výsledok násobenia nulový poslaná správa je bez chyby a nenulový výsledok je syndrómom chyby.[26][28]

Transponovaná matica H je vytvorená pomocou syndrómov pre chyby na danej bitovej pozícií. Pomocou tabuľky syndrómov 3.1 bola vytvorená matica, kde syndróm prevedený do binárnej sústavy predstavuje jeden riadok matice o veľkosti 12. Pozícia chybového bitu 63 je prvým riadkom matice a pozícia 1 je posledným riadkom matice.

3.3.3 Syndrome Decoding

Na základe znalosti transponovanej matice H je možné vykonať dekódovanie správy. Prijatá správa je vynásobená s maticou H^T a výsledný syndróm je porovnávaný. Pokiaľ je výsledný syndróm 0 tak behom prenosu nenastala chyba a správa je nepozmenená. Pokiaľ je výsledok nenulový, najprv sa porovná výsledok so syndrómami jedno-bitovej chyby a ak je nájdená zhoda, tak je chyba na danej pozícií opravená. Pokiaľ nie je nájdená zhoda tak sa syndróm porovná so syndrómami dvoj-bitových chýb. Syndrómy dvoj-bitových chýb je možné dostať pomocou kombinácie bez opakovania zo syndrómov jedno-bitových chýb, $\binom{63}{2} = 1953$. Výsledný syndróm dvoj-bitovej chyby je vytvorený pomocou vektorového sčítania (XOR), medzi špecifickou kombináciou dvoch jedno-bitových chýb, ktorých pozície poukazujú na chyby v pozícií bitov. Pokiaľ je nájdená zhoda so syndrómom dvoj-bitovej chyby a overenie parity, pomocou paritného bitu, je úspešné, tak sú chyby na daných dvoch pozíciách opravené. V prípade zlyhania overenia parity, správa obsahuje troj-bitovú chybu a je neopraviteľná. Pokiaľ nie je nájdená žiadna zhoda syndrómov, tak správa obsahuje viac ako tri chyby a je neopraviteľná.[29]

Správnosť implementácie bola overená na poskytnutých validných štruktúrach chránených polom HEC z dokumentácií o XG-PON protokole[7]. Z rovnakej dokumentácie bola použitá tabuľka HEC verifikácie pre dekódovanie výsledného syndrómu.

3.3.4 Korekcia správy

Pomocou znalosti chyby na jednej pozícií alebo dvoch pozíciach, je vytvorený chybový vektor, ktorý obsahuje hodnotu jedna na chybových pozíciách a nuly na ostatných pozíciách. Následne je tento vektor sčítaný s vektorom prijatej správy a opravená správa je uložená.

Tab. 3.1: Tabuľka syndrómov.

Pozícia chybového bitu	**Syndróm (base 16)**	**Pozícia chybového bitu**	**Syndróm (base 16)**
63	A9C	47	A09
62	54E	46	F98
61	2A7	45	7CC
60	BCF	44	3E6
59	F7B	43	1F3
58	D21	42	A65
57	C0C	41	FAE
56	606	40	7D7
55	303	39	977
54	B1D	38	E27
53	F12	37	D8F
52	789	36	C5B
51	958	35	CB1
50	4AC	34	CC4
49	256	33	662
48	12B	32	331
Pozícia chybového bitu	**Syndróm (base 16)**	**Pozícia chybového bitu**	**Syndróm (base 16)**
31	B04	15	1DD
30	582	14	A72
29	2C1	13	539
28	BFC	12	800
27	5FE	11	400
26	2FF	10	200
25	BE3	9	100
24	F6D	8	080
23	D2A	7	040
22	695	6	020
21	9D6	5	010
20	4EB	4	008
19	8E9	3	004
18	EE8	2	002
17	774	1	001
16	3BA	0	N/A

4 Plugin pre Wireshark

Pre vytvorenie pluginu bol použitý jazyk LUA, ktorý je prívetivejší pre vývoj. Najčastejší spôsob pripojenia pluginu k Ethener alebo *Internet Protocol* (IP) paketom je cez *User Datagram Protocol* (UDP) alebo *Transmission Control Protocol* (TCP) port. V prípade XG-PON paketu nie je možné využiť tento spôsob, pretože sa nejedná o klasickú štruktúru TCP/IP ale o nový protokol linkovej vrstvy. Protokoly linkovej vrstvy sú definované v tabuľke hodnôt typov hlavičiek linkovej vrstvy, ktorá je zadefinovaná skupinou Tcpdump Group. Ethernet má pridelenú hodnotu 1, ale nachádza sa v nej cez 100 ďalších štandardizovaných typov. Šestnásť hodnôt linkovej vrstvy je rezervovaných pre užívateľské použitie a majú názvy `DLT_USER0 (147)` až do `DLT_USER15 (162)`. Vytvorený plugin je možné v nastaveniach aplikácie Wireshark pripojiť k jednej z rezervovaných hodnôt. Vytvorený PcapNG súbor musí obsahovať rovnakú hodnotu linkovej vrstvy pre spojenie dát s pluginom. Veľkosť PcapNG súboru je limitovaná veľkosťou pamäti pre uschovanie dát, ktorá je 262144 bajtov. [30]

4.1 Pripojenie pluginu na DLT_USER

Pri otvorení PcapNg súboru s definovanou DLT_USER linkovou vrstvou aplikácia Wireshark vyhľadá v tabuľke User DLT, ktorý dissector protokolu použiť. Pre nastavenie tabuľky User DLT, musí byť plugin správne načítaný v aplikácií Wireshark. Pre správne načítanie pluginu je potrebné vložiť vytvorený súbor do príslušného adresáru, ktorý je špecifikovaný v podsekcií 2.3.2. Potom je potrebné prejsť do záložky Edit -> Preferences -> Protocols -> DLT_USER -> Encapsulations Table (Edit). Stlačením tlačidla +, následným výberom požadovaného DLT z menu, pridaním názvu pluginu do stĺpca Payload dissector a stlačením tlačidla OK pre uloženie výberu sa vytvorí daný záznam v tabuľke a pri následnom načítaní súboru bude použitý.[31] Názov vytvoreného pluginu je `xg-pon` a pre spojenie pluginu s vytvoreným súborom, pomocou nástroja pre príkazovú riadku, je nutné vybrať záznam s názvom `User0 (DLT 147)`.

4.2 Vytvorenie pluginu

Pre vytvorenie pluginu je potrebná základná štruktúra zobrazená vo výpise 4.1. Ako prvé je potrebné vytvoriť inštanciu `Proto`, kde je definovaná skratka a plný názov protokolu. Ďalej je možné vytvoriť neobmedzený počet protokolových polí, vytvorením inštancie `ProtoField`, ktorá obsahuje argumenty meno, filter a dátový typ

pola. Inštanciu `Protofield` je možné zavolať s funkciou `new` s nešpecifikovanou dĺžkou daného pola. Inštanciu je možné vytvoriť pomocou funkcií `uint8/16/24/32/64`, ktoré špecifikujú dĺžku pola v bitoch. Vo funkcií je možné využiť ďalšie argumenty ako dátový typ báze, slovník s popisom jednotlivých hodnôt a bitovú masku pre výber špecifických bitov z pola. Pre každú položku protokolu je potrebné vytvoriť jeho pole. Inštancie protokolových polí je potrebné priradiť k inštancií `Proto`, pre ich dostupnosť v rámci funkcie `dissector`. Funkcia `dissector` je hlavnou časťou pluginu, pretože je volaná pri načítaní súboru s dátami protokolu. Jej úlohou je vytvoriť stromovú štruktúru dát, ktorú je možné v aplikácií Wireshark pozorovať. Pridanie listu do stromovej štruktúry je vykonané pomocou metódy `:add()`, ktorá obsahuje argumenty, inštancia triedy `Protofield`, časť pamäti, ktorá je označená vo Wiresharku a hodnota pola. Pokiaľ nie je použitý argument hodnoty tak je hodnota odčítaná z argumentu pamäti. Funkcia `dissector` obsahuje 3 argumenty:

- `buffer` - obsahuje samotné dáta, ku ktorým sa dá pristúpiť pomocou delenia podla offsetu,
- `pinfo` - obsahuje hodnoty stĺpcov listu paketu,
- `tree` - koreň stromovej štruktúry.

Výpis 4.1: LUA plugin - základná štruktúra.

```
local alloc_id_type = {[1] = "XGEM", [255] = "Deallocate this ID"}

xgpon_link_layer = Proto("XG-PON", "10 Gigabit passive optical network")

PSBD = ProtoField.new("PSBd","psbd", ftypes.NONE)
ALLOC_ID_TYPE = \
ProtoField.uint8("alloc_id_type", "Alloc-ID Type", base.HEX, alloc_id_type)

xgpon_link_layer.fields = {PSBD, ALLOC_ID_TYPE}

function xgpon_link_layer.dissector(buffer, pinfo, tree)
    packet_length = buffer:len()
    offset = 0
    pinfo.cols.protocol = xgpon_link_layer.name
    subtree = tree:add(xgpon_link_layer, buffer(), "XG-PON")
    local comtree = subtree:add(PSBD, buffer(offset, 24))
end
```

4.3 Implementácia pluginu

Plugin je navrhnutý na základe zapracovaných informácií o protokole XG-PON, ktoré sa nachádzajú v kapitole 1. Plugin obsahuje inštancie `Protofield` pre všetky spracované hlavičky, payload, pätičku a v nich obsiahnuté polia, ktoré sa nachádzajú v teoretickej časti práce. Zároveň obsahuje polia všetkých validných PLOAM správ. Všetky inštancie `Protofield` obsahujú názov filtru pre ich vyhľadanie v rámci

Wireshark filtrovania paketov. Názvy filtrov všetkých polí sú odvodené od ich oficiálneho anglického pomenovania z oficiálnej dokumentácie XG-PON [7]. Názvy filtrov sú malými písmenami a prevažne obsahujú skratky polí. Pokiaľ názov pola obsahuje medzeru, v názve filtru je medzera nahradená podtržítkom (_) napríklad dvojica *XGTC Header* a *xgtc_header*. Najobvyklejšia číselná reprezentácia polí je decimálna ale taktiež je použitá reprezentácia hexadecimálna a bajtová. Pokiaľ pole obsahuje niekoľko hodnôt so špecifickým popisom hodnoty alebo je indikačným bitom, tak má vytvorenú premennú s dátovým typom slovník, ktorá bližšie špecifikuje vypísanú hodnotu v stromovej štruktúre. Polia, ktoré majú veľkosť menej ako 8 bitov, obsahujú špecifikovanú bitovú masku, podľa ktorej je vybraná príslušná hodnota na pozícií bitov v danom bajte.

4.3.1 Prepojenie smerov komunikácie

Najprv bolo zamýšľané vytvoriť jednotlivé pluginy pre oba smery komunikácie. Každý plugin by bol pripojený na separátnu hodnotu DLT_USER a následne prepojený. Avšak nastal problém s prepojením, pretože nie je možné spojiť dva rozdielne pluginy počas behu programu. Po preskúmaní možností prepojenia, naviazania a vytvorenia pluginu bolo rozhodnuté spojiť oba smery do jedného pluginu a pridať na začiatok dát rámca rozhodujúci bajt pre rozlíšenie smerov komunikácie.

Plugin začne sekvenčne spracovávať jednotlivé pakety s prvým rozhodujúcim bajtom, podľa ktorého sa rozhodne spracovávať zostupný alebo vzostupný smer komunikácie. Ako prvý je pridaný koreň stromu s nápisom pre zostupný alebo vzostupný smer. Pre zostupný smer má PLOAM správa svoje označené miesto a pomocou Hlend štruktúry je možné jasne vidieť keď nie je prítomná žiadna správa alebo je prítomná jedna a viac správ. Avšak prítomnosť PLOAM správy vo vzostupnom smere je podmienená vyvolaním pomocou zostupného smeru. Preto bol v plugine vytvorený slovník, ktorý sleduje chovanie PLOAM správ, BWmap partície a ovláda prítomnosť PLOAM správy a DBRu hlavičky vo vzostupnom smere.

Výpis 4.2: LUA plugin - Alokačný slovník.

```
Allocation_dict = {[1023] = {[1023] = {0, 0, 1}}}
```

Štruktúru slovníku je možné vidieť vo výpise 4.2. Slovník je inicializovaný s jedným záznamom, ktorý reprezentuje broadcast ONU-ID. Prvá kľúčová hodnota obsahuje ONU-ID, ktoré je hlavným ID pre uloženie potrebných informácií. Hodnota kľúču je ďalší slovník, do ktorého je uložené samotné ONU-ID ako kľúč a jeho hodnota je pole o veľkosti tri. Aktívne ONU-ID môže mať pridelené niekoľko Alloc-ID, ktoré budú uložené do hodnoty hlavného kľúču s príslušnými hodnotami. Prvá hodnota v poli je veľkosť alokácie (GrandSize), druhá hodnota je DBRu flag a posledná

hodnota je PLOAMu flag.

4.3.2 Zostupný smer

V zostupnom smere je prvá očakávaná hlavička PSBu. Ďalej pokračuje XGTC hlavičkou, z ktorej je vybraná Hlend štruktúra a jednotlivý počet PLOAM správ a alokačných štruktúr. Podľa počtu je zobrazený dáný počet PLOAM správ a alokačných štruktúr. Z alokačných štruktúr sú vybraté ONU-ID, Alloc-ID a k nim prislúchajúce hodnoty prideleného grantu, ktoré sú uložené do Alokačného slovníku. Pri prítomnosti jednotlivých typov PLOAM správ je zobrazený daný typ správy a v prípade neexistujúcej správy je obsah správy preskočený. Pokiaľ správa očakáva odpoveď PLOAM správy vo vzostupnom smere, tak je do Alokačného slovníku pridaná redundantná hodnota PLOAMu flag pre dané ONU-ID. Za XGTC hlavičkou nasledujú XGEM rámce, ktoré sú spracované pomocou veľkosti XGEM payloadu, ktorá sa nachádza v XGEM hlavičke. XGEM rámce zaberajú ostatnú dĺžku celého rámca.

4.3.3 Vzostupný smer

Vo vzostupnom smere je ako prvá očakávaná XGTC hlavička. Z XGTC hlavičky je vybrané ONU-ID, podľa ktorého je následne pristúpené do Alokačného slovníku. Pokiaľ sa v Alokačnom slovníku nenachádza príslušné ONU-ID program neočakáva PLOAM správu a DBRu hlavičky. Pokiaľ sa v Alokačnom slovníku nachádza slovník pre dané ONU-ID, je z neho vybraná hodnota PLOAMu flag. Pri hodnote 1 je očakávaná PLOAM správa v rámci XGTC hlavičky a pokiaľ je hodnota 0 tak je očakávaná XGTC hlavička bez PLOAM správy. Z indikačného pola v XGTC hlavičke je vybraný flag o stave PLOAM fronty a v prípade prítomnosti PLOAM správy a ONU-ID, ktoré nie je broadcast, je táto hodnota uložená do Alokačného slovníku príslušnému ONU-ID. Podľa typu PLOAM správy je vypísaná príslušná správa a v prípade neexistujúcej správy je obsah správy preskočený. V prípade, že Alokačný slovník je prázdny, program očakáva za XGTC hlavičkou jednu alokáciu bez DBRu hlavičky. Pokiaľ slovník nie je prázdny, tak program vyberie z príslušného slovníku, patriaceho aktuálnemu ONU-ID, všetky pridelené granty a vzostupne, podľa čísla ID, očakáva prítomnosť dát s veľkosťou grantu s prípadnou DBRu hlavičkou. Na konci dát je očakávaná XGTC pätička.

5 Testovanie

Testovanie nástroju a pluginu bolo vykonané na zachytených dátach pomocou sieťového analyzátora (viz. sekcia 1.8) a prístupnej PON sieti v rámci vývojovej skupiny Optolab. Výstupný súbor a samotný plugin boli testované v programe Wireshark.

5.1 LUA konzola

V aplikácii Wireshark je k dispozícii robustný nástroj vo forme LUA konzoly,[32] ktorý je neoceniteľným pomocníkom pri testovaní a vytváraní pluginov. Pomocou LUA konzoly môže byť interaktívne vykonávaných niekoľko skriptov v jazyku LUA priamo v prostredí Wiresharku. Tento nástroj je zvlášť užitočný pri ladení pluginov, ktoré slúžia na analýzu a dekódovanie rôznych sieťových protokolov. S pomocou LUA konzoly môže byť efektívne testovaná funkcionalita pluginov a v prípade potreby môžu byť rýchlo upravované alebo ladené. Týmto spôsobom môže byť zabezpečená spoľahlivá analýza sieťovej prevádzky a môže byť získané hlbšie pochopenie fungovania rôznych sieťových protokolov.

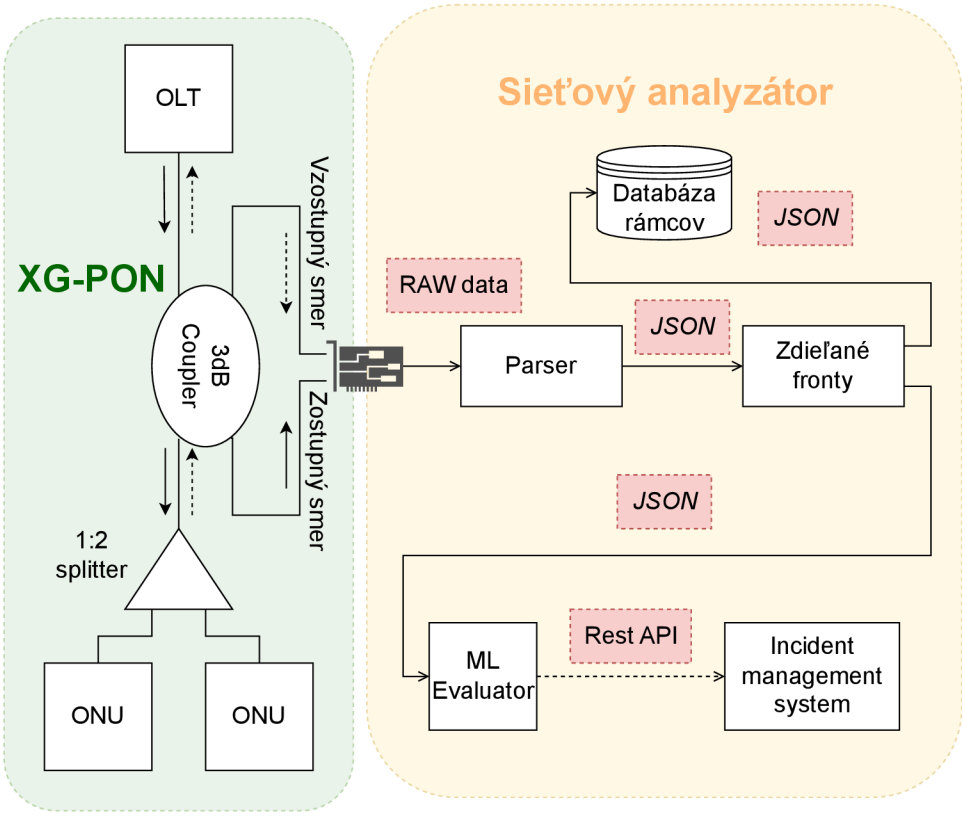
5.2 Použitá PON pre zachytenie dát

Pre zachytenie dát bola použitá XG-PON sieť (viz. obr. 5.1) s jedným OLT a dvomi ONU. Rámce zostupného a vzostupného smeru sú odpočúvané za použitia 2 na 2 3dB väzobného členu (coupler) pre oddelenie zostupného smeru od vzostupného smeru. Oba zachytené signáli sú prevedené z optickej domény do binárnej za použitia sieťovej karty založenej na FPGA dizajne. Samotné schéma analyzátoru odpovedá sieťovému analyzátoru z teórie v sekcií 1.8.[33]

Dostupné ONU-ID pre pridelenie v sieti boli 9 a 11. Pridelené Alloc-ID pre ONU-ID 9 boli 2569 a 3081 a pre ONU-ID 11 bolo 1035. OLT ponúkalo štyri možné profily pomocou PLOAM Profil správ. Použitý profil pre všetky pridelené granty bol profil s indexom 1. Profil s indexom 1 obsahuje verziu profilu 2, vyžaduje použitie FEC, veľkosť delimiteru je 4 a jeho hodnotu 0xA56679E, veľkosť preamble 8 a jeho hodnotu 0xAAAAAAAA.

5.2.1 Zachytené dáta

Pre testovanie pluginu a nástroju pre spracovanie dát boli využité 2 typy dát. Prvá skupina dát obsahuje dáta s ONU aktiváciou. Druhá skupina dát obsahuje podrobnejší prehľad rámcov vzostupného smeru. Keďže sú jednotlivé signály smerov spracovávané osobitne, tak rámce zostupného smeru sa nachádzajú v jednom binárnom



Obr. 5.1: Diagram pripojeného analyzátoru do použitej XG-PON sieti.

súbore a rámce vzostupného smeru v druhom.

Prvá skupina dát

Dáta obsahujú štyri na sebe nezávislé aktivácie dvoch prístupných ONU v PON. Jednotlivé súbory obsahujú v názve písmená `ds` pre označenie súboru s rámcami zostupného smeru a `us` pre označenie súboru s rámcami vzostupného smeru. Pre rozlíšenie aktivácií, jednotlivé súbory obsahujú v názve skratku `act` doplnenú o číslo jednotlivého zachytenia. Na začiatku každého paketu sa nachádza 16 bajtov, z ktorých prvé štyri obsahujú veľkosť daného paketu v bajtoch.

Dáta zostupného smeru neobsahujú metadáta z protokolu zapuzdrenia paketov. Avšak hlavička paketu, pomocné správy a hlavička rámca je prítomná. Preto v rámci implementácií nástroju, metadáta nie sú povinnou súčasťou paketu. Uložené dáta rámca v PcapNG súbore boli ďalej testované v programe Wireshark za pomoci vytvoreného pluginu. Pri prvom testovaní dát nebolo možné zosynchronizovať Hlend štruktúru rámca s vytvoreným pluginom. Toto zistenie viedlo k vytvoreniu dekódovania HEC. Za použitia dekódovania HEC bola úspešne nájdená Hlend štruktúra a použitá notácia endianness, v ktorej boli dáta rámca uložené. Následne bol plugin

upravený aby správne spracoval dáta rámca zostupného smeru. Množstvo prítomných paketov je okolo 200000 a veľkosť na disku je v jednotkách GB.

Dáta vzostupného smeru obsahujú všetky štruktúry protokolu zapuzdrenia paketov. Množstvo prítomných paketov oproti zostupnému smeru je výrazne nižšie. Konkrétne prvá aktivácia obsahuje 7 paketov. Pri zobrazení dát rámcov nesedelo číslo ONU-ID, ktoré sa nachádza v XGTC hlavičke. Po bližšom skúmaní celého paketu s dátami vzostupného smeru bolo zistené, že FEC pod blok metadát obsahuje hodnoty s neopraviteľnými CW (viz. obr. A.2). Zároveň nebolo jasné kedy sa v XGTC hlavičke nachádza PLOAM správa, akú má veľkosť daný grant a kedy je prítomna alokačná hlavička DBRu. Preto bol vytvorený Alokačný slovník, ktorý tieto hodnoty získa z BWmap partície a odpočúvania PLOAM správ, z rámcov zostupného smeru, aby bola jasne daná prítomnosť polí. Po konzultácií neopraviteľných CW bola vygenerovaná nová skupina dát.

Druhá skupina dát

Druhá skupina pozostáva z jednej komunikácií zostupného a vzostupného smeru. Rámce zostupného smeru sú porovnateľné s prvou skupinou dát. Dáta zostupného smeru obsahujú štyri PLOAM Profil správy. Dáta vzostupného smeru sa od prvej skupiny líšia v množstve prítomných rámcov. Veľkosť dát a množstvo prítomných rámcov sa blížia k rovnakému číslu, narozdiel od prvej skupiny kde množstvo rámcov vzostupného smeru je výrazne nižšie. Na základe údajov zo zobrazených dát bol ošetrený prístup k nulovým hodnotám Alokačného slovníku. Pri absencií grantov v Alokačnom slovníku, plugin spracuje dáta bez PLOAM správy v XGTC hlavičke a bez alokačnej hlavičky DBRu. Niektoré dáta rámca boli nulové alebo nedostatočné veľké pre overenie HEC v rámci XGTC hlavičky. Preto bola vytvorená kontrola veľkosti pri spracovaní paketu.

5.3 Poradie paketov v súbore

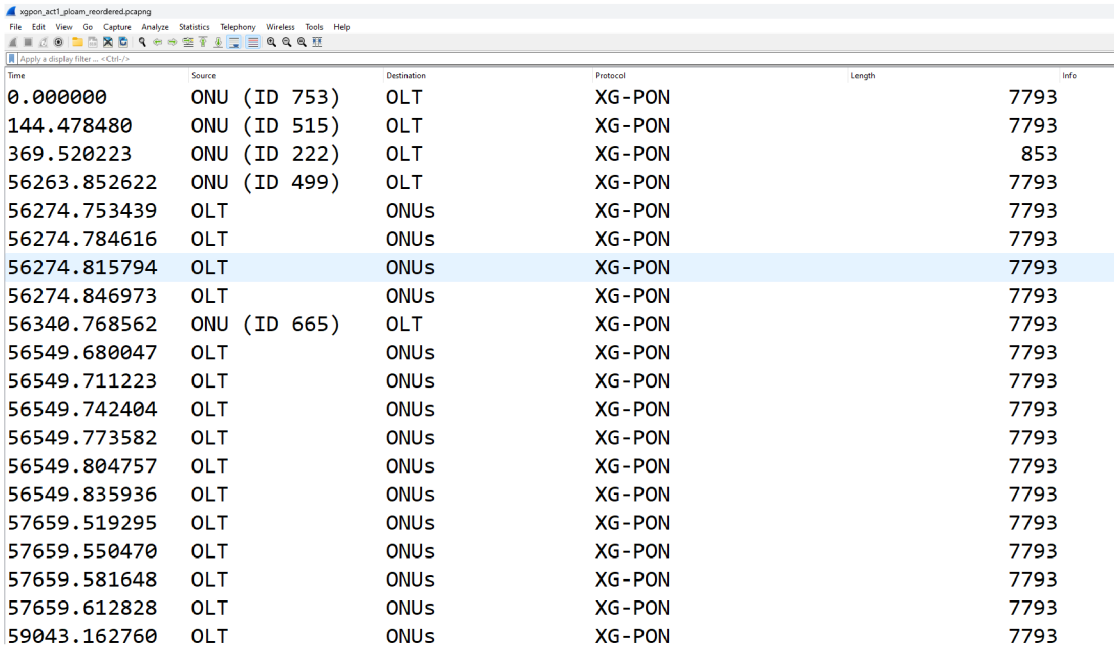
Pri použití dát oboch smerov v nástroji pre spracovanie paketov je výsledný súbor nezoradený, pakety sa nachádzajú v poradí v akom boli vložené do súboru. Pre zoradenie paketov v rámci súboru je možné použiť nástroj príkazovej riadky s názvom `reordercap`. Nástroj sa výlučne nachádza v operačnom systéme Linux po nainštalovaní balíku Wireshark. Nástroj prijíma ako dva potrebné argumenty cestu k danému PcapNG súboru a názov výstupného súboru, ktorý bude obsahovať zoradené pakety podľa časových značiek paketov. Príklad použitia je možné vidieť vo výpise 5.1.

Výpis 5.1: Príklad použitia nástroju reordercap.

```
reordercap xgpon.pcapng xgpon_reordered.pcapng
```

6 Výsledky

Z obdržaných dát bolo vygenerovaných niekoľko `pcapng` súborov, ktoré boli následne zobrazené v programe Wireshark. Z prvej skupiny boli použité všetky štyri aktivácie pre vygenerovanie jednotlivých súborov, ktoré obsahujú rámce zostupného smeru s PLOAM správou a všetky rámce vzostupného smeru. V súboroch sú jednotlivé pakety zoradené podľa časového razítka paketu.



Obr. 6.1: Pakety súboru prvej aktivácie zobrazené v aplikácií Wireshark.

Rámec zostupného smeru obsahuje v stĺpci `Source` reťazec OLT, ktorý znázorňuje vyslanie paketu zariadením OLT. V stĺpci `Destination` obsahuje reťazec ONUs, ktorý reprezentuje broadcast správu doručenú všetkým zariadením ONU v rámci PON. Rámec vzostupného smeru obsahuje v stĺpci `Source` reťazec ONU, za ktorým nasleduje zátvorka s reťazcom ID a číslom ONU-ID patriace danému ONU, ktoré správu vyslalo. V stĺpci `Destination` obsahuje reťazec OLT, pretože správa je zaslaná zariadeniu OLT. Pomocou oboch stĺpcov je možné jasne rozoznať smer komunikácie paketu. Všetky pakety obsahujú v stĺpci `Protocol` reťazec XG-PON ako názov použitého protokolu v dátach. Stĺpcec `Time` obsahuje jednotlivé časy vyslania paketu, relatívne k prvému paketu. Stĺpec `Length` obsahuje aktuálnu dĺžku rámca v bajtoch s rozhodujúcim bajtom.

V obr. 6.2 je zobrazený rámec zostupného smeru z prvej aktivácie. Rámec pozostáva z hlavičky PSBd, XGTC hlavičky s jednou alokačnou štruktúrou a jednou PLOAM správou, XGTC payloadu s Nečinnými rámcami. Koreň rámca obsahuje vo

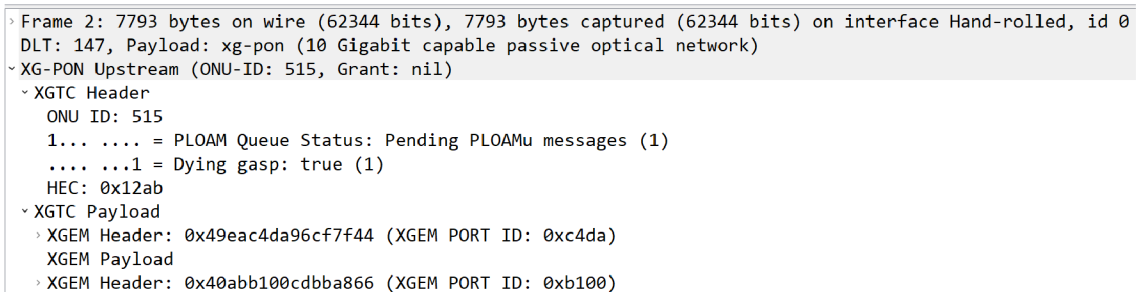
svojom názve počet jednotlivých PLOAM správ a alokačných štruktúr pre lepší prehľad obsahu rámca. V BWMap partícií je možné vidieť pridelený grant s Alloc-ID 9, ktorého veľkosť je 4 slová a neobsahuje alokačnú hlavičku DBRu ani PLOAM správu. Počiatočný index profilu je 0, ktorý je následne zmenený na index 1. Záznam s Alloc-ID 9 je pridelený po priradení ONU-ID a vyrovnaní oneskorenia pomocou PLOAM správ. Typ prenesenej PLOAM správy je Ranging_Time (0x04) a je zaslaný pre ONU s ID 11. Všetky PLOAM správy obsahujú svoje meno v názve listu PLOAM správy. Z bitovej masky je zrejmé, že netreba vyrovnávať oneskorenie. XGEM hlavička obsahuje vo svojom názve XGEM Port-ID a jeho hodnotu pre lepší prehľad XGEM rámca.

```
> Frame 31: 7793 bytes on wire (62344 bits), 7793 bytes captured (62344 bits) on interface Hand-rolled, id 0
  DLT: 147, Payload: xg-pon (10 Gigabit capable passive optical network)
v XG-PON Downstream (BWmap length: 1, PLOAM count: 1)
  v PSBd
      PSYNC: 0xc5e51840fd59bb49
    > SFC structure: 0xff4dcf784d541ad8
    > PON-ID Structure: 0x90310000002c241b
  v XGTC Header
    v HLend Structure: 002039df
        BWmap length: 1
        PLOAM count: 1
        HEC: 0x19df
    v BWmap
      v Allocation Structure 1
          Alloc ID: 9
          .... ..0. = FLAG DBRu: false (0)
          .... ...0 = FLAG PLOAMu: false (0)
          Start Time: 0
          Grand Size: 4
          0... .... = Forced wake-up indication: false (0)
          .00. .... = Profile Index: 0
          HEC: 0x0aad
    v PLOAMd
      v PLOAM Message 1 (Ranging_Time)
          ONU ID: 11
          Message type ID: 0x04
          SeqNo: 45
        v Bit Mask: 0x01
            .... ...1 = Bit P: Delay is absolute; ignore S (1)
            .... ..0. = Bit S: Positive (Increase Value) (0)
          Equalization Delay: 0x000961ef
          Padding: 00000000000000000000000000000000000000000000000000000000
          Message integrity check (MIC): 0x77d2b43bf8480b94
  > XGTC Payload
```

Obr. 6.2: Zobrazený rámec zostupného smeru.

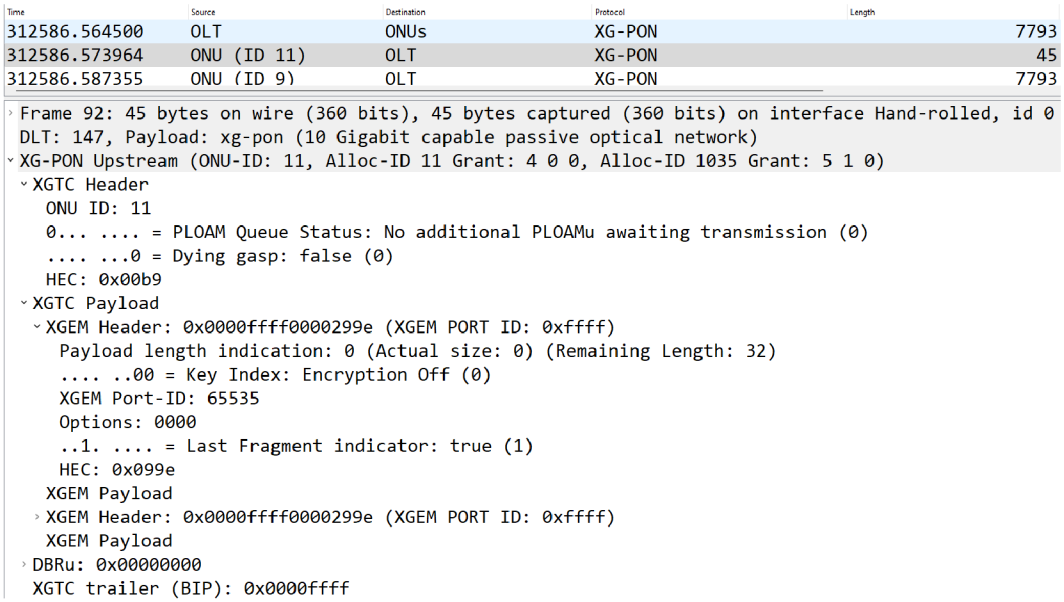
Na obr. 6.3 je zobrazený rámec vzostupného smeru z prvej aktivácie. Koreň vzostupného rámca obsahuje vo svojom názve ONU-ID odosielateľa a pridelené granty k danému ONU-ID. Pokiaľ nie je nájdený záznam s daným ONU-ID v Alokačnom slovníku tak je vypísaná hodnota grantu *nil*.

Z druhej skupiny dát boli vybrané rámce vzostupného smeru, ktoré obsahujú očakávané ONU-ID 9 a 11. Konkrétne bol vybraný rámec s ONU-ID 11 (viz. obr. 6.4), ktorý obsahuje dva pridelené granty. Prvý grant pre Alloc-ID obsahuje veľkosť grantu 4 slová bez alokačnej hlavičky DBRu a bez PLOAM správy. Druhý grant



Obr. 6.3: Zobrazený rámec vzostupného smeru.

s Alloc-ID 1035 obsahuje veľkosť grantu 5 slov s alokačnou hlavičkou DBRu a bez PLOAM správy. V XGEM rámci prvého grantu je možné vidieť dve XGEM hlavičky s veľkosťou payloadu (PLI) nula. Pole PLI obsahuje dodatočné hodnoty vypočítanej dĺžky paylodu a zostávajúcej dĺžky dát v tomto poradí. Podľa XGEM Port-ID s hodnotou 0xFFFF a pola LF s hodnotou 1, je umožné usúdiť že sa jedná o Nečinný XGEM rámec. V rámci druhej alokácie je možné vidieť alokačnú hlavičku DBRu, za ktorou nasleduje XGTC pätička.



Obr. 6.4: Zobrazený rámec vzostupného smeru z druhej skupiny dát.

Záver

Táto práca sa zaoberala návrhom a implementáciou modulu pre aplikáciu Wireshark a návrhom a implementáciou nástroju, ktorý dokáže vytvoriť PcapNG súbor zo spracovaného paketu s XG-PON rámcami.

Po úvodnom preskúmaní princípov XG-PON sietí a ich štruktúry bol navrhnutý formát PcapNG súboru, ktorý umožňuje podporu XG-PON. Boli preskúmané metódy tvorby modulu pre aplikáciu Wireshark. Na základe prieskumu bola vybraná metóda pre implementáciu a vytvorený prototyp modulu pre XG-PON sieť v zostupnom smere. Následne bol prototyp rozšírený aj o vzostupný smer, aby bola pokrytá kompletná funkcionalita modulu.

Dôležitou súčasťou práce bolo overenie funkčnosti modulu na reálnych dátach z reálnej XG-PON siete. Modul bol pomocou dát vyladený, aby bolo možné demonštrovať schopnosť správneho spracovania dát v reálnom prostredí a poskytnúť užitočné informácie pre analýzu a diagnostiku siete.

Implementácia modulu zahŕňala detailné spracovanie štruktúry XG-PON rámcov, paketu sieťového analyzátoru a blokov PcapNG formátu, pričom boli využité existujúce knižnice. Zároveň bolo implementované overenie XG-PON štruktúr pomocou hybridnej chybovej korekcie.

V závere bola overená funkčnosť implementácie celku na reálnych dátach a boli zhodnotené dosiahnuté výsledky. Budúce vylepšenie modulu by zahŕňalo testovanie pomocou všetkých dostupných PLOAM správ, ktoré sa môžu v komunikácií vyskytnúť a spracovanie dát z prevádzky poskytovateľa internetových služieb, vytvorených pomocou XG-PON systému. Ďalšie rozšírenie modulu by zahŕňalo spracovanie enkapsulovaných ethernet rámcov v SDUs a zobrazenie defragmentovaných dát.

Literatúra

[1] Frank J. Effenberger. The xg-pon system: Cost effective 10 gb/s access. *J. Lightwave Technol.*, 29(4):403–409, Feb 2011. URL: https://opg.optica.org/jlt/abstract.cfm?URI=jlt-29-4-403.

[2] Xiuchao Wu, Kenneth N Brown, Cormac J Sreenan, Pedro Alvarez, Marco Ruffini, Nicola Marchetti, David Payne, and Linda Doyle. An xg-pon module for the ns-3 network simulator. 2013.

[3] Vivens Ndatinya, Zhifeng Xiao, Vasudeva Rao Manepalli, Ke Meng, and Yang Xiao. Network forensics analysis using wireshark. *International Journal of Security and Networks*, 10(2):91–106, 2015.

[4] Scott D Fether. Pcap next generation: Is your sniffer up to snuff? *ISSA Journal*, 16(7), 2018.

[5] Lukas Koci, Tomas Horvath, Petr Munster, Michal Jurcik, and Miloslav Filka. Transmission convergence layer in xg-pon. In *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, pages 104–108, 2015. doi:10.1109/TSP.2015.7296232.

[6] Tomas Horvath, Petr Munster, Adrian Tomasov, Petr Dejdar, Vaclav Oujezsky, and Sobeslav Valach. Transmission convergence layer analysis of passive optical networks with a novel fpga card. In *Seventeenth Conference on Education and Training in Optics and Photonics: ETOP 2023*, page 127230R. Optica Publishing Group, 2023. URL: https://opg.optica.org/abstract.cfm?URI=ETOP-2023-127230R, doi:10.1117/12.2668427.

[7] International Telecommunication Union. G.987.3: Gigabit-capable Passive Optical Networks (GPON): Transmission Convergence Layer Specification. Recommendation G.987.3, ITU-T, Január 2014. ITU-T Series G: Transmission Systems and Media, Digital Systems and Networks.

[8] Vivekanand Jha, Rakesh Kumar Singh, et al. Comprehensive performance analysis of dynamic bandwidth allocation schemes for xg-pon system. *Optical Switching and Networking*, 47:100711, 2023.

[9] John S Sobolewski. Cyclic redundancy check. In *Encyclopedia of Computer Science*, pages 476–479. 2003.

[10] Tomas Horvath, Lukas Malina, and Petr Munster. On security in gigabit passive optical networks. In *2015 International Workshop on Fiber Optics in Access Network (FOAN)*, pages 51–55, 2015. doi:10.1109/FOAN.2015.7320479.

[11] Rolando Herrero. Hybrid error correction in fragmented iot media streams. *Transactions on Emerging Telecommunications Technologies*, 33(11):e4601, 2022.

[12] Marek Kváš. *Firmware pro Xavier PON Analyzátor PON sítí*. DFC Design, s.r.o., Strmá 3001/11B, 616 00 Brno, Czech Republic, 2.4 edition, Január 2023.

[13] Jay Beale, Angela Orebaugh, and Gilbert Ramirez. *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier, 2006.

[14] Erik Hjelmvik. Pcapng file format. Accessed on: 2023-11-27. URL: `https://pcapng.com/`.

[15] Wireshark Foundation. Pcapng. Accessed on: 2023-11-26. URL: `https://wiki.wireshark.org/Development/PcapNg`.

[16] N. Nishanth, J. Zareena, and S. Suresh Babu. Pseudo random alteration of sequence numbers (pras): A novel method for defending sessiion hijacking attack in mobile adhoc network. In *2013 15th IEEE International Conference on Communication Technology*, pages 20–25, 2013. `doi:10.1109/ICCT.2013.6820344`.

[17] Samuele Santi. python-pcapng: Python library for reading and writing pcapng files. `https://pypi.org/project/python-pcapng/`, 2023. Accessed on: 2023-11-27.

[18] Wireshark Foundation. Chapter 9. packet dissection. Accessed on: 2023-11-27. URL: `https://www.wireshark.org/docs/wsdg_html_chunked/ChapterDissection.html`.

[19] Wireshark Foundation. Python. Accessed on: 2023-11-28. URL: `https://wiki.wireshark.org/Python`.

[20] Mika Sundland. Creating a wireshark dissector in lua, 2017. Accessed on: 27.11.2023. URL: `https://mika-s.github.io/wireshark/lua/dissector/2017/11/04/creating-a-wireshark-dissector-in-lua-1.html`.

[21] Wireshark Devepolment Team. Wireshark dissector readme. Accessed on: 2023-11-27. URL: `https://github.com/wireshark/wireshark/blob/master/doc/README.dissector`.

[22] Ke Sun, Yifan Zhao, Dan Hao, and Lu Zhang. Static type recommendation for python. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–13, 2022.

[23] Thomas W. Price, Samiha Marwan, and Joseph Jay Williams. Exploring design choices in data-driven hints for python programming homework. In *Proceedings of the Eighth ACM Conference on Learning @ Scale*, L@S '21, page 283–286, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3430895.3460159.

[24] Alfredo Deza and Noah Gift. *Robust Python CLI*. Pragmatic AI Solutions, 2022.

[25] Eli Bressert. Scipy and numpy: an overview for developers. 2012.

[26] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[27] Christoforus Juan Benvenuto. Galois field in cryptography. *University of Washington*, 1(1):1–11, 2012.

[28] Jie Hao, Shu-Tao Xia, Kenneth W Shum, Bin Chen, Fang-Wei Fu, and Yixian Yang. Bounds and constructions of locally repairable codes: parity-check matrix approach. *IEEE Transactions on Information Theory*, 66(12):7465–7474, 2020.

[29] Jacques Stern. A new identification scheme based on syndrome decoding. In *Annual International Cryptology Conference*, pages 13–21. Springer, 1993.

[30] The Tcpdump Group. Link-layer header types. Accessed on: 2024-04-21. URL: https://www.tcpdump.org/linktypes.html.

[31] James Gibbard. Making wireshark link layer dissectors. online Blog. URL: https://www.gibbard.me/wireshark_lua_user_link_layer/.

[32] Andrei Costin. Lua code: security overview and practical approaches to static analysis. In *2017 IEEE Security and Privacy Workshops (SPW)*, pages 132–142. IEEE, 2017.

[33] Tomas Horvath, Petr Munster, Adrian Tomasov, Petr Dejdar, Vaclav Oujezsky, and Sobeslav Valach. Transmission convergence layer analysis of passive optical networks with a novel fpga card. In *Education and Training in Optics and Photonics*, page 127230R. Optica Publishing Group, 2023.

Zoznam symbolov a skratiek

BER Bit Error Rate

CRC Cyclic redundancy check

CW Code Words

DWDM Dense wavelength-division multiplexing

FEC Popredná chybová korekcia

HEC Hybrid Error Correction

IP Internet Protocol

OAM Operations Administration and Management

OLT Optický sieťový terminál

OMCC ONU management and control interface (OMCI) channel

OMCI Management and Control Interface

ONUs Optické sieťové jednotky

PcapNG PCAP Next Generation

PHY Physical interface

PLOAM Fyzická vrstva operácií, administratívy a údržby

SDU Servisná Dátová Jednotka

SDUs Servisné Dátové Jednotky

TCP Transmission Control Protocol

TDM Time-division multiplexing

T-CONT Transmission container

UDP User Datagram Protocol

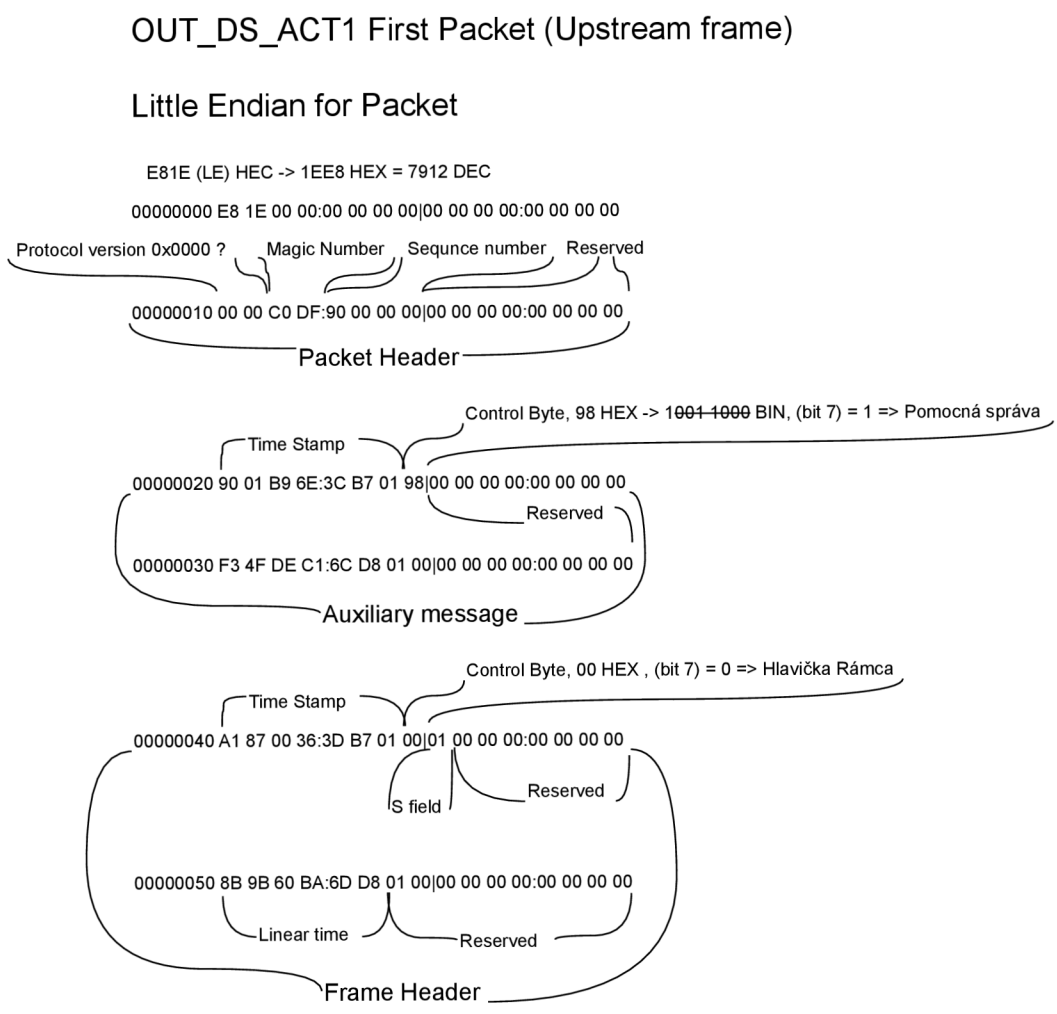
XGEM XG-PON encapsulation method

XGTC XG-PON transmission convergence

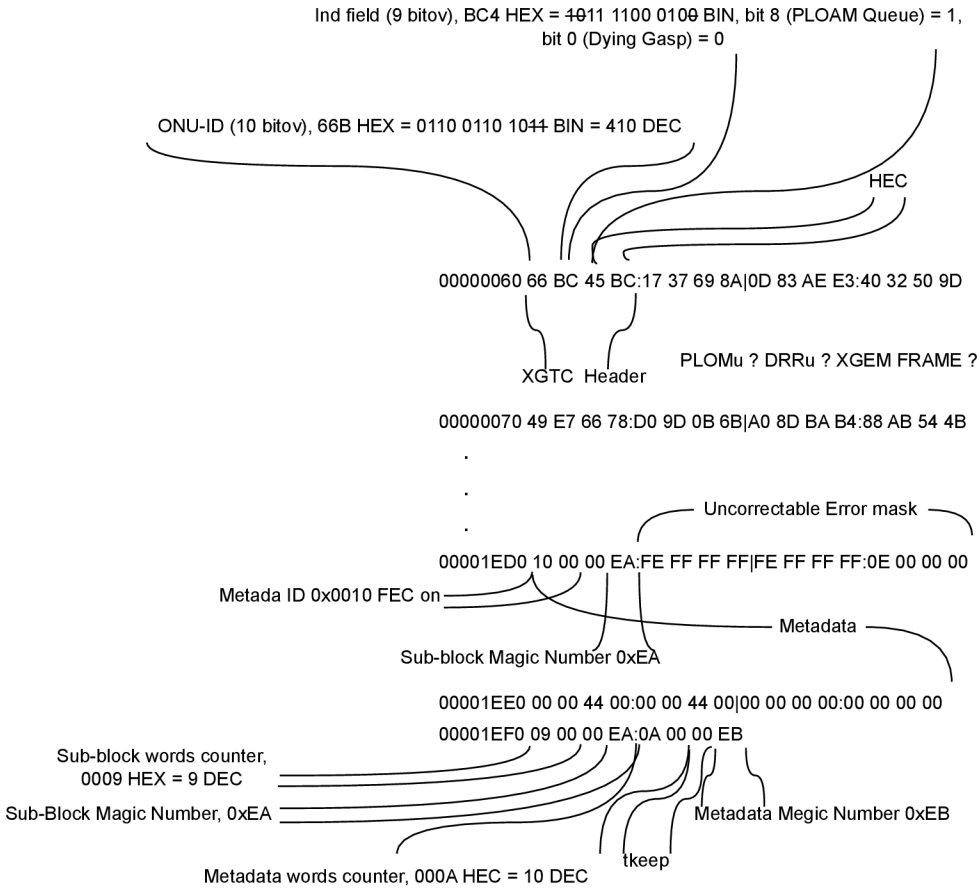
XG-PON 10Gb Pasívna Optická Sieť

A Výpočet hodnôt prvých paketov

A.1 Prvý paket pre Vzostupný smer XGPON zo súboru out_ds_act1.bin

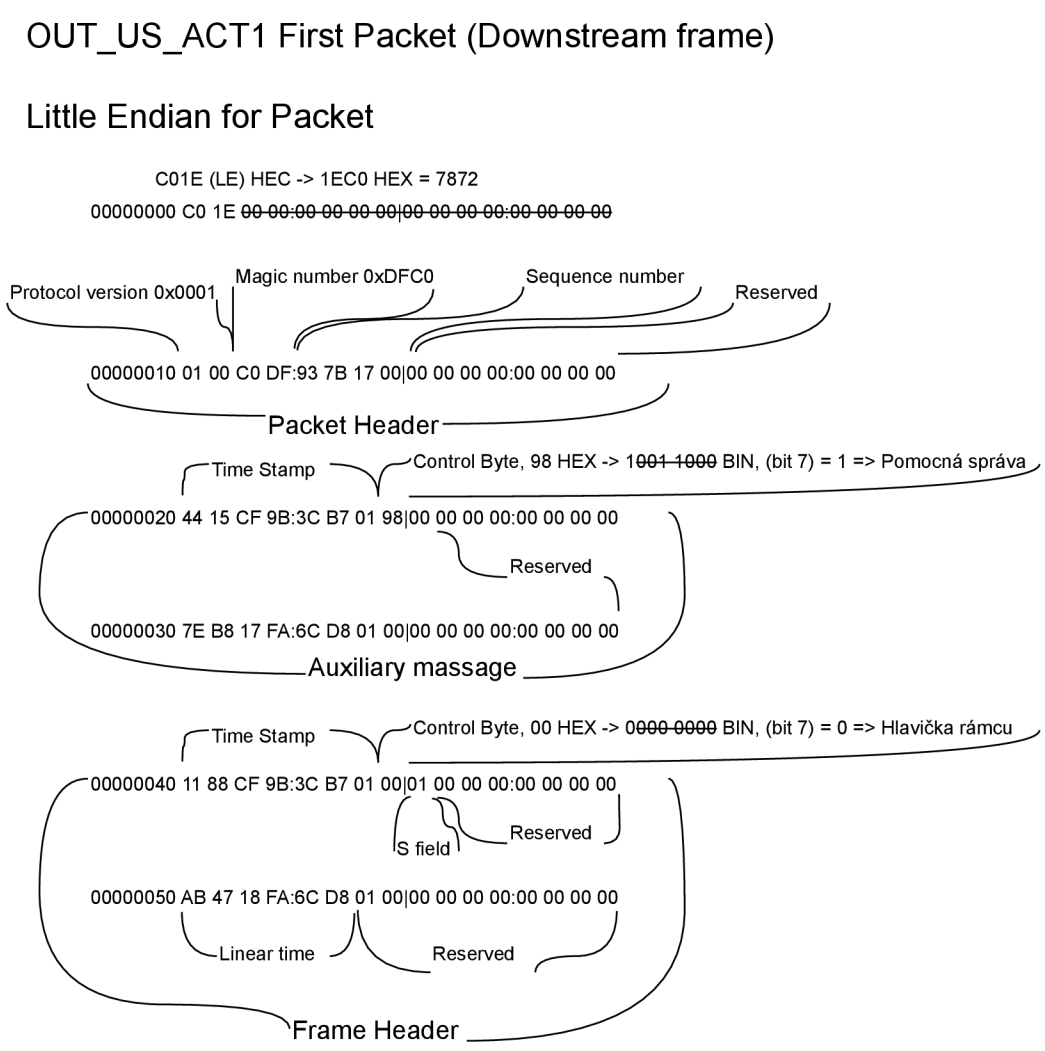


Obr. A.1: Ukážka výpočtu paketu pre vzostupný smer prvá časť.

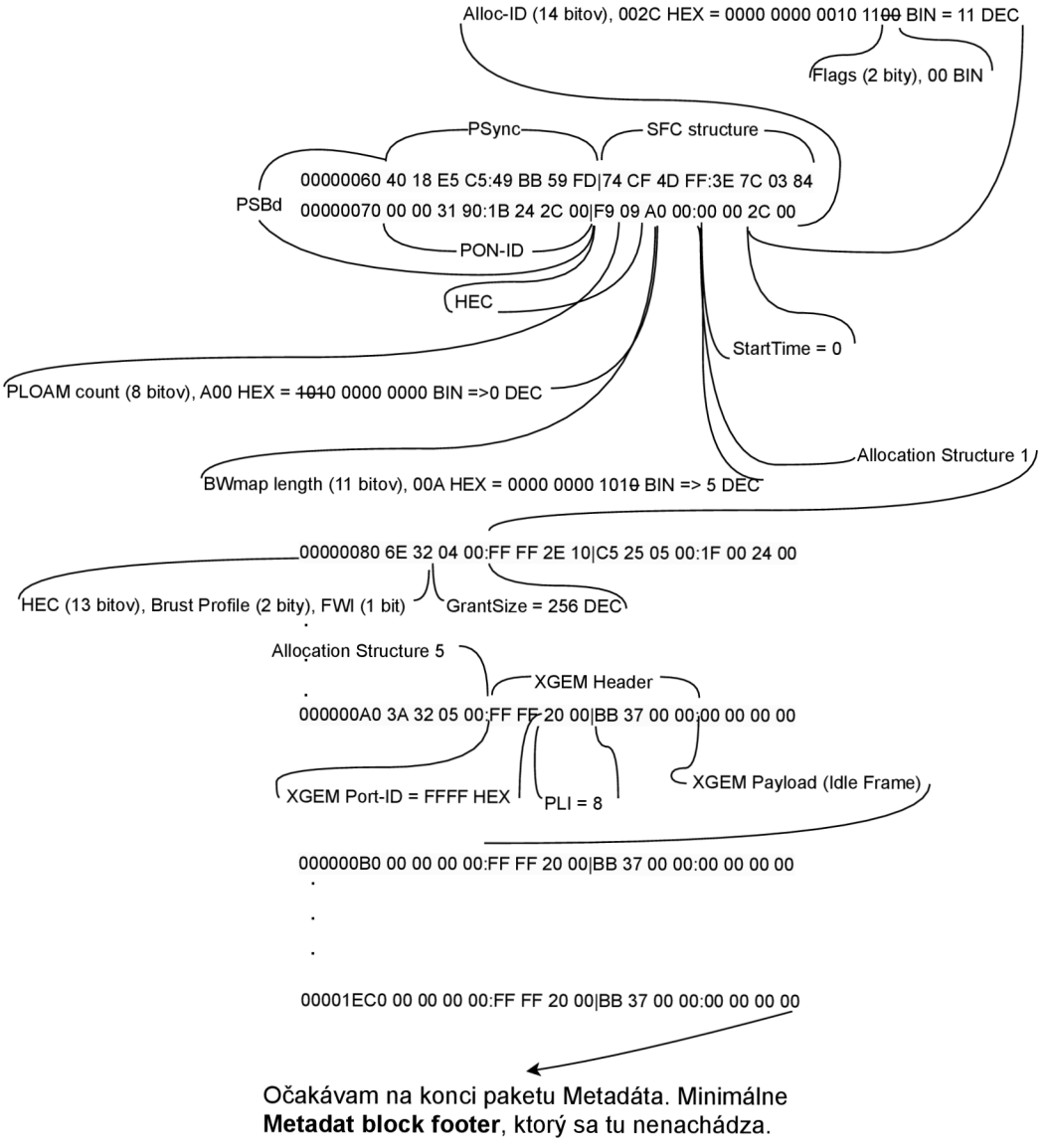


Obr. A.2: Ukážka výpočtu paketu pre vzostupný smer druhá časť.

A.2 Prvý paket pre Zostupný smer XGPON zo súboru out_us_act1.bin



Obr. A.3: Ukážka výpočtu paketu pre zostupný smer prvá časť.



Obr. A.4: Ukážka výpočtu paketu pre zostupný smer druhá časť.