Ako Vyvíjať Driver, Časť 3

V tretej časti lekcie o vývoji Drivera si vysvetlíme, čo je Packet Processor a ako spracúva rôzne typy a kategórie paketov.

Packet Processor je modul Frameworku 4, ktorého úlohou je zabezpečiť logickú vrstvu komunikácie so zariadením. Má na starosti celý životný cyklus paketov vytvorených v Logike zariadenia. To znamená odosielanie paketov a následné párovanie prijímaných paketov, ako aj ich doručenie do správnej časti kódu Logiky zariadenia vyvíjaného programátorom Driveru na ďalšie spracovanie.

Packet Processor taktiež notifikuje Logiku zariadenia o zmenách stavu komunikácie, ako napríklad v situácii, keď externý systém nereaguje.

Poslednou dôležitou úlohou je priamo notifikácia C4 Servera o zmenách stavu v komunikácii so zariadením.

Interná architektúra Packet Processora sa skladá z dvoch hlavných častí.

Prvou je tzv. Packet Queue. Jej úlohou je spravovať všetky pakety, ktoré je potrebné odoslať do zariadenia.

Druhou časťou je Processor, ktorý zabezpečuje na jednej strane odosielanie dát do modulu Link, a na druhej strane prijímanie paketov z modulu Link. Pakety sú následne spracované podľa toho, akého sú typu.

Po tom, ako sa vytvorí Link, pokračujeme vytvorením a nainicializovaním Packet Processora v časti Driver. V rámci inicializácie sa nastavujú parametre Driver Context, Link, ID Bus Controllera, ako aj callback funkcia, ktorá slúži na spracovanie nespárovateľných príchodzích paketov.

Nastavenie eventu OnPacketProcessorStatusChanged poskytuje vývojárovi možnosť notifikácie z Packet Processora a Linku o tom, v akom stave je spojenie. Ak sa spojenie preruší, práve cez tento event dostáva o tom vývojár notifikáciu.

Pozrime sa teraz bližšie na pakety.

V nasledujúcej časti si vysvetlíme, ako sa pakety spracúvajú – zasielajú do zariadenia, aké sú kategórie paketov z pohľadu ich toku, ako aj typy odchodzích paketov z pohľadu čakania na odpoveď.

Pakety v Packet Queue môžu byť spracované tromi rôznymi spôsobmi.

Prvou možnosťou je nadefinovať štandardný odchodzí paket. Ide o jednorazové odoslanie paketu v štandardnej priorite. Takýto paket sa zaradí na koniec fronty a keď príde na rad, tak sa odošle.

Druhou možnosťou je nadefinovať tzv. periodický paket. Ide o novinku pri programovaní Driveru, ktorá výrazne šetrí čas vývojárovi. Umožňuje jedenkrát zaregistrovať paket, ktorý sa následne bude opakovane vykonávať na komunikácii so zariadením, až kým sa komunikácia nezruší. Takýto paket sa odošle, spracuje a znovu sa zaradí do fronty.

Inicializácia takéhoto paketu v zdrojových kódoch je, napríklad v prípade centrálnej jednotky, odosielanie paketu typu GetTimeRequest na sledovanie času v zariadení.

Treťou možnosťou je zadefinovať prioritný paket. Ide o jednorazovo vykonateľný paket, ktorý keďže je zadaný ako prioritný, tak sa zaradí do poradia hneď za paket, ktorý sa práve vykonáva, a bude prioritne odoslaný ihneď po spracovaní aktuálne spracovávaného paketu. Typickými paketmi tohto typu sú príkazy od používateľa.

Prioritný paket sa používa napríklad vtedy, keď chceme odoslať zapnutie alebo vypnutie outputu.

Spôsob, akým Packet Processor narába s paketmi vychádza z konkrétneho typu komunikačného protokolu externého systému.

Z tohto hľadiska rozlišujeme dve kategórie paketov - odchodzie a príchodzie.

Odchodzie pakety sú vytvárané vývojárom v rámci Logiky zariadenia. Príkladom takéhoto paketu je SetOutput paket.

Odchodzie pakety sú oddedené od IOutputBinaryPacket. Každý odchodzí paket musí obsahovať funkciu GetRawBytes, ktorá transformuje dáta paketu do binárnej formy.

Na druhej strane, príchodzie pakety sú vytvárané zariadením. Môžu byť buď odpoveďou na otázky odoslané do zariadenia, alebo ich zariadenie môže posielať samé od seba.

Príkladom takéhoto paketu je OutputStatusResponse paket.

Príchodzie pakety sú definované IInputPacket interfejsom.

Do príchodzích paketov sú v Packet Parseri vložené dáta, ktoré prichádzajú zvonku a príchodzí paket ich stransformuje do tvaru, ktorý následne môže byť používaný v rámci vnútornej Logiky celého Drivera.

Rozlišujeme dva typy odchodzích paketov. Môžu byť buď čakajúce – waitable, čo znamená, že čakajú na odpoveď zo zariadenia; alebo nečakajúce – non-waitable, čo znamená, že Packet Processor po ich odoslaní neočakáva žiadnu odpoveď zo zariadenia.

Teraz si každý z nich popíšeme bližšie.

Keď príde z Logiky zariadenia odchodzí paket, ktorý čaká na odpoveď, Packet Processor ho odošle do zariadenia s tým, že si ho ponechá vo fronte s nastaveným Timeoutom pre odpoveď. Packet Timeout sa nastavuje v parametroch Drivera na Bus Controlleri a hovorí o tom, v akom časovom rozpätí má príchodzí paket prísť naspäť. Ak v tomto rozmedzí príde a v poradí paketov sa nachádza paket, ktorý bol odoslaný a

čaká na odpoveď, tak Packet Processor priradí k príchodziemu paketu originálny odchodzí paket. Aby bolo možné pakety spárovať, každý odchodzí paket, ktorý čaká na odpoveď má na sebe informáciu, ktorá časť Logiky zariadenia ho vytvorila a poslala, a teda kam má byť naspäť zaslaná odpoveď, tzv. callback.

Keď sú pakety spárované, pošlú sa spolu na spracovanie do Logiky zariadenia, a čakajúci paket sa odstráni z fronty.

Ako príklad si môžeme uviesť opakované zisťovanie času na zariadení, kedy odpoveď príde do callbacku OnGetTimeResponse, kde sa následne spracuje.

GetTimeResponseContext je nadefinovaný ako posledný parameter odchodzieho paketu, a je to Context, ktorý následne uvidíme v callbacku. Je to objekt, ktorý vyrába vývojár na to, aby bolo možné spárovať odosielané informácie s prijatými.

Tu vidíme definíciu callbacku OnGetTimeResponse, kam ako parametre prídu príchodzí paket a responseContext, a ú následne sú v tomto callbacku spracované.

Môže nastať situácia, že sa odošle paket, ktorý čaká na odpoveď, ale odpoveď mu v rozpätí zadefinovanom v parametri Packet Timeout na Bus Controlleri nepríde ani na tretíkrát. Takýto typ rozpadnutého spojenia na úrovni Packet Procesora väčšinou znamená poruchu na zariadení.

V takomto prípade je vývojár notifikovaný o tom, že došlo k rozpadu spojenia formou zavolania funkcie OnPacketProcessorStatusChanged. Zároveň Packet Processor odošle log do Systému C4. Avšak Packet Processor sám osebe sa nepokúša o obnovenie spojenia, len o tom informuje vývojára, ktorého úlohou je ďalej naprogramovať obnovu komunikácie po jej rozpade.

Event OnPacketProcessorStatusChanged, ktorý sa zavolá, keď nastanú problémy na spojení, je aktivovaný v rámci inicializácie Drivera. Implementácia tohto eventu je úlohou vývojára, pretože každé zariadenie pristupuje k takejto situácii inak.

V určitých situáciách je potrebné do zariadenia poslať také dáta, na ktoré zariadenie neodpovedá naspäť žiadnym spôsobom, a teda nečakáme na odpoveď zo zariadenia. V takýchto prípadoch sa odchodzí paket odošle a ihneď sa odstráni z fronty.

Sú typy zariadení, ktorých komunikačný protokol je nadizajnovaný tak, že určité typy informácií zariadenie posiela samé od seba. Packet Processor sám o sebe nedokáže vyhodnotiť, kam má poslať prichádzajúce dáta, pri ktorých nemá párovanie na odchádzajúce dáta. Spracuje ich preto takým spôsobom, že všetky takéto neočakávané pakety zo zariadenia posiela do UnhandledPacket callbacku. Ďalej o tom, akého sú tieto dáta typu a akým spôsobom majú byť ďalej spracované už rozhoduje vývojár Logiky zariadenia.

Tu vidíme OnUnhandledPacket callback, ktorý sa nachádza v sekcii Driver.

Úvodná časť lekcie o vývoji Drivera bola venovaná teoretickým základom, druhá časť bola zameraná na modul Link, a v tretej časti sme preskúmali rôzne kategórie a spôsoby spracovania paketov v Packet Processore. Táto lekcia nám tak poskytuje všetky informácie potrebné na integráciu zariadenia na úrovni, ktorá je v Gamanete štandardom.

Ak chceme zariadenie integrovať plnohodnotne, v ďalších lekciách si vysvetlíme, ako naimplementovať ďalšie typy Pluginov a ako ich vzájomne kombinovať tak, aby bolo možné zariadenie plnohodnotne konfigurovať.