VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

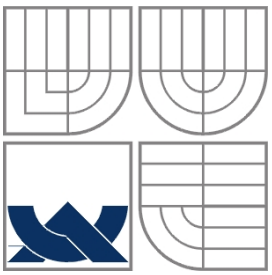
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# ANALÝZA A REKONSTRUKCE KOMUNIKACE TYPU INSTANT MESSAGING (XMPP A MSN)

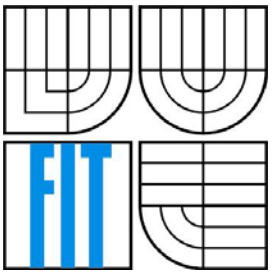
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE ANDREAS ZDUBA
AUTHOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# INSTANT MESSAGING NETWORK ANALYSIS AND RECONSTRUCTION (XMPP AND MSN)

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ANDREAS ZDUBA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. VLADIMÍR VESELÝ

BRNO 2012

# Abstrakt

Tato práce se věnuje zkoumání protokolů sloužích ke komunikaci mezi uživateli. Na na základě analýzy ze zachycené komunikace jsou vybrány nejdůležitější elementy. Výstup práce tvoří experimentální nástroj pro rekonstrukci ze vstupních PCAP souborů a popis parser protokolů v jazyce NPL pro další využití.



# Abstract

This work is devoted to examining the protocols used for communication between users, and based on the analysis of intercepted communications the most important elements are selected. Output consists of experimental work tool for the reconstruction of the input files and PCAP parser description in the language of NPL for future use.



# Klíčová slova

XMPP, MSN, PCAP, rekonstrukce protokolu.



# Keywords

XMPP, MSN, PCAP, protokol reconstruction.



# Citace

Andreas Zduba: Analýza a rekonstrukce komunikace typu Instant Messaging, bakalářská práce, Brno, FIT VUT v Brně, rok 2012

# Analýza a rekonstrukce komunikace typu Instant Messaging

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vladimíra Veselého.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Andreas Zduba
10.5.2012

## Poděkování

Na tomto místě bych rád poděkoval Ing. Vladimíru Veselému za cenné rady, připomínky a především za trpělivost, s níž přistupoval k mé práci.

# Obsah

# 1 Úvod

V dnešní době máme k dispozici velké množství různých technologií pro přenos textových zpráv v reálném čase (IM – Instant Messaging). Většina z nich je založena na architektuře klient-server, která přináší výhody centralizované správy a problému adresace koncových uživatelů (NAT- technika pro mapování vnitřních adres lokální sítě na veřejnou adresu. Jelikož se uživatel přihlásí k serveru jako první, jsme schopni zjistit jeho adresu. S příchodem sociálních sítí byl protokol XMPP použit jako základ pro komunikaci mezi uživateli. Posílání zpráv mezi uživateli v reálném čase je dnes v hojné míře využíváno nejenom u komunikace prostřednictvím počítače, ale i při komunikaci pomocí jiných inteligentních klientských zařízení – chytrých mobilních telefonů, PDA a tabletů.

V mé bakalářské práci zkoumám způsob, jakým je komunikace jednotlivých klientů zajištěna, jakými prostředky a s jakým zabezpečením probíhá přenos zpráv. Cílem je na základě zjištěných informací sestavit funkční systém, který by byl schopen zachytit a zrekonstruovat jednotlivé zprávy posílané mezi uživateli. V této práci jsou vybrány k prozkoumání dva z používaných protokolů a sice XMPP a MSN.

Kapitola 2 se zabývá protokolem XMPP. Kapitola má za úkol čtenáře seznámit se se základními principy protokolu. Jelikož je tento protokol velmi rozsáhlý, jsou vybrány pouze prvky týkající se IM komunikace. U jednotlivých prvků vysvětluji jejich formát a funkce v komunikaci. Protože se jedná o velmi používaný protokol, uvádím možnosti zabezpečení a za jakých okolností jsme schopni komunikaci analyzovat.

Kapitola 3 se zabývá protokolem MSN. Na rozdíl od XMPP se jedná o jednoúčelový protokol k IM komunikaci. Na úvod kapitoly je představen formát zpráv a jednoduchý příklad komunikace. Poté jsou popsány konstrukce, které jsou ke komunikaci použity.

Kapitola 4 popisuje implementaci systému pro rekonstrukci komunikace MSN a XMPP. Úvod seznamuje čtenáře s jazyky, které jsou pro tvorbu systému použity. Dále se zabývá výběrem konstrukcí IM komunikace, které by měly být systémy pro rekonstrukci MSN a XMPP odchyceny.

Kapitola 5 se zabývá testováním výsledných systémů (XMPP, MSN) a zda dochází ke korektní rekonstrukci vybraných konstrukcí protokolů XMPP a MSN na základě odchycené komunikace.

Kapitola 6 popisuje vše, čeho bylo v této práci docíleno a zabývá se tím, jak by bylo možné tuto práci dále rozvinout.

# 2 XMPP

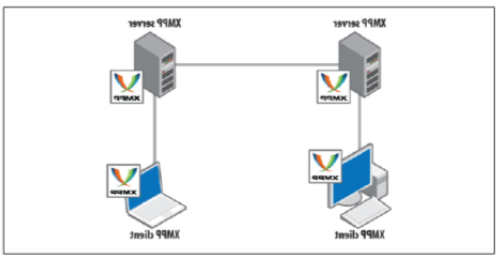
Úkolem této kapitoly je seznámit čtenáře se základními principy, architekturou protokolu, vysvětlit a prezentovat syntaktické a sémantické konstrukce, které jsou pro rekonstrukci komunikace klíčové. Závěrem si uvedeme pár dalších uplatnění tohoto protokolu. Veškeré informace jsou čerpány z [1].

## 2.1 Základy protokolu XMPP

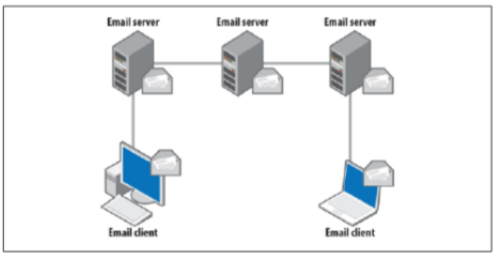
Jedná se o technologii umožňující real-time komunikaci, která vznikla v roce 1999 (verze 0.9) se zaměřením na IM a Presence. V roce 2003 se dočkala standardizace a byla rozdělena do dvou částí. XMPP-Core [2] a XMPP-XEP [3], kde XMPP-Core se věnuje zasíláním zpráv uživatelům a možnosti pozorovat přítomnost uživatelů (Presence). Druhá část XMPP-XEP se věnuje rozšiřitelnosti o další funkčnost, jako jsou audio, video hovory (Jingle), zasílání dat, komunikace s ostatními sítěmi jako je ICQ, Facebook. Jelikož se v této práci zabývám rekonstrukcí IM komunikace, zaměřím se zde zejména na popis části XMPP-Core.

## 2.2 Architektura

Celá komunikace je založena na architektuře klient-server, podobně jako je tomu například u webových stránek a emailu. Protokol se nejvíce přibližuje emailové architektuře, s tím rozdílem, že email není přeposlán z domény klienta přímo serveru cílové domény dotyčné osoby, ale je přeposílán přes jednotlivé emailové servery, které hledají cestu k cílové doméně (obr. 2.11). XMPP posílá zprávu přímo. Klient odešle zprávu svému serveru a ten potom naváže spojení se serverem uživatele (server je schopen si zjistil adresu cílového serveru z databáze a využít tuto adresu k přímému zaslaní zprávy), kterému je zpráva určena a odešle ji (obr. 2.12).



**Obr. 2-1 Email klient-server [1]**




**Obr. 2-2 XMPP klient-server [1]**

## 2.3 Způsob komunikace

Veškerá komunikace, klient-server, server-server probíhá pomocí zpráv ve formátu XML [2]. Aby bylo možné zprávy doručit patřičným adresátům, obsahují atributy *from*, *to*, ve kterých jsou uloženy tzv. **JID (Jabber ID)**, pomocí kterého jsme schopni zprávy v síti správně směrovat. Tento identifikátor se ještě rozděluje na tzv. **Full JID (úplný Jabber ID)** alice@wonderland.lit/hole, kde alice je uživatelským účtem, za zavináčem následuje název serveru a část za lomítkem je označována jako zdroj (XMPP umožnuje běh více klientů pod stejným účtem a tudíž je potřeba jednotlivé instance nějakým způsobem rozlišit pomocí identifikátoru zdroje, resource). **Bare JID (holý Jabber ID)** neobsahuje část o zdroji a je používán servery. Pokud chce klient navázat spojení se serverem, musí nejprve určit typ zabezpečení, které budou během komunikace používat. Jsou nabízeny dva mechanismy zabezpečení. Plain-text, SSL/TLS. Aby byla rekonstrukce možná, je potřeba komunikaci zaznamenat buď bez zabezpečení, nebo v podobě s Plain-text heslem. V případě SSL, TLS není možné šifrovanou komunikaci rekonstruovat. Dnes je zabezpečení velkým tématem a XMPP klienti nejsou pozadu a umožnují všechny výše uvedené druhy zabezpečení (někteří klienti jako například *Jabbim* dokonce nešifrované spojení vůbec neumožňují).

Po výběru zabezpečovacího mechanismu posílá klient první zprávu s obsahem `<stream:stream>` indikující zahájení komunikace XMPP. Po zahájení komunikace je toto spojení ponecháno jako trvalé (keep alive), kde při neaktivitě klienta zasílá server ověřovací zprávy, zda je klient pořád ještě připojen a v případě, že se mu nedostane odpovědi, dojde k ukončení spojení. Celá komunikace tedy probíhá v kořenovém elementu `<stream>`. Uvnitř jsou definovány pouze tři druhy zpráv, které se mohou v `<stream>` elementu objevit. Jedná se tzv. *xml-stanzas* (xml slohy). *Xml-stanzas* rozlišujeme:

- *IQ* slouží pro získávání, zasílání informací, požadavků mezi serverem a klientem. Pomocí těchto zpráv mohou být realizovány i různé rozšíření.
- *Message* přenáší v případě IM obsah textové zprávy určené druhému klientovi.
- *Presence* je určena pro nastavování, získání přítomnosti uživatele. Jako příklad jednoduché komunikace může být následující výpis.

Abych naznačil komunikaci obou stran, používám znaky C: pro klienta a S: pro server.

```
C: <stream:stream>

C:   <presence/>

C:   <iq type="get">
       <query xmlns="jabber:iq:roster"/>
     </iq>

S:   <iq type="result">
       <query xmlns="jabber:iq:roster">
         <item jid="alice@wonderland.lit"/>
         <item jid="madhatter@wonderland.lit"/>
         <item jid="whiterabbit@wonderland.lit"/>
       </query>
     </iq>

C:   <message from="queen@wonderland.lit"
              to="madhatter@wonderland.lit">
       <body>Off with his head!</body>
     </message>

S:   <message from="king@wonderland.lit"
              to="party@conference.wonderland.lit">
       <body>You are all pardoned.</body>
     </message>

C:   <presence type="unavailable"/>

C: </stream:stream>
```

**Obr. 2-3 Příklad XMPP komunikace [1]**

Z ukázky je vidět průběh komunikace. Nejdříve je mezi klientem a serverem navázáno dlouhotrvající spojení (*long lived*), které je označováno `<stream>` elementem. Abychom mohli dát serveru najevo, že jsme v síti XMPP přítomni, zašleme *Presence* (*stavovou*) zprávu, která informuje o výskytu uživatele v síti. Po úspěšném připojení do sítě je potřeba získat od serveru seznam kontaktů (*roster*). Zde bych chtěl zmínit, že výskyt kontaktů, historie komunikace uložené na straně klienta není nijak implementován v samotném XMPP, jedná se o schopnost klienta. Abychom získali daný seznam, je potřeba zaslat *IQ* zprávu s žádostí o seznam kontaktů. Vzápětí obdržíme od serveru zprávu *IQ* se všemi kontakty patřící danému účtu (Odpověď lze poznat pomocí hodnoty *result* atributu *type*). Další akcí, kterou lze z příkladu vypozorovat je zaslání zprávy (element `<message>`) uživateli s JID alice, která má účet zřízen v doméně wonderland.lit a jedná se o všeobecný zdroj, jelikož je obsah za lomítkem prázdný. Z bezpečnostních důvodů nebývá adresa *from* vyplněná při odesílání klientem. Adresu *from* až následně doplní server po odeslání cílovému uživateli. Na konec komunikace vidíme odhlášení uživatele ze sítě pomocí `<presence>` elementu a následným ukončením dlouhotrvajícího spojení elementem `<stream>`. Tato ilustrace je velmi stručná avšak běžná komunikace se příliš nelíší od komunikace v našem příkladu. Podrobnější vysvětlení o nejdůležitější části pro rekonstrukci IM komunikace následuje v další podkapitole.

## 2.4 Syntaxe a sémantika konstrukcí

Jak již bylo dříve zmíněno, je komunikace založena na XML zprávách, a tudíž syntaxe jednotlivých konstrukcí je dána XML předpisem. Jaký obsah mohou jednotlivé konstrukce nabývat je definováno v souborech jmenného prostoru XMPP (*xmlns*). Definice se skládá ze dvou částí, pro které se tyto definice vyskytují. První část je XMPP-Core, kde se nachází *xmlns* soubory pro IM a Presence komunikaci. Druhá část se týká XMPP-XEP, a jedná se o velké množství souborů definujících různá rozšíření, mezi které patří například služba Jingle, která umožňuje audio hovory mezi uživateli pomocí XMPP. Dále se budu zabývat vysvětlením jednotlivých *xml-stanz* vyskytujících se v části XMPP-Core, které jsou z hlediska analýzy a rekonstrukce IM komunikace klíčové [1].

### 2.4.1 Message

Jedná se o *xml-stanzu* obsahující zprávu uživatele. Tato sloha je odesílána pomocí režimu *fire-and-forget* kdy server po odeslání zprávy patřičnému klientovy neočekává potvrzení o přijetí. Jedná se o násilné odesílání, kdy se nebere v úvahu možná ztráta zprávy. Hlavním důvodem pro zavedení tohoto režimu je snaha předejít zahlcení sítě. Zpráva je označena elementem `<message/>` a její atributy jsou:

- *from* - obsahuje adresu odesílajícího klienta, která není odesílatelem vyplněna, ale z bezpečnostních důvodů zde server později dosadí svou adresu (sniff – možnost odposlechu při putování zprávy od klienta k serveru a zneužití údajů poskytnutých klientem)
- *to* - je adresa klienta, kterému je zpráva určena (nejedná se o přímou komunikaci klientů, ale komunikace probíhá přes server. Z toho vyplývá, že cílová IP adresa je adresa serveru uživatele, od kterého je zpráva zasílána)
- *type* - z názvu plyne, že se jedná o typ message zprávy. Důležité hodnoty, kterých může nabývat, jsou:
    - *normal* – Komunikace mezi dvěma klienty, kteří nemusí mít jeden druhého ve svém kontaktním seznamu, tudíž se nemusí jednat o autorizované uživatele.
    - chat – Mohou navázat mezi sebou komunikaci jen lidé, kteří jsou navzájem přidáni do seznamu kontaktů a jednotlivými uživateli authorizováni. Jedná se již o plnohodnotné chatové sezení.
    - *groupchat* – Jedná se o podobný princip jako u „chat“ s rozdílem, že tyto vlastnosti jsou vyžadovány od všech účastníků diskuze. Všechny zprávy jsou směrovány do tohoto sezení (při vytvoření groupchat komunikace je vytvořeno nové JID zastupující dané sezení, a zprávy zasílané do tohoto sezení jsou potom rozesílány všem zúčastněným).
- *id* – Jedná se o volitelnou hodnotu sloužící k identifikaci uživatele.

Obsahem *message* stanzy je element `<body>`, který obsahuje text předávaný mezi uživateli a možnost výskytu elementu, který oznamuje o aktivitě, popřípadě neaktivitě druhé strany (o tom jestli jsou zasílány s *message* zprávou i `<chatstate>` elementy rozhoduje uživatel. Ať už při přidávání uživatele do seznamu kontaktů, nebo nastavením klientského programu). Příklad jednoduché *message* zprávy:

```
<message from="you@yourdomain.tld/work"
         to="daughter@yourdomain.tld"
         type="chat">
  <body>Hi honey!</body>
  <active xmlns="http://jabber.org/protocol/chatstates"/>
</message>
```

**Obr. 2-4 Zpráva uživateli s údajem chatstate [1]**

### 2.4.2 Presence

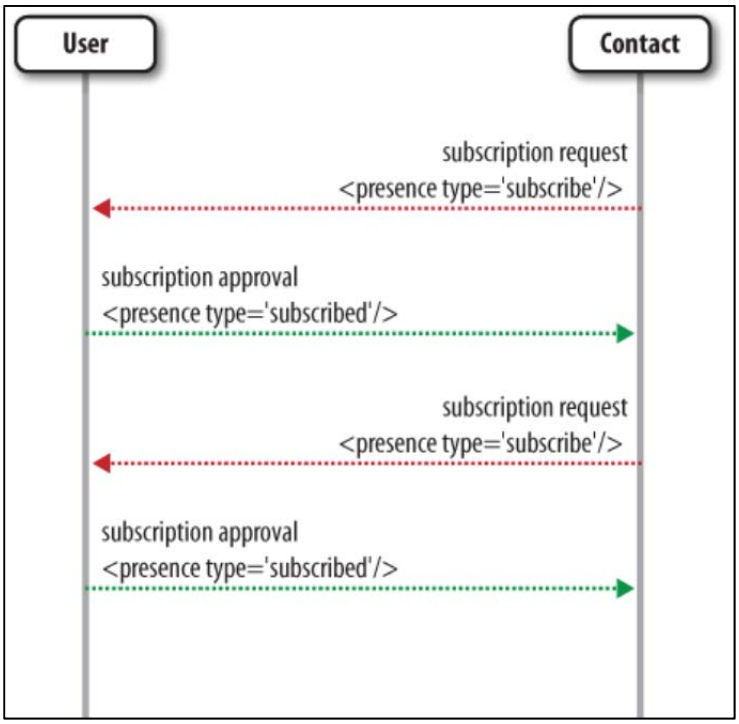
Slouží k informování o přítomnosti uživatelů. K tomu, aby uživatelé mohli přijímat zprávy o přítomnosti ostatních uživatelů ze seznamu kontaktů, je potřeba jim tuto vlastnost povolit (jedná se o tzv. „subscribe“ mechanismus a obvykle k němu dochází již při přidání kontaktu do seznamu. Popisu tohoto mechanismu se věnuji níže v podkapitole). Po dokončení „subscribe“ mechanismu vysvětlím, jakým způsobem dochází k zasílání zpráv informujících o přítomnosti uživatele. Aby nemusel tento úkol řešit klient a celý úkon probíhal centralizovaně, zajišťuje tento proces server, ke kterému je klient připojen. Po určitých intervalech server zasílá klientu testovací zprávy (hodnota *probe* atributu *type*), které slouží k získání odpovědi od uživatele. Po obdržení odpovědi, rozešle tuto zprávu o stavu všem uživatelům z kontaktního seznamu, kteří mají povolené zasílání zpráv o přítomnosti. V případě, že se odpovědi do určité doby nedostane, vyhodnocuje tento stav server jako „offline“, vygeneruje „offline“ zprávu a rozešle tuto zprávu jako v předchozím případě. Zpráva o stavu je zapouzdřená do elementu `<presence>`. Tento element obsahuje atributy:

- *from* – obsahuje adresu (JID) od koho je daná zpráva zasílána.
- *to* – obsahuje adresu (JID) komu je daná zpráva určena (jedná se o bare JID, protože je zpráva zasílána na server, ke kterému je uživatel přihlášen).
- *type* – tento atribut slouží k povolení stavových zpráv uživatelům (subscribe, subscribed, unsubscribed), zasílání sondovacích zpráv (probe). V ostatních případech je tento atribut vynechán.

#### 2.4.2.1 Výměna stavových zpráv mezi uživateli je následující

a. Nový kontakt zašle *presence* zprávu typu *subscribe* a čeká na odezvu od uživatele (v případě povolení obdrží zprávu typu *subscribed*. V opačném případě *unsubscribed*)
b. Tento proces se opakuje i ze strany uživatele (Děje se tak, aby obě strany měli možnost sledovat stavy navzájem mezi sebou (Tomuto mechanismu se říká *bidirectional subscription*). Po dokončení sdílí kontakt s uživatelem navzájem svou přítomnost v XMPP síti.

Příklad celého procesu:




**Obr. 2-5 Vyjednání povolení o zasílání přítomnosti uživatele [1]**

#### 2.4.2.2 Příklad výměny testovacích zpráv (probe)

- Server v intervalech zasílá testovací zprávu uživateli. Tato zpráva slouží jako dotaz k získání informací o přítomnosti uživatele. Uživatel jako odpověď zašle zprávu vypovídající o jeho přítomnosti v síti XMPP. Pokud uživatel do určité doby neodpoví, server předpokládá „*offline*“ stav uživatele a na základě toho vygeneruje vlastní zprávu, která kromě „offline“ informace, obsahuje element `<delay>` s časovým údajem (*stamp*), který informuje o času odpojení.
- Serverem vygenerovaná zpráva je zaslána všem kontaktům ze seznamu, které mají povolené přijímání stavových zpráv. Zprávy jsou zasílány jen v případě změny stavu, aby nedocházelo ke zbytečnému zahlcování sítě.

Příklad testovacích zpráv:

```
<presence from="alice@wonderland.lit/rabbithole"
          to="sister@realworld.lit"
          type="probe"/>

<presence from="alice@wonderland.lit/rabbithole"
          to="madhatter@wonderland.lit"
          type="probe"/>
```

**Obr. 2-6 Probe zpráva zaslána uživatelům [1]**

```
<presence from="sister@realworld.lit/home"
          to="alice@wonderland.lit/rabbithole"
          type="unavailable">
  <delay xmlns="urn:xmpp:delay"
         stamp="2008-11-26T15:59:09Z"/>❶
</presence>

<presence from="madhatter@wonderland.lit/foo"
          to="alice@wonderland.lit/rabbithole"/>

<presence from="madhatter@wonderland.lit/bar"
          to="alice@wonderland.lit/rabbithole"/>
```

**Obr. 2-7 Odpověď uživatelů na probe zprávu [1]**

Obsahem presence zpráv jsou následující elementy:

- *status* – Informuje o obecném stavu uživatele (slouží k rychlé představě, zda je uživatel přítomen, nebo vykonává jinou činnost). Obsahem status elementu jsou následující hodnoty:
  - *chat* – Uživatel je připojen a dostupný ke komunikaci přes XMPP.
  - *away* – Název již napovídá, že se jedná o krátkodobou nepřítomnost uživatele (obvykle se jedná o případ, že uživatel není aktivní po určitou dobu, nebo si sám tento status sám zvolil).
  - *xa* – Obdoba „*away*“ stavu, s tím rozdílem, že se jedná o delší časový úsek. Obvykle tomu tak je v případě, že uživatel stráví určitou dobu v „*away*“ stavu a po uplynutí této doby se změní stav na „*xa*“ (extended away).
  - *dnd* – Jedná se o zkratku „do not disturb“, která vypovídá o přítomnosti uživatele, který nechce být rušen. Tento stav má obvykle za následek přepnutí klientského programu do tichého režimu, kdy příchod nových zpráv neruší uživatele od činnosti, kterou vykonává.

- *show* – Slouží pro bližší popis k danému stavu (obsah tohoto elementu je buď implicitně zadán, nebo si text může doplnit uživatel sám).
- *priority* – V případě, že máme spuštěných více klientů s jedním účtem (toto je umožněno pomocí *resource* části vyskytující se za *bare JID* částí klienta viz. 2.3 způsob komunikace), můžeme určit, kterému účtu budou směrovány zprávy v případě, že „*to*“ adresa bude určena jenom pomocí *bare JID*. Hodnoty priority jsou od -128 do 127. V případě, že hodnota jde do záporu, znamená to, že si nepřejeme, aby byly zprávy doručovány na tento účet (platí jenom v případě zasílání zpráv na *bare JID*. V případě *Full JID*, je zpráva vždy doručena)

Příklad zasílání informací o stavu uživatele:

- Standartní stav [1]

```
<presence>
  <show>away</show>
  <status>Having a spot of tea</status>
</presence>
```

- Rozšířený stav o prioritu [1]

```
<presence from="me@myserver.tld/office">
  <priority>7</priority>
</presence>

<presence from="me@myserver.tld/TV">
  <priority>-1</priority>
</presence>
```

### 2.4.3 IQ

*IQ (Info/Query)* zpráva se stará o veškerou ostatní komunikaci. Mezi základní typy, které jsou v rámci rekonstrukce protokolu implementovány:

- *Roster* – Správa kontaktů.
- *vCard* – Správa osobních údajů uživatele.
- *Blocking* – Správa blokování kontaktů.

IQ zpráva je prezentována elementem `<iq>` a je koncipována tak, aby umožňovala rozšíření. To, jestli jsou některé rozšiřující zprávy popsány v XMPP-XEP podporovány ze strany serverů a jaké (pomocí *discovery #items*, *info*), není předmětem této práce, a tudíž je zde zmíněno jen jako zajímavost.

Obecně se jedná o informační (dotazovací) zprávu, která jako výsledek vrací vždy nějakou odpověď (request – response). Její atributy jsou velice podobné atributům u předchozích typů zpráv.

- *from – JID* uživatele zasílajícího zprávu.
- *to – JID* přijímací strany (zprávy mohou být vyměňovány pouze mezi klientem a serverem).
- *id* – Jedná se identifikační číslo, které si náhodně zvolí strana, generující informační (dotazovací) zprávu. Strana odpovídající na tuto zprávu musí použít stejné id číslo, aby přijímací strana věděla ke kterému dotazu odpověď přiřadit (hlavním důvodem je, zopakování požadavku po uplynutí určité doby od odeslání). U zpráv typu *message* a *presence* může být ve většině případů *id* vynecháno.
- *type* – určuje typ zasílané zprávy. Hodnoty jsou následující:
    - *get* – Jedná se o žádost o informace ze serveru.
    - *result* – Odpověď na žádost, nebo i na set (v případě set se jedná pouze o potvrzení přijmutí hodnot).
    - *set* – Klient zasílá nějaké informace na server (přidání uživatele do kontaktů).

Následují jednotlivé typy dotazů, vysvětlíme jejich význam a předvedeme názornou ukázku použití.

#### 2.4.3.1 Roster

Obsahem této zprávy jsou kontakty uživatele. V IQ zprávě se roster vyskytuje v elementu `<query>` a s kombinací atributu *type* elementu `<iq>`, jsme schopni určit význam obsahu.

- Přidání, odstranění, úprava kontaktu ze seznamu kontaktů. Na následujícím obrázku je vidět přidání nového kontaktu elementu `<item>` s atributem *jid*, který určuje jeho identifikaci. Obsahem kontaktu je podelement `<group>`, který oznamuje, do které skupiny bude nový kontakt zařazen

```
<iq id="u4tsf153" type="set">
  <query xmlns="jabber:iq:roster">
    <item jid="mapbot@wonderland.lit">
      <group>Bots</group>
    </item>
  </query>
</iq>
```

**Obr. 2-8 Úprava kontaktu ze strany klienta. Změna je odesílána na server [1]**

- Zaslání požadavku o seznam kontaktů (dáno hodnotou *get* atributu *type*). Jelikož se jedná o žádost na server, zůstává `<query>` element prázdný a oznamuje nám, že chceme získat seznam kontaktů.

```
<iq from="alice@wonderland.lit/rabbithole"
    id="jh2gs675"
    to="alice@wonderland.lit"
    type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

**Obr. 2-9 Server zasílá na server požadavek o seznam kontaktů [1]**

- Získání seznamu ze serveru. Odpovědí serveru na dotaz je naplněný `<query>` element, obsahující všechny kontakty. Kontakty jsou prezentovány elementy `<item>`. Atribut *name* slouží jako jméno pro zobrazení v klientovi. *Subscription* určuje, zda daný kontakt může získávat zprávy o přítomnosti uživatele. Jako podelement se zde může vyskytnout `<group>` element určující skupinu, do které je kontakt zařazen.

```
<iq from="alice@wonderland.lit"
    id="jh2gs675"
    to="alice@wonderland.lit/rabbithole"
    type="result">
  <query xmlns="jabber:iq:roster">
    <item jid="whiterabbit@wonderland.lit"
          name="The White Rabbit"
          subscription="none"/>
    <item jid="lory@wonderland.lit"
          name="The Lory"
          subscription="to"/>
    <item jid="mouse@wonderland.lit"
          name="The Mouse"
          subscription="both"/>
    <item jid="queen@wonderland.lit"
          name="Her Royal Highness"
          subscription="from"/>
    <item jid="sister@realworld.lit"
          name="Sis"
          subscription="both"/>
  </query>
</iq>
```

**Obr. 2-10 Serverem vrácený uživateli seznam kontaktů [1]**

#### 2.4.3.2 vCard

Osobní údaje o uživateli jsou zasílány uvnitř elementu `<iq>` obecně známého formátu (*vCard*), který je běžně používán pro výměnu osobních údajů (například v e-mailu). Tento formát je v XMPP definován jako rozšíření [3]. Nasazení formátu *vCard* mělo být pouze dočasné, nicméně ke změně nedošlo a formát *vCard* je používán dodnes (o tom, že bylo zvažováno použití jiného formátu než vCard lze vypozorovat ze samostatného jmenného prostoru, do kterého *vCard* v rámci komunikace XMPP patří). Tento jmenný prostor je definován dodnes a nebyl nahrazen žádným jiným pro zasílání *vCard* informací. Ukázka zapouzdření tohoto formátu do *iq* zprávy:

```
<iq from="alice@wonderland.lit"
    id="pw91nf84"
    to="mouse@wonderland.lit/pool"
    type="result">
  <vCard xmlns="vcard-temp">
    <N>
      <GIVEN>Alice</GIVEN>
    </N>
    <URL>http://wonderland.lit/~alice/</URL>
    <PHOTO>
      <EXTVAL>http://www.cs.cmu.edu/~rgs/alice03a.gif</EXTVAL>
    </PHOTO>
  </vCard>
</iq>
```

**Obr. 2-11 Informace o uživateli alice@wonderland.lit ve formátu vCard [1]**

#### 2.4.3.3 Blocking

Slouží pro případ blokování veškeré komunikace od zadaných uživatelů [3]. Celý princip funguje na základě zaslání zprávy s JID dotyčné osoby (v každé zprávě je obsažen pouze jeden kontakt), která má být blokována, a server za nás vyřídí zbytek. Od této chvíle se osoba, která blokování nastavila, jeví pro vybrané kontakty v režimu offline a veškeré zprávy, které budou blokovanými uživateli zasílány, budou serverem zahazovány. Informace blokování je zapouzdřena v elementech `<blocklist>` pro získání seznamu blokovaných kontaktů, `<block>` pro nastavení blokování, `<unblock>` pro odblokování kontaktu. Příklady zapouzdření v *iq* zprávě:

- Zablokování uživatele

```
<iq from="you@yourdomain.tld/newjob"
    id="i3s91xc3"
    to="you@yourdomain.tld"
    type="set">
  <block xmlns="urn:xmpp:blocking">
    <item jid="spammers.lit"/>
  </block>
</iq>
```

- Přijmutí seznamu blokovaných kontaktů

```
<iq from="you@yourdomain.tld/newjob"
    id="92h1nv8f"
    to="you@yourdomain.tld"
    type="get">
  <blocklist xmlns="urn:xmpp:blocking"/>
</iq>

<iq from="you@yourdomain.tld"
    id="92h1nv8f"
    to="you@yourdomain.tld/newjob"
    type="result">
  <blocklist xmlns="urn:xmpp:blocking">
    <item jid="boss@bigcompany.com"/>
    <item jid="spammers.lit"/>
  </blocklist>
</iq>
```

- Odblokování uživatele

```
<iq from="you@yourdomain.tld/newjob"
    id="ng23h57w"
    to="you@yourdomain.tld"
    type="set">
  <unblock xmlns="urn:xmpp:blocking">
    <item jid="boss@bigcompany.com"/>
  </unblock>
</iq>
```

# 3 MSN

Kapitola má za úkol čtenáře seznámit se základními principy, architekturou protokolu a také vysvětlit a prezentovat syntaktické a sémantické konstrukce, které jsou pro rekonstrukci komunikace klíčovými faktory.

## 3.1 Základy protokolu

V případě MSN se jedná o rozsáhlou skupinu služeb společnosti Microsoft. Z hlediska komunikace v reálném čase se jedná o službu MSN Messenger. Tato služba vznikla v roce 1995 za účelem umožnit uživatelům komunikaci v reálném čase pomocí výměny zpráv. Služba byla založena na protokolu MSNP verze 1 a s postupem času se protokol dostal až na nynější verzi 20. Na rozdíl od XMPP protokolu MSNP neumožňuje žádné nástavby a tudíž zde odpadají konstrukce, které se u XMPP vyskytují. Jednotliví uživatelé disponují účty, které slouží jako identifikace na síti (obvykle ve formě e-mailu, je možné zaregistrovat si již existující e-mailový účet nebo vytvořit novou registraci, například hotmail).

## 3.2 Způsob komunikace

Informace, ze kterých čerpám v této kapitole, se nachází v [4]. Jedná o klient-server architekturu. Jelikož uživatelské účty neobsahují žádnou část naznačující, ke kterému serveru by se daný uživatel měl připojit (jak tomu bylo například u XMPP protokolu), připojují se všichni uživatelé k jednomu serveru. Tomuto serveru se říká *Notification Board* (NB) a jeho úkolem je oznamovat ostatním uživatelům našeho seznamu kontaktů, že jsme online. Na rozdíl od XMPP sítě, kde mohlo přihlášení do sítě proběhnout zašifrovaně a následující komunikace byla tím pádem pro nás nečitelná, protokol MSNP žádnou techniku zabezpečení pomocí zašifrování nevyužívá. Všechny zprávy jsou zasílaný nezabezpečeně v textovém formátu. Využívaný protokol se syntaxí podobá protokolu HTTP (příkaz se nachází na prvním řádku, jednotlivé parametry jsou od sebe odděleny mezerami pro oddělení jednotlivých částí hlavičky slouží znaky <*CR*><*LF*>). Ke komunikaci jsou využívány tříznakové příkazy (*PNG*, *RNG*, *USR*, *CAL*, *QNG*, *XFR*, *ANS*, atd.), které určují sémantiku zpráv. Díky této technice a poměrně málo parametrům, které se vyskytují u jednotlivých příkazů, je MSNP nenáročný na data. Je zde možný i výskyt parametrů, které řadím do dvou kategorií:

- *Parametry příkazu* – Všechny parametry se vyskytují na stejném řádku, jako samotný příkaz. Pro oddělení příkazu od parametrů a parametrů samotných zde slouží mezera. Celý příkaz s parametry je zakončen ASCII znaky <*CR*><*LF*>.

- *Parametry obsahu* – Jedná se o parametry pro správnou interpretaci formátu obsahu dané zprávy. Může se například jednat o kódování zprávy, použitý font, verzi MIME.

Pokud se jedná o informační zprávu, je obsah od příkazu s parametry oddělen pomocí zdvojení znaků <*CR*><*LF*> (takovéto oddělení hlavičky od obsahu je použito v protokolu HTTP). Po úspěšném přihlášení k NB je v případě odeslání informační zprávy nejprve zaslán požadavek na server o připojení k tzv. *Switch Board* serveru (SB). Jedná se o server, který řídí všechny požadavky týkající se IM komunikace. Výsledkem je nové připojení, které slouží k IM komunikaci mezi vybranými uživateli (chat, groupchat). Po dobu konverzace je toto spojení udržováno pomocí testovacích zpráv a spolu s ukončením konverzace je ukončeno i SB spojení. Pro představu uvádím jednoduchý příklad MSN Messenger komunikace.

- Dohodnutí se na verzi protokolu, kterou bude klient i server během komunikace používat zařizuje *Dispatch Server* (DS).

| | |
|---|---|
| S: | VER 1 MSNP11 CVR0\r\n |
| C: | VER 1 MSNP11 CVR0\r\n |

- Po úspěšném dohodnutí verze pomocí DS je nutná autentizace uživatele. Tuto autentizaci provádí autentizační server nazývaný *Notification Server* (NS). Po dokončení autentizace dojde k zaslání nové adresy NS, který byl danému uživateli přidělen (zde je vidět, že serverů je víc a dochází k tzv. *load balacingu* – rozložení zátěže)

| | |
|---|---|
| S: | USR 3 TWN I example@passport.com\r\n |
| C: | XFR 2 NS 207.46.106.145.1863 0 207.46.104.20:1863\r\n |

  Po dobu připojení zasílá NS sondovací zprávy, které se po určitých intervalech opakují. Pokud jsou v daných intervalech obdrženy odpovědi, NS prodlužuje interval zasílání těchto zpráv, aby nedocházelo k zahlcení sítě.

| | |
|---|---|
| S: | PNG |
| C: | QNG 42 |
| S: | PNG |
| C: | QNG 47 |

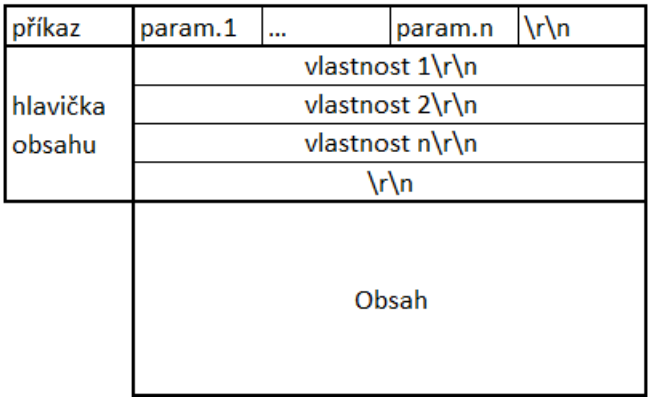
- Pokud se uživatel pokusí o komunikaci s jiným uživatelem, je nejdříve zjištěna dostupnost uživatele, následně po odezvě uživatele dochází k vytvoření samostatného spoje, který slouží jako jakýsi mezičlánek mezi uživateli (tímto způsobem je řešen problém technologie NAT).

```
S: RNG 1697467578 64.4.61.112:1863 CKI 119649.227224237 radnecx@gmail.com Radek U messenger.msn.com 1
C: ANS 1 zduba.andreas@gmail.com;{12FC37FA-52CC-6C02-7F02-275A70A17A63} 119649.227224237 1697467578
S: ANS 1 OK
S: JOI zduba.andreas@gmail.com zduba.andreas@gmail.com 2416181284:2147619840
S: MSG radnecx@gmail.com Radek 109
```

## 3.3 Syntaxe a sémantika konstrukcí

Proti protokolu XMPP je syntaxe velmi jednoduchá. Zpráva je rozdělena na dvě hlavní části:

- *Hlavička* - dělí se na dvě části:
  - *Příkaz* – začíná třemi znaky, které určují význam příkazu. Následují mezerou oddělené parametry, které jsou zakončeny dvojicí znaků *\r\n*
  - *Vlastnosti obsahu* – určují dodatečné informace k obsahu zprávy (kódování, font atd.). Vlastnost začíná názvem, který je od hodnoty oddělen dvojtečkou a mezerou. Za názvem následuje hodnota (hodnot může být i více. V tom případě je za hodnotou použit středník s mezerou, k oddělení jednotlivých hodnot) vlastnosti ukončená znaky *<CR><LF>*. Za poslední vlastností se dvakrát opakují znaky *<CR><LF>*, pro oddělení hlavičky od obsahu
- *Obsah* – v našem případě se zde vyskytuje předávaná zpráva mezi uživateli (text)




**Obr. 3-1 Popis protokolu MSN**

Následují jednotlivé příkazy relativně aktuální verze protokolu MSNP18. V dnešní době je nejnovější verzí MSNP21, ale jelikož se nejedná o software s veřejně dostupnými informacemi, snažím se vyjít z nashromážděných informací pomocí různých nástrojů jako je Wireshark, nebo literatury z internetu viz [5]. Vybrány jsou nejdůležitější příkazy pro analýzu a rekonstrukci IM komunikace. Důležité je spíše vědět o aktivitě uživatele na internetu, než vědět o příkazech, které slouží hlavně pro informace serveru.

### 3.3.1 Přihlášení do sítě MSN Messenger

Prvním krokem je přihlášení se pomocí uživatelského účtu. Účet je v podobě e-mailu (např. franta@seznam.cz , franta@hotmail.com). Abychom byli schopni se pomocí účtu přihlásit, musíme se domluvit na verzi protokolu, pomocí které budeme komunikovat. Proto klient zašle následující zprávu na DS server MSN messengeru.

```
S:      VER 1 MSNP11 CVR0
```

Obecný formát:

```
VER trid protocol1 protocol2 … protocolN CVR
```

- *VER* – název příkazu pro dohodnutí verze (Version)
- *trid* – id transakce
- *protocolN*, seznam podporovaných verzí protokolu
- *CVR0* – jedná se o informace, které jsou zasílány na server. Informace souvisí s minimálními požadavky na provozování dané služby a z hlediska analýzy IM nás nemusí dále zajímat

Po úspěšném dohodnutí verze protokolu dochází na samotnou autentizaci uživatele. Požadavky na přihlášení vyřizuje DS, který rozhodne, která adresa NS bude klientovi zaslána pro autentizaci.

```
C:      XFR 2 NS 207.46.106.145:1863 0 307.46.104.20:1863
```

Obecný formát:

```
XFR trid NS address 0 current_address
```

- *trid* – id transakce
- *NS* – napovídá, že se jedná o přesměrování na *Notification Server*
- *address* – IP a port přesměrovávaného serveru
- *0* – vždy nula
- *current_address* – IP a port nynějšího serveru (DS)

```
C:      USR 10 SSO I buddy@live.com
```

Obecný formát:

```
USR trid SSO I email
```

- *trid* – id transakce
- *SSO* – zkratka pro Single Sign On (zavedeno od msnp15, pro jediné přihlášení ke službě)
- *I* – identita. Příznak, že další parametr bude uživatelský účet.
- *email* – uživatelský účet pro přihlášení

Nyní je uživatel autentizovaný, přihlášeny k NS a začíná komunikovat s ostatními uživateli. Jak už bylo zmíněno v předchozí části, klient nejprve provede volání uživatele. V případě odpovědi, je vytvořeno nové spojení SB, které je sdíleno pouze vybranými účastníky.

```
C:      RNG 1697467578 64.4.61.112:1863 CKI 119649.227224237 radnecx@gmail.com Radek U messenger.msn.com 1
```

Obecný formát:

```
RNG sessid address authtype ticket invitepassport invitename
```

- *sessid* – id relace do které jste pozváni. Stejné id je potřeba při odezvě na pozvání pomocí ANS
- *address* – IP, port SB, ke kterému se máte připojit
- *authtype* – typ autentizace, který je pro připojení do SB použit. Vyskytuje se pouze CKI
- *ticket* - ticket, který jako sessid je potřeba vložit do odpovědi ANS
- *invitepassport* – identifikační údaj osoby, která vás vyzývá ke komunikaci
- *invitename* – identifikace osoby v přívětivější podobě. Obvykle je zobrazeno jméno osoby

Následuje odpověď na pozvání.

```
S:      ANS 1 zduba.andreas@gmail.com;{12FC37FA-52CC-6C02-7F02-275A70A17A63} 119649.227224237 1697467578
```

Obecný formát:

```
ANS trid your_passport ticket sessid
```

- *trid* – id transakce
- *your_passport* – obdržený údaj z předchozího kroku
- *ticket* – opět se jedná o údaj obdržený z předchozího kroku
- *sessid* – obdržen v předchozím kroku

Nyní se zúčastnění nachází v SB připojení a mohou si mezi sebou vyměňovat zprávy pomocí příkazu `MSG`.

```
C:      MSG radnecx@gmail.com Radek 109\r\n
        MIME-Version: 1.0\r\n
        Content-Type: text/plain; charset=UTF-8\r\n
        X-MMS-IM-Format: FN=Segoe%20UI; EF=; CO=0\r\n
        \r\n
        Ahoj
```

Jako první parametr je uživatelský účet dotyčné osoby, následuje jméno, které se nám má zobrazovat, délka zprávy. Na dalších řádcích se vyskytují vlastnosti obsahu, o kterých jsem již mluvil. Jedná se o vlastnosti jako *MIME* verze, typ obsahu (*text*), kódování, a styl textu (*font size*, *font-family*, atd.). Za poslední vlastností následuje dvojice *\r\n* a za touto dvojicí se již nachází obsah zprávy. Takto jsou vyměňovány veškeré zprávy mezi uživateli až do doby než jeden z uživatelů zašle příkaz `BYE` a SB sezení ukončení. Uživatel se nakonec odpojí ze sítě pomocí příkazu `OUT`.

# 4 Implementace

Tato kapitola pojednává o technikách, které byly pro implementaci rekonstrukce protokolů XMPP a MSN použity. K implementaci byly použity jazyky Python a NPL. Bližšímu popisu použitých konstrukcí se věnuji níže v samostatné podkapitole věnované implementaci v jazyce Pythonu. Jazyk NPL je speciálně vyvinutý jazyk pro popis PDU (Ethernet, TCP, UDP, HTTP) společností Microsoft. Jazyk NPL lze použít pouze v kombinaci s aplikací Microsoft Network Monitor (MNM). Prostředí MNM slouží pro zobrazení výsledků, naformátovaných dat. Bližší popis v podkapitole o NPL.

Kapitola 4.1 se věnuje celému cyklu tvorby programu pro rekonstrukci komunikace v jazyce Python. Na úvod jsou vysvětleny rysy, kterými se jazyk Python vyznačuje.

Kapitola 4.1.1 popisuje postup tvorby abstraktního modelu, výběru klíčových vlastností pro modelování. Jelikož abstraktní model není z teoretického hlediska programový kód, vyskytuje se zde další podkapitola.

Kapitola 4.1.2 se zabývá transformací abstraktního modelu do programového kódu.

## 4.1 Python

Jedná se o skriptovací jazyk. Mezi přednosti tohoto jazyka patří podpora pro objektově orientované programování, přehlednost kódu díky syntaktickým konstrukcím, rychlost zpracování příkazů (knihovny přeloženy do jazyka C) a mnoho dalších, které ale z hlediska této práce nemá smysl zmiňovat. Naopak, je třeba brát v úvahu, že se jedná o jazyk, který ke svému vykonávání vyžaduje interpret. Na základě tohoto faktu je nutné program navrhnout tak, aby při výskytu chyb nebyl běh programu zastaven, ale daná chyba byla patřičně ošetřena a bylo zajištěno pokračování v provádění programu.

### 4.1.1 Abstraktní model

Jelikož jsou protokoly XMPP a MSN rozsáhlé a řada příkazů, pomocí kterých tyto protokoly komunikují, se netýkají komunikace typu Instant Messaging, je potřeba vybrat ty vlastnosti, které důležité jsou. Následující část se zabývá výběrem důležitých vlastností pro tvorbu abstraktního modelu pro protokol XMPP.

- XMPP

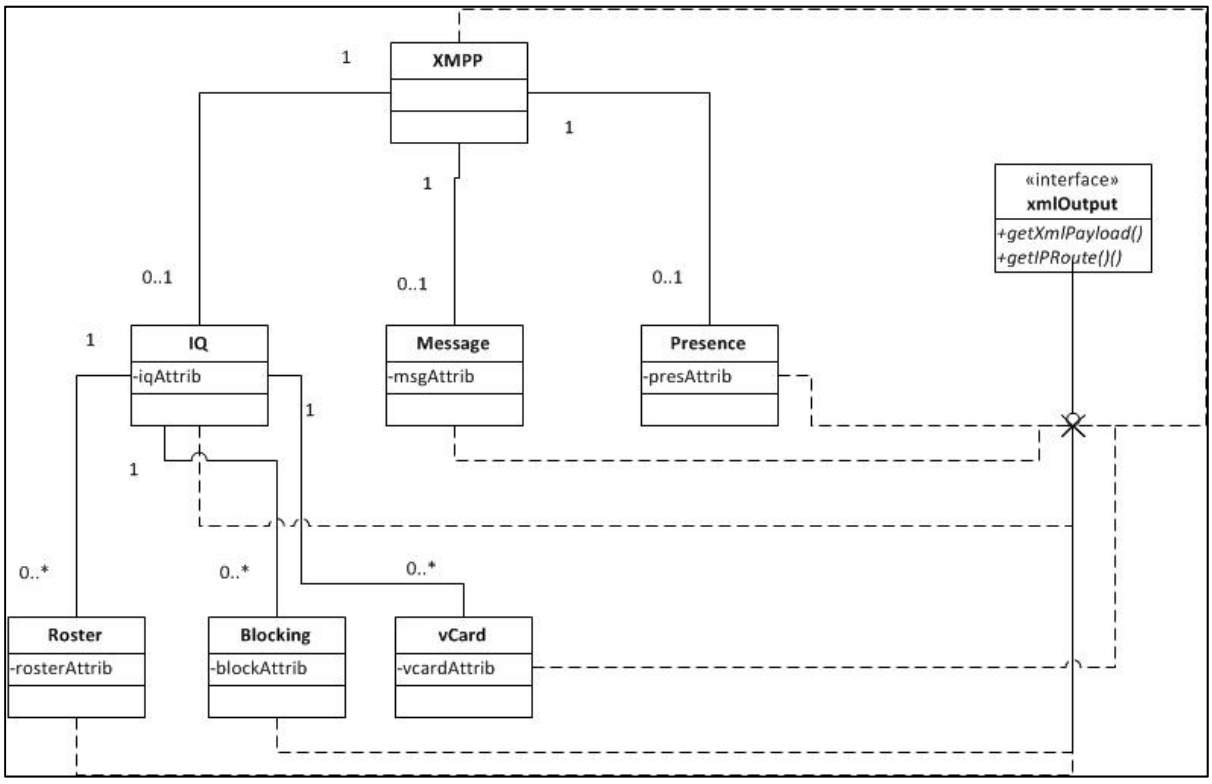
  Vybrány byly následující entity:

  - IQ
    - Seznam kontaktů
    - Blokace
    - vCard (osobní údaje)

- Message
  - Mezi-uživatelská komunikace
- Presence
  - Přítomnost uživatele

V protokolu XMPP se jedná o elementy, pomocí kterých je řízena veškerá ostatní komunikace. Jelikož jsou elementy `<iq>`, `<message>` a `<presence>` příliš rozsáhlé, je vhodné kvůli údržbě, znovu-použitelnosti modelovat části těchto elementů jako samostatné entity. Jelikož se jedná o analýzu a rekonstrukci komunikace, je potřeba, aby každá z těchto entit uměla tuto rekonstrukci uskutečnit. Tohoto chování je možné docílit za pomocí jednotného rozhraní (`getXmlPayload()`, `getIPRoute()`), které musí dodržet výše definované entity.

- `getXmlPayload` – Metoda, která rekonstruuje vstupní data do výstupní podoby ve formátu XML
- `getIPRoute` – Jelikož formát XML určuje pouze syntaxi, je potřeba metoda, která je schopna na základě zanalyzovaného obsahu přiřadit IP adresu k patřičné osobě (jelikož se komunikuje přes server, je nutné toto odlišit a chybně nepřiřazovat adresy)




**Obr. 4-1 Abstraktní model protokolu XMPP**

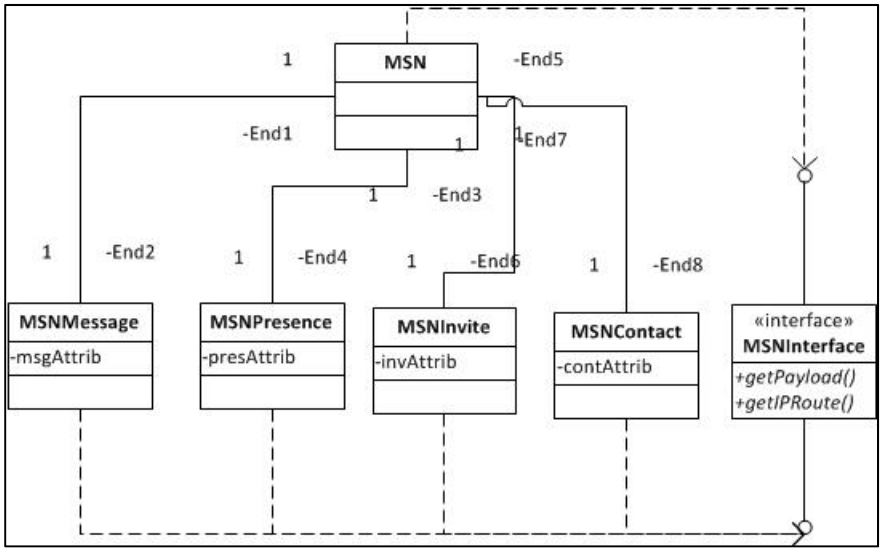
Další část se zabývá výběrem vlastností pro protokol MSN

- MSN

  Oproti protokolu XMPP se jedná o textový formát, podobný protokolu HTTP. Opět je potřeba se zaměřit pouze na vlastnosti, entity související s komunikací typu IM.

  - *Message*
  - *Presence*
  - *Invite* (Výzva uživatele k chatu)
  - *Contact*

  Na rozdíl od XMPP je forma zpráv velmi jednoduchá a krátká, tudíž není třeba výše vypsané entity dále rozkládat. Opět se zde vyskytují rozhraní `getPayload`, `getIPRoute`, která nám zajišťují korektní datový výstup naší aplikace.




**Obr. 4-2 Abstraktní model protokolu MSN**

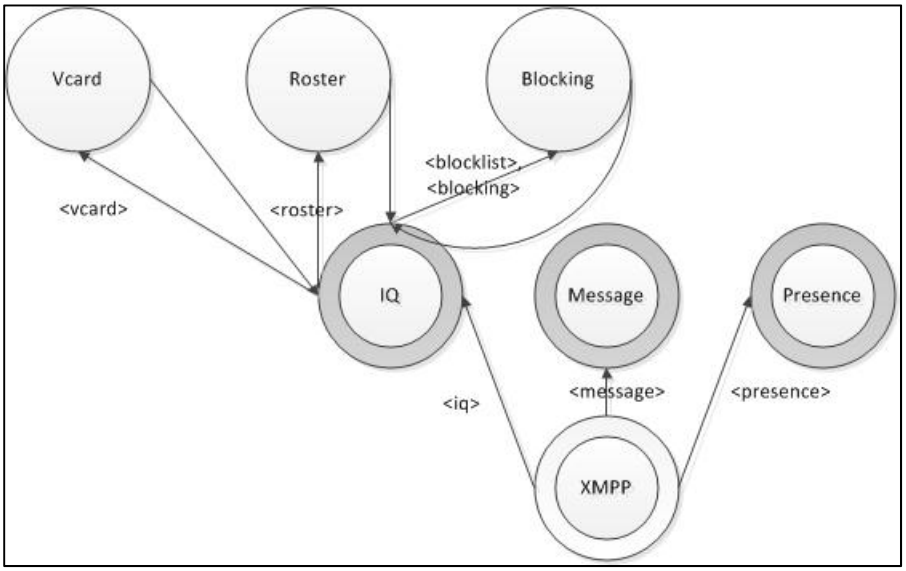
### 4.1.2 Implementace abstraktního modelu

Jelikož se nejedná o složitý model a důležitou vlastností je rozšiřitelnost, byla část aplikace, která zprostředkovává rekonstrukci IM komunikace implementována na základě jednoduchého konečného automatu.

#### 4.1.2.1 Popis konečného automatu protokolu XMPP

Počátečním stavem je XMPP, kde dochází načtení vnitřního obsahu a výběru přechodu. Stav XMPP předává řízení dalším stavům a ty jsou:

- *IQ* - Zpracovává obsah *iq stanzy*. Pro splnění podmínek přechodu je nutná zpráva s `<iq>` elementem. Jelikož je IQ stav nejrozšířenější, vyskytují se zde další stavy a to:
  - *Vcard* – Podmínka přechodu je `<vcard>` element.
  - *Roster* – Podmínka přechodu je `<roster>` element.
  - *Blocking* – Podmínka přechodu je `<blocklist>`, nebo `<blocking>` element.
- *Message* – Zpracovává zprávy od uživatelů v podobě *message stanz*. Podmínka přechodu z XMPP do Message stavu je `<message>` element.
- *Presence* – V tomto stavu dochází ke zjištění přítomnosti uživatele. Přechod je proveditelný v případě výskytu `<presence>` elementu.
- V případě `IQ` stavu se můžeme za pomocí elementů `<vcard>`, `<roster>`, `<blocking>` dostat do stavů `Vcard`, `Roster`, `Blocking`, které po úspěšném dokončení analýzy obsahu provedou přechod zpět do stavu `IQ`.

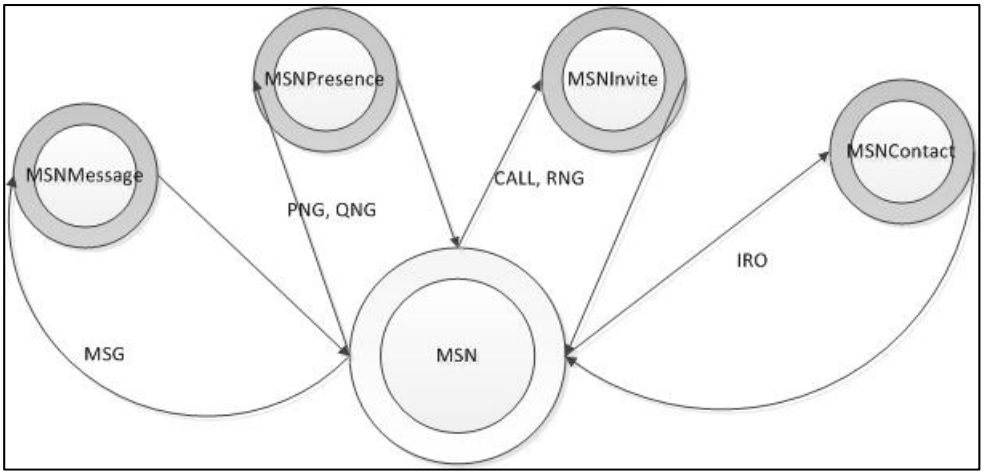



**Obr. 4-3 Konečný automat sloužící k rekonstrukci IM pro část XMPP**

#### 4.1.2.2 Popis konečného automatu protokolu MSN

Počátečním stavem je MSN, kde dochází k výběru konkrétního stavu pro načtení obsahu zprávy.

- *MSNMessage* – Zpracovává uživatelské zprávy. Přechod je proveditelný v případě výskytu `MSG` příkazu.
- *MSNPresence* - Zpracovává přítomnost uživatele v síti. Přechod je proveditelný pro příkazy `PNQ` a nebo `QNG`.
- *MSNInvite* – Stav reprezentuje zahájení komunikace s uživatelem. Přechod je proveditelný pomocí `CALL` a `RNG` příkazů.
- *MSNContact* – Jedná se o stav zpracovávající kontakty uživatele. Do tohoto stavu se dostaneme za pomocí `IRO` příkazu



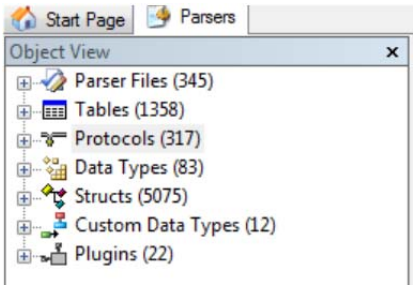

**Obr. 4-4 Konečný automat sloužící k rekonstrukci IM pro část MSN**

Výsledná aplikace se skládá z následujících bloků:

- `Main`
  - Vstupní bod programu kde dochází k volání ostatních částí
- `getParams`
  - Načítání, validace zadaných parametrů
- `getPayload`
  - Deleguje rekonstruování IM komunikace na část XMPP, MSN přes definované rozhraní (`getPayload`)
- `indent`
  - Získaný XML výstup z části XMPP, MSN je díky této pomocné funkci upraven do čitelné podoby, aby bylo možné provést přibližnou kontrolu generovaného výstupu při testování

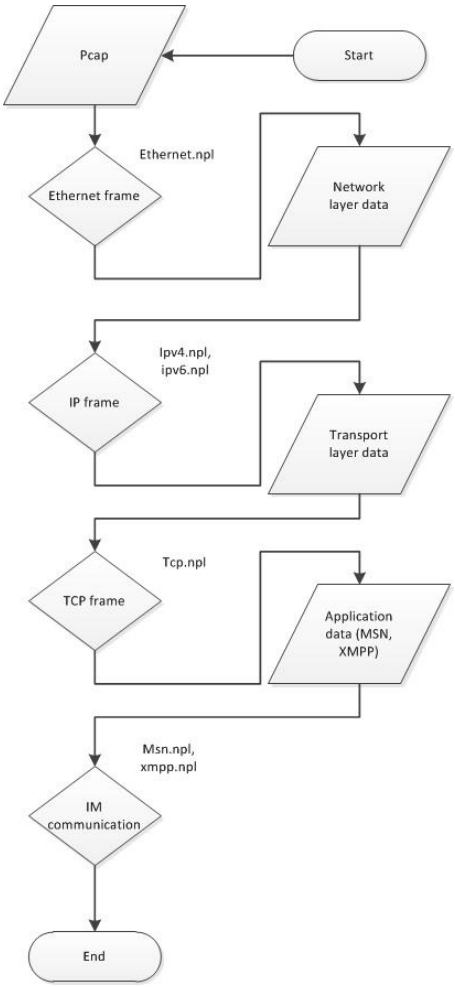
## 4.2 NPL (Network Parsing Language)

Jazyk pro zpracování síťových dat (NPL) byl vyvinut společností Microsoft a zakomponován jako rozšíření do aplikace MNM. MNM slouží k odchytávání a rozpoznávání síťových paketů. Aby bylo možné jednotlivé pakety rozeznat a patřičně je zobrazit, jsou v MNM definovány parsery síťových protokolů, které jsou psány v jazyce NPL. Spousta analyzátorů je již předem definována a dodávána společně s produktem MNM. Součástí nejsou jenom analyzátory definující formát a zobrazení protokolů, ale najdeme zde i předdefinované datové typy, tabulky, které přiřazují binárním hodnotám řetězce, ale také zásuvné moduly, které rozšiřují možnosti využití. Základ tvoří stovky předdefinovaných parserů pro popis PDU na jednotlivých vrstvách ISO/OSI modelu. Všechny ostatní přídavné parsery je potřeba ručně přidat do souboru *my_parser.npl*, který se nachází v *Dokumenty\Network Monitor 3\Parsers*.




**Obr. 4-5 soubory analyzátorů v Parser Files [4]**

Postup při rozeznávání paketu:




1. Zachycená data v podobě PCAP souboru jsou porovnávána s analyzátory.
2. Dále jsou data kromě dat fyzické vrstvy předána parserům síťové vrstvy.
3. Jako další následují analyzátory transportní vrstvy (zde probíhá registrace analyzátoru MSN a XMPP, jelikož další vrstva je již aplikační, která obsahuje data IM komunikace).
4. Parsery aplikační vrstvy (MSN, XMPP).

### 4.2.1 Syntaxe a sémantika NPL

Předtím než se začneme zabývat konstrukcemi tohoto jazyka, je potřeba si vysvětlit, jakým způsobem dochází k rozpoznání jednotlivých částí datového paketu a jakým způsobem jsou jednotlivým částem přiřazeny protokoly. Veškeré informace jsou čerpány ze zdroje [4]. Jako demonstrační příklad budou použity úryvky z parseru *hsrp.npl*. Myšlenkou je stromové větvení jednotlivých analyzátorů. Na nejnižší vrstvě dochází k aplikování analyzátorů pro rozpoznání typu rámce. Aby bylo možné tyto analyzátory aplikovat je třeba registrace v MNM. Jelikož tyto kroky za nás obstarává základní balíček, není třeba se tím dále zabývat. Zbývá nám zaregistrovat vlastní analyzátor provázáním k patřičné vlastnosti jiného analyzátoru. (obsah, port, atd.).

```
case 1900:
   HTTP http;
case 2049:
   RPC Rpc;
```

**Obr. 4-6 obsah vlastnosti port, udp analyzátoru [4]**

```
[ RegisterAfter (UDP.http, Hsrp, 1985) ]
Protocol HSRP
{
}
```

**Obr. 4-7 Registrace HSRP analyzátoru (protokolu) na port 1985 [4]**

Nyní můžeme začít s popisem samotného protokolu. Na začátek je nutné ještě podotknout, že většina konstrukcí v MNM produkuje implicitně výstup, který je zobrazen uživateli. Celý popis se nachází v těle klíčového slova `Protocol`.

- Protocol [název] { data } (viz Obr 4.2.2)

Uvnitř složených závorek máme k dispozici binární hodnoty, kterým přiřazujeme datové typy, manipulujeme s nimi a jsme schopni je zobrazit. Uvnitř složených závorek máme k dispozici zbytek nepřiřazených binárních hodnot, které začínají od konce posledního analyzátoru. Aby bylo možné přiřadit jednotlivým hodnotám proměnné a posouvat se tak v hodnotách dále, vyskytují se zde datové typy. Aby bylo možné se pomocí proměnných posouvat dále, je třeba proměnou definovat normální notací a nikoli přetypováním (pokud je proměnná vytvořena přetypováním, nedochází k posunu v datech a proměnná nezabírá místo v rámci)

- *Primitivní datové typy* – Decimal, Number, Time.
- *Vlastní datové typy* - UINT, String, Xml.
- *Bitové proměnné* – V případě, že je potřeba aby proměnná nezabírala velikost celého datového typu, ale jenom část, je toto chování umožněno definováním dvojtečky a počtu bitů za název proměnné.

Každá z proměnných tvoří ve výstupu samostatné pole. Pokud je například žádoucí určitá pole seskupit a nabídnout možnost skrytí a zobrazení, jsou pro tento účel definovány struktury. Kromě struktur je možnost rozšíření proměnné o složené závorky, pomocí kterých docílíme stejného výsledku. Existují dva typy struktur:

- *Veřejné* – Seskupují data a nabízí možnost skrytí a rozbalení seskupených dat.
- *Soukromé* – Neseskupuje data a tudíž zde není možnost skrytí, či rozbalení. Uplatnění nachází v případech, kdy můžeme definovat pouze jediný příkaz (datovou proměnnou). Toto chování se vyskytuje například v konstrukcích jako je `switch` či `while`. Uvnitř struktury je možné definovat libovolné množství proměnných.

Pokud se v hodnotách vyskytují periodicky se opakující data, není třeba zdlouhavě vypisovat všechny proměnné. K tomuto účelu je definován příkaz `while`.

`while` – Konstrukce se podobá jazyku C. Tělo smyčky je vykonáváno tak dlouho, dokud je podmínka v hranatých závorkách vyhodnocena jako pravdivá

```
[MaxLoopCount = 10]
while [FrameOffset < FrameLength]
{
   uint8 x;
}
```

**Obr. 4-8 Příklad while konstrukce [4]**

Jelikož se v protokolech vyskytuje i logika, pomocí které se rozhodujeme, o jaký typ informace se jedná, máme k dispozici konstrukci `switch`.

`Switch` – Jedná se o hlavní konstrukci pro řízení logiky. Jelikož zde chybí konstrukce `if`, obsahuje `Switch` alternativní zápis, který nahrazuje konstrukci `if`.

```
switch(Opcode) {
   case 0:
      struct {
         UINT8 Hellotime;
         UINT8 Holdtime;
      }
   default:
      BLOB(2) Implementation
}
```

**Obr. 4-9 Switch konstrukce [4]**

V případě, že je potřeba výstupní formát proměnných upravit, není toto v jazyce NPL umožněno. Nicméně je možnost vytvořit zásuvné moduly, které plní některé úkony, které jazyk NPL neumí. Opět je již spousta zásuvných modulů k dispozici v aplikaci Network Monitor. Ze zásuvných modulů, které jsem ve své práci použil je `FormatString`.

- `FormatString` – Umožňuje naformátovat výstup jednotlivých konstrukcí (jedná se o zásuvný modul). Všechny konstrukce podporují rozhraní pro zobrazení, tudíž je možnost

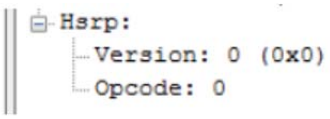
konstrukci jako je *Protocol* přiřadit nějaký text, který se má zobrazit ve výstupu (*FormatString* je možné přirovnat k funkci *printf* jazyka C).

```
Protocol HSRP
{
   UINT8 Version;
}
```

**Obr. 4-10 Zobrazení verze [4]**

```
UINT8 Version;
UINT8 Opcode = FormatString("%u", this);
```

**Obr. 4-11 Úprava zobrazení výstupu Opcode [4]**




**Obr. 4-12 Výsledné zobrazení pomocí FormatString modulu [4]**

## 4.3 Popis XMPP a MSN pomocí NPL

### 4.3.1 XMPP

Ze všeho nejdříve je nutné analyzátor pro protokol XMPP zaregistrovat. V implicitním režimu jsou data přenášena na portu 5222 protokolu TCP. Na základě těchto údajů jsem zvolil zaregistrování analyzátoru za analyzátor *SIP* s hodnotou portu 5222.

```
[ RegisterAfter (TCPPayload.SIP, XMPP, 5222) ]
```

**Obr. 4-13 Registrace analyzátoru na analyzátor nižší vrstvy**

Data protokolu XMPP jsou zapouzdřena do formátu XML. Jelikož se v MNM nachází analyzátory XML dokumentů, máme možnost tento analyzátor nad daty zavolat a nechat si výstup zobrazit jako XML strom, který je oproti binární podobě mnohem čitelnější.

```
Xml(0) XmlPayload;
```

### 4.3.2 MSN

Jako první věc je potřeba vytvořit pravidlo, pomocí kterého bude MNM analyzátor pro MSN uplatňovat. Jelikož je MSN komunikace uskutečněna v režimu TCP nad portem 1863, zvolil jsem zaregistrování analyzátoru za port *MQQB* analyzátoru TCP.

```
[ RegisterAfter (TCPPayload.MQQB, MSN, 1863) ]
```

**Obr. 4-14 Registrace analyzátoru MSN k analyzátoru nižší vrstvy**

Po úspěšném zaregistrování vytvoříme definici protokolu s názvem MSN a použitím zásuvného modulu `FormatString` pro přidání typu zprávy. Nyní už můžeme začít s rozeznáváním binárních hodnot, které máme v rámci protokolu k dispozici. První 3 byty jsou znaky, které určují typ příkazu MSN. Jelikož se data v závislosti na typu příkazu liší, použil jsem příkaz `switch` (provede se vždy jen návěští, které se shoduje s podmínkou v příkazu). Jako podmínka je použit datový typ `String`, který je v režimu přetypování, tudíž nejsou data zabrána datovým typem, ale jen využita k porovnání. Jelikož se ostatní návěští příkazu `switch` od sebe nikterak neliší, uvádím pouze ta, která využívají příkazů, které doposud nebyly vysvětleny. Jako příklad jsem vybral návěští pro příkaz `MSG` (Message, zpráva od uživatele).

```
Switch (String(FrameData, FrameOffset, 2, 3))
{
    case "MSG":
        _struct msg
        {
```

**Obr. 4-15 Návěští pro rozeznání MSN příkazu**

V podkapitole věnované popisu významu struktur bylo zmíněno, že pokud je před deklarací struktury uvedeno podtržítko, jedná se o soukromou strukturu, která nezabírá data. Použití soukromé struktury v návěští *case* je kvůli použití více příkazů pro přiřazení dat do datového typu (bez použití struktury je možné v návěští *case* uvést pouze jeden příkaz). První příkaz je typu `String` s názvem *name*, které určuje název MSN příkazu. V případě, že zpráva typu MSN obsahuje text, druhému uživateli, obsahuje UID (id uživatele). V opačném případě se může jednat pouze o potvrzovací, informační zprávu s ID zprávy. Pro tyto účely je potřeba použít konstrukce `if`, která nám zajistí správné přiřazení typu zprávy na základě celočíselné, nebo textové hodnoty. Jelikož jazyk NPL konstrukci `if` neobsahuje, musíme si vystačit s konstrukcí `switch`, která je schopna příkaz `if` také vykonat (`switch` obsahující pouze dvě návěští může vykonávat stejné operace jako konstrukce `if-else`, popřípadě `if-else if`).

```
Switch (StringTerm(FrameData, FrameOffset, " ", 0))
{
    case value > 0:
        AsciiStringTerm(" ", 0) id = FormatString("Message Counter, %s", this);
    default:
        AsciiStringTerm(" ", 0) UID = FormatString("User ID, %s", this);
}
```

**Obr. 4-16 konstrukce if-else v podobě konstrukce switch**

Dále se snažíme z rámce vyčíst zbývající velikost dat pomocí `AsciiStringTerm`, která je zachycena v datovém typu `Length`.

Na konec návěští se vyskytuje příkaz cyklu jazyka NPL. Tento příkaz je obalen ve struktuře, aby data, která jsou v cyklu obsažena, bylo možné skrýt, nebo zobrazit.

```
AsciiStringTerm("\r\n", 0) Length;
struct Content
{
    while [ FrameOffset < FrameData ]
    {
        AsciiStringTerm("\r\n", 0) part;
    }
}
```

**Obr. 4-17 proměná length, příkaz while**

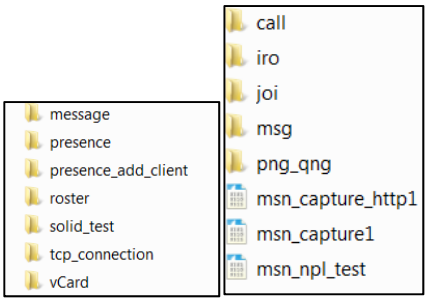
Cyklus `while` je prováděn tak dlouho, dokud data obsažená v rámci (`FrameData`) jsou větší než aktuální posun v rámci (`FrameOffset`). Jelikož chceme zobrazit obsah po řádcích, vystačíme si s jedním příkazem, který nám vytváří proměnné *part*, které pojmou data od začátku ke konci řádku.

# 5 Testování

Po implementaci systémů pro protokoly XMPP a MSN bylo zapotřebí ověřit, zda tyto systémy produkují očekávané výstupy. Jelikož systémy pro rekonstrukci IM komunikace produkují odlišný výstup od vstupních dat protokolů XMPP a MSN a nedají se tudíž jednoduše porovnat pomocí krátkého skriptu, vytvořil jsem sadu krátkých pcap souborů, které se zaměřují na jednotlivé části IM komunikace (například seznam kontaktů, zpráva mezi uživateli, režijní zprávy, změna stavu atd.). Všechny tyto soubory se vyskytují i se zdrojovými kódy systémů na digitálním médiu jako příloha. Testování spočívalo v porovnávání údajů obsažených v nasbíraných pcap souborech s vygenerovanými výstupy systému.

## 5.1 Test Python skriptu

Pro systém v Python skriptu byl pro každý klíčový prvek IM komunikace vytvořen pcap soubor.




**Obr. 5-1 Složky obsahující pcap soubory pro protokoly XMPP a MSN**

```
$ python2.6 xmpp_msn.py --iFile=roster --oFile="roster.xml"
```

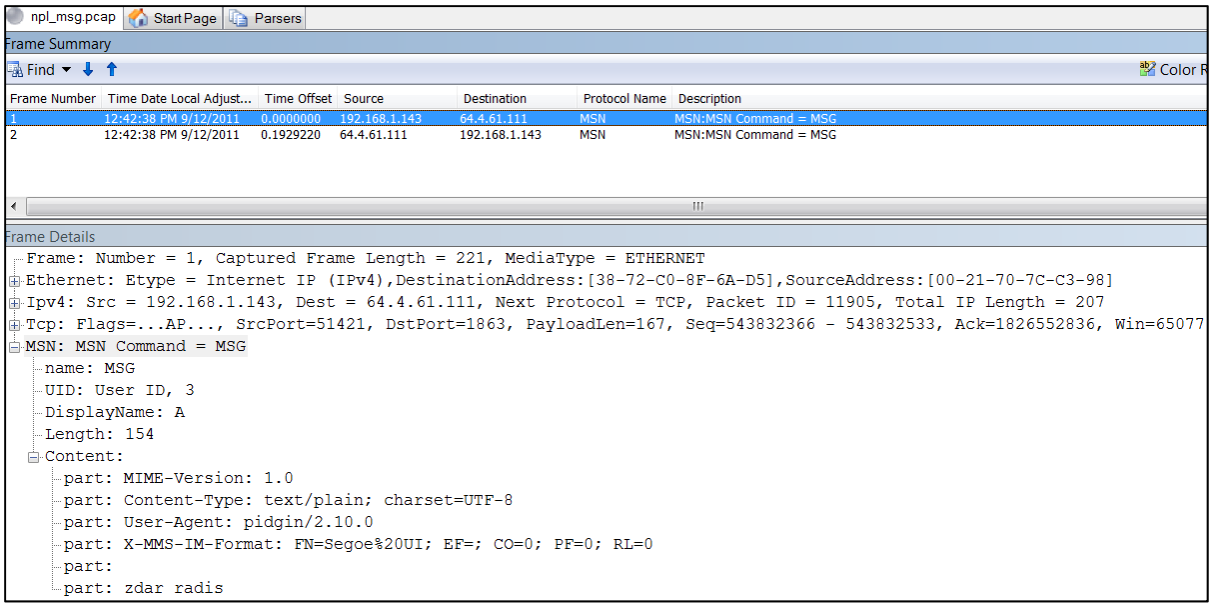
**Obr. 5-2 Zadání roster pcap souboru na vstup systému v jazyce Python, výstup do souboru roster.xml**

Následně byly pomocí vytvořeného systému nad testovacími pcap soubory vytvořeny výstupy, které byly pečlivě porovnány vůči vstupním pcap souborům zda jsou zachyceny všechny informace.

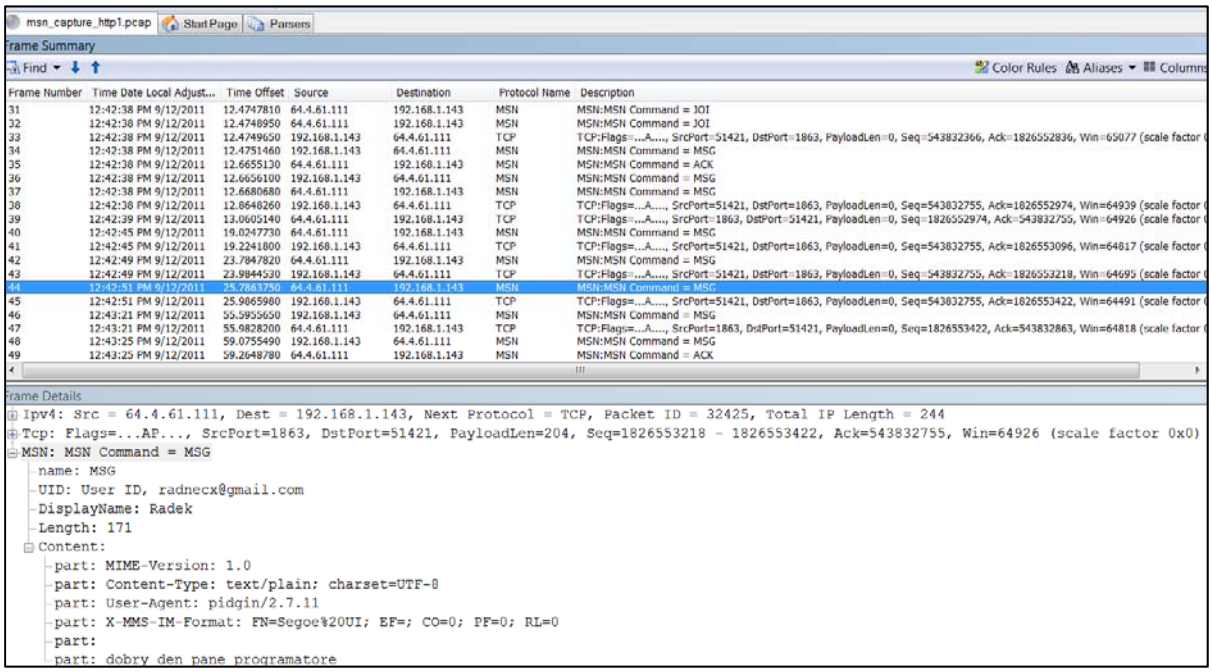

**Obr. 5-3 Ukázka porovnávání pcap souboru (vlevo) s vygenerovaným výstupem (vpravo).**

## 5.2 Test NPL skriptu

Testování NPL skriptu proběhlo v prostředí MNM. Aby byly odhaleny chyby jednotlivých částí IM komunikace, byly nejdříve využity pcap soubory zaměřující se na tyto části. Následně byl použit pcap soubor, který obsahoval všechny předešlé konstrukce IM komunikace, aby bylo ověřeno, že skript zvládne jakékoliv konstrukce (na neimplementované konstrukce v systému je nutné upozornit, aby nedocházelo k zmatení, tudíž jsou neimplementované konstrukce značeny NA příznakem).




**Obr. 5-4 Výstup npl skriptu v prostředí MNM zobrazující zprávu mezi uživateli**




**Obr. 5-5 Zobrazení pcap souboru testující veškeré konstrukce IM komunikace (zachyceno pro protokol MSN)**

# 6 Závěr

Tato práce se zabývá popisem dvou technologií využívaných pro komunikaci v reálném čase (XMPP, MSN). Byly vysvětleny základní principy a seznámení se s vyskytujícími se zprávami. Díky tomu jsme byli schopni objasnit sémantiku probíhající komunikace. Byly provedeny pokusy o zachycení komunikace, které byly zanalyzovány a srovnány s konstrukcemi získanými z dostupných materiálů. Na základě těchto pokusů následoval výběr klíčových konstrukcí pro možnou rekonstrukci komunikace mezi uživateli. MSN je proprietární protokol, proto nebylo možné najít oficiální zdroj, pomocí kterého bych byl schopen ověřit validitu zrekonstruovaných konstrukcí. Proto jsem se snažil odchytit co nejvíce IM komunikace pro odlišné verze protokolu MSN a tuto komunikaci potom porovnat s nejdůležitějšími konstrukcemi, které byly pro protokol MSN vybrány. Dále jsem se v této práci věnoval implementaci systému pro rekonstrukci jednotlivých zpráv v jazycích Python, C#. Systémy pro analýzu a rekonstrukci protokolů XMPP a MSN by měly umožnit získat důležité informace z obsahu komunikace a vytvořit patřičný výstup těchto informací, který je předurčen k dalšímu zpracování. Tato práce se zabývala analýzou a rekonstrukcí IM komunikace dnes dvou nejpoužívanějších protokolů (XMPP a MSN) a myslím si, že do budoucna je možné dále na těchto základech stavět. Muže se například jednat o rozpoznávání sezení mezi uživateli podle časového uspořádání, nebo o kontextové rozpoznávání komunikace a vytvořit tak profil uživatele, který by obsahoval údaje o vztazích vůči ostatním uživatelům.

# Reference

[1]. **Peter, Saint-Andre, Kevin, Smith and Remko, Troncon.** *XMPP The Definitive Guide.* s.l. : O'REILLY, 2009.

[2]. **Saint-Andre, Peter.** *XMPP Core.* [RFC 6120] 2011. ISSN 2070-1721.

[3]. —. *XMPP Instant Messaging and Presence.* [RFC 6121] 2011. ISSN 2070-1721.

[4]. **Hawker, Michael A.** *Writing a Parser from Wire to Window.* [Dokument] s.l. : Microsoft, 2009.

[5]. **Braekevelt, Matthias.** *Web MSNPiki.* [Online] http://msnpiki.msnfanatic.com.

[6]. **freezing...@gmail.com.** Documentation for changes in MSN Protocol Version 21. [Online] http://code.google.com/p/msnp-sharp/wiki/KB_MSNP21.