Semestrální práce z předmětu Y36SPS

vypracoval Jakub Weberschinke, stduent 2.ročníku FEL, ČVUT

Zadání

Vytvořit v Javě program, který umožní odesílání SNMP paketů, které bude moci zároveň uživatel libovolně modifikovat.

Postup

Abych se vyhnul znovuvynalézání kola a psaní programu od základů, musel jsem nejdříve najít API, ze kterého bych mohl vycházet. Vybíral jsem z těchto kandidátů:

- SNMP4j

  Open Source API používající licenci Apache. Nepochynou výhodou tohoto API je jeho velká flexibilita a možnost pracovat se všemi verzemi protokolu SNMP. Používá však příliš mnoho tříd a celkově je obtížné se v kódu vyznat, pracuje s některými komponentami J2EE.

- iReasoning SNMP

  Komerční API určené pro tvorbu monitorovacích agentů a klientů. API pro klienty a agenty je dodáváno odděleně s oddělenou dokumentací, čímž je přehlednější než předchozí. Vzhledem ke komerční podstatě produktu jsou dodané metody ve formě byte kódu.

- SNMP Packet

  Open Source API menší než předchozí dvě, výrazně přehlednější a intuitivnější. Nepodporuje však šifrované verze, pracuje jen s v1 a v2c.

Nakonec jsem se rozhodl zvolit API od společnosti iReasoning, zejména kvůli tomu, že agenty lze částečně tvořit v grafickém rozhraní a ušetřit si tak programování.

Rozhodl jsem se pro přístup, kdy dotazy budou odesílány na základě vstupů z konzolové aplikace, kam uživatel postupně zadá parametry dotazu. Zde se projevil první nedostatek, neboť se ukázalo, že toto API pracuje se session, do které se nejdříve zadají parametry spojení, tedy IP, port, komunita a verze protokolu. Ta se pokusí ihned s druhou stranou pokusí navázat spojení. Poku by tedy na daném stroji neběžel nějaký SNMP agent, session se nevytvoří a tím pádem nebudeme schopni vytvořit ani paket. Pokud se vytvoří session, předá se řízení odpovídající metodě iReasoning knihovny.

Agent bude pracovat tak, že v momentě, kdy obdrží paket, zobrazí do konzole informaci o paketu, tj. kam byl dotaz směřován a co se očekává jako návratová hodnota. Následně bude čekat na vstup od uživatele, který následně odešle.

Klient byl napsán tak, že od uživatele vyžadoval sekvenční zadávání atributů, na základě kterých se vytvořila session. Následně si uživatel vybere, který typ dotazu zvolí a nad jakým OID bude tento operovat. Pro otestování klienta byl použit jeden ze vzorových agentů pracující s nativním MIB-2 podstromem, konkrétně nad podstromem 1.3.6.1.2.1.2. Dotazy snmp byly úspěšně odeslány a vracely správnou odpověď.

Další část byla vytvoření agenta, který by byl schopen analyzovat příchozí zprávu a odeslat na ní odpověď dle vstupu uživatele. V této části jsem narazil na další nedostatek tohoto API, které prvotní zpracování SNMP paketu (jeho rozbití na javovské objekty) řeší v rámci dodaného byte kódu a proto do ní není možné nijak zasáhnout. Požadavek je tedy možné zpracovat až přímo v jednotlivých metodách, které odpovídají dotazům do konkrétních uzlů MIB stromu. Toto jsem vyřešil tak, že v metodách jednotlivých dotazů byl odkaz na mojí metodu, která zobrazila v konzoli informace o dotazu, které byly propagovány z metody dotazu a vyžádal od uživatele vstup.

Když jsem chtěl celý program otestovat, při kompilaci mi vznikla chyba NullPointerException. Program jsem několikrát prošel, ale nenašel jsem pro vznik této chyby žádný důvod. Při dalších pokusech o odhalení chyby jsem zjistil, že i jedna z ukázkových aplikací, která shodou okolností pracovala se stejným podstromem jako já, vykazuje stejnou chybu. Přitom v kódu tohoto programu jsem nic neupravoval a tento příklad používal stejné knihovny jako ten, na kterém jsem původně testoval klienta.

Při analýze chyby jsem dospěl k závěru, že chyba se musí nacházet v knihovnách dodaných výrobcem. Vzhledem k povaze knihovny však není možné chybu blíže prozkoumat a případně jí odstranit.