Design of Data Retention System in IPv6 network

FIT VUT Technický report

Tomáš Podermański, Matěj Grégr, Miroslav Šoltés, Martin Žádník

Technický report č. FIT-TR-2011-07
Fakulta informačních technologií, Vysoké učení technické v Brně

Last modified: December 7, 2011

Design of Data Retention System in IPv6 network

Tomáš Podermański[1], Matěj Grégr[1], Miroslav Šoltés[1], and Martin Žádník[1]

Brno University of Technology, email: tpoder@cis.vutbr.cz, igregr@fit.vutbr.cz, xsolte00@stud.fit.vutbr.cz, izadnik@fit.vutbr.cz

Abstract. Data monitoring and data retention are vital for network management and troubleshooting. The network and provided services are expected to be seamlessly available. Administrators often collect information about the on-going traffic in the form of IP flow records to reveal potential malicious activity that might violate the network usage policy but also to meet legal requirements on providing data about electronic communication to authorized organizations. It is vital not only to collect data about traffic but also to track the identity of users who are responsible for the traffic. The deployment of IPv6 renders unique user identification quite problematic or at least a complex task in comparison with IPv4 environment. In this report, we suggest a data retention system with user identification capabilities in IPv6 as well as in IPv4 network. This is achieved by extending flow records with information obtained by monitoring state of network devices via SNMP and monitoring state of control servers such as Radius. The system has been successfully deployed in BUT network.

1 Introduction

Data retention system generally allows a network operator or law enforcement agencies to track malevolent users or security incidents. The network data retention system, especially in large networks, has to be able to handle a significant volume of data. Storing the network traffic itself is not a feasible option due to limited storage capacity and speed. Therefore the system is usually based on some form of flow monitoring - e.g. NetFlow. NetFlow data provides necessary information for data retention - source, destination, duration and type of communication together with amount of transferred data. However, are NetFlow data sufficient for the data retention system, if IPv6 protocol is deployed in the network?

The IPv6 protocol creates new challenges. Unlike IPv4, an IPv6 address no longer identifies a user or PC uniquely, because an IPv6 address can be randomly generated and keeps changing. This document discusses general specification of data retention system according to ETSI TS 102 657 [2] document and address the major monitoring issues of IPv6 connectivity. A practical solution for monitoring both IPv4 and IPv6 traffic is proposed. The proposed data retention system is able to monitor and identify a user in both IPv4 and IPv6 traffic. The solution requires an extension of the monitoring data collected from network devices. A new data structure based on extension of NetFlow records is presented.

Data retention system is deployed at the Brno University of Technology (BUT) campus network. The document discusses the results of data retention traffic monitoring and future challenges.

2 IPv6 and user identification

Many internal resources require the ability to track the end user's use of services. IPv6 address tracking (or data retention) is also a legal obligation of ISPs required by governments. If a local security policy requires better control, either fixed IPv6 addresses must be centrally assigned and logged, or stateless configuration using DHCPv6 has to be deployed. If stateless auto-configuration is deployed, a new monitoring system is required.

Temporary IPv6 Addresses: Auto-configuration is a new IPv6 feature that allows a node to automatically generate an IPv6 address on its own. This behavior is different from IPv4 address configuration, in which the IP address is configured either manually or using DHCP. An IPv6 node can be configured through either stateless or stateful auto-configuration. The basic stateless configuration combines a network prefix obtained from the router with the IEEE EUI-64 identifier based on the MAC address. This allows keeping the link between an address and user/host, but the host part of the address can easily be tracked all over the Internet.

Because of user privacy, IPv6 addresses with randomly generated 64-bits interface identifiers are preferred instead of IEEE EUI-64. The RFC 4941 standard defines a way to generate and change temporary addresses. The important requirement is that the sequence of temporarily generated addresses on the interface must be totally unpredictable.

However, this requirement contradicts the need to identify a malevolent user. Private, temporary addresses hinder the unique identification of users/hosts connecting to a service. This affects logging and prevents administrators from effectively tracking which users are accessing what services.

Stateful IPv6 configuration: The DHCP Unique Identifier (DUID) can be used to identify a user in an IPv6 network. However, DUID has several disadvantages. Its value is not easily searchable, since every client stores its values at different places on the local disk. The value is changed whenever the operating system is reinstalled. Experience at BUT shows that using stateful configuration for address assignment is extremely difficult. First of all, even if an IPv6 address is assigned to a host with Windows 7 or Vista using DHCPv6, the host will not use this address for communication but will use a temporary address instead. Secondly, the DHCPv6-client is not supported in Windows XP, which is still widely in use.

Stateless IPv6 configuration: The first part of the IPv6 address - the network prefix - is assigned using RA messages as described in the previous chapter. RA messages do not provide any type of unique identifiers that could be used to identify the host. The second part of the IPv6 address - the interface ID - is generated using EUI-64 or privacy extensions. EUI-64 could be used for

host identification since its value is derived from the MAC address. However, Windows 7 and Vista use randomly generated interface ID's instead, by default. Thus, neither stateful nor stateless configuration provides the unique ID needed for user identification. More information has to be obtained, as discussed in the following section.

3 Proposed architecture

The proposed architecture follows the standard defined in ETSI document TS 102 657 [2]. Although the architecture of data retention system is only briefly described it provides high level overview of the whole system as well as it defines several basic building blocks.

At the top level ETSI defines two communicating entities, Communication Service Provides (CSP) and Authorized Organization (AO). ETSI suggests to establish two communication channels between AO and CSP as a Handover Interface (HI). The first channel (HI-A) delivers administrative request/response information whereas the second channel (HI-B) transfers only Retained Data (RD). Figure 1 displays this model.

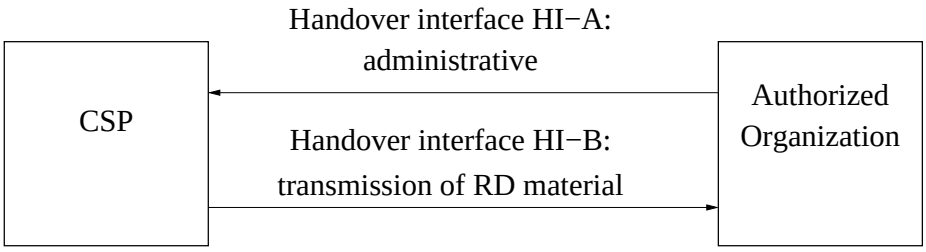

Fig. 1. Model of CSP and AO Handover Interface (taken from [2])

The architecture and processes of AO are rather out of the scope of this document. We focus on breaking down the CSP block. Based on ETSI, three functional blocks can be identified within the CSP.

An administrative function (AF) implements both channels of HI and internal interface to acquire retained data from Data Store Management Function (DSMF). The task of AF is then to receive and acknowledge requests for RD, transform and issue these requests in a syntax of DSMF, report on the state of on-going queries and finally deliver the result of the queries as RD over HI-B.

A Data Collection Function collects data from the various internal network elements and prepare the data for retention. This includes both synchronous and asynchronous communication with probes, switches, routers, servers such as DHCP, DNS, RADIUS and user database.

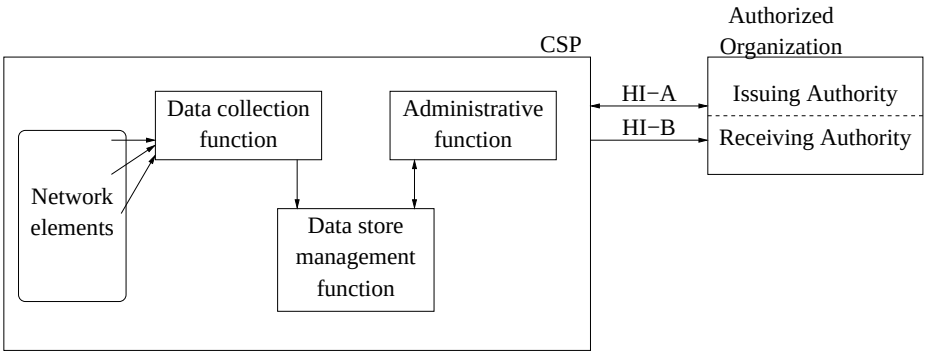

Fig. 2. Break down of CSP function (taken from [2])

At last ETSI defines a data store management function. This process handles indexing and storing the data, executing queries and managing the maximum retention period.

Please note that the ETSI TS 102 657 states that the the decomposition of internal architecture is only informative and may vary according to specifics of the CSP. Moreover, it also allows for outsourcing some functions to a third party depending on the national agreement.

We propose a monitoring system that fits the needs of data retention in IPv6 network. It considers the various sources of collected data, their processing and storage as well as presentation interface. The architecture of the system is depicted in Figure 3.

The Figure displays a mapping of ETSI blocks on implementation primitives that are relevant for IPv4 and IPv6 network. Let's start with description of DCF. There are three data sources of different type at the input of the system. The first data source constitutes NetFlow generated by probes and routers in the network. The NetFlow data are collected and stored. The second data source constitutes of SNMP which allows to transfer data from switches and routers. SNMP poll reads accessible variables in each device based on its MIB tree and stores this data in the database. The third source of data are event messages and logs of management servers such as RADIUS, DHCP. These data are parsed and extracted information is stored in database as well. The DSMF function joins the data in database with stored NetFlow data. A configuration interface allows to setup and control DCF and DSMF processes. A data interface handles queries on stored data generated by various applications among which belongs an AF function.

We propose to assemble several various tools to implement whole monitoring system according to the proposed architecture. In the following sections, we discuss the specifics of each data retention block and we describe how each block is implemented.

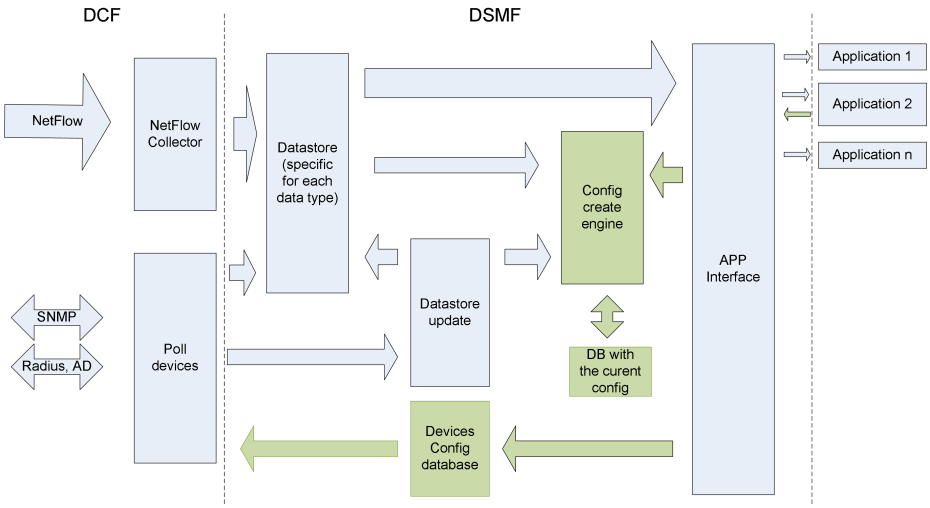

Fig. 3. Proposed architecture of monitoring system

3.1 Data Collection Function

A Data Collection Function collects data from the various internal network elements and prepare the data for retention. ETSI breaks down Collected data into following categories (for each category we list some data examples and their possible source within the scope of IP network).

1. Subscriber data: information relating to a subscription to a particular service (Data: Name, Address, Identifier; Source: User Database).
2. Usage data Information relating to usage of a particular service (Data: Flow Records, SIP records; Source: NetFlow, IPFIX, SIP proxy, ...).
3. Equipment data: information relating to an end-user device or handset (Data: MAC address, OS; Source: Neighbor Cache/Forwarding table in switch, DHCP server, RADIUS server).
4. Network element data: information relating to a component in the underlying network infrastructure (Data: location and identifier of an access point, statistics from interfaces; Data: Network elements via SNMP)
5. Additional service usage: information relating to additional services used (Data: SMTP, IMAP; Source: Application servers).

DCF must be able to collect data via various interfaces and protocols such as syslog, SNMP, NetFlow/IPFIX, file logs and database interface. Some elements work asynchronously, i.e., transmit data on its own upon an event such as value exceeding a threshold, timer expiration and others. Some elements work synchronously, that is, they must be actively queried for data. The output of DCF is data ready for retention.

The variability of communication as well as the variability of devices manufactured by various vendors renders the DCF very complex. We propose to adopt and customize existing solutions for network administration and monitoring which already cover this complexity.

From a wide variety of network tools Network Administration Visualized (NAV) [4] suite (a collection of libraries put together with output to database and handy GUI) seems to be good solution for collecting data from network elements. NAV has been developed since 1999 and nowadays it is maintained by UNINETT. NAV is freely distributed under the GNU GPLv2 license and supports both IPv4 and IPv6 deployment. NAV is able to poll or receive SNMP messages and events from over 80 network devices (most commonly switches and routers) from 11 different vendors. Moreover, it is able to store logs and messages from Radius server as well.

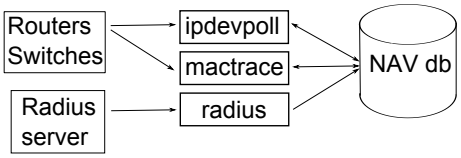

Fig. 4. Part of NAV architecture

Figure 4 displays subset of the whole NAV architecture. It consists of several backend processes that runs as either daemons or cron jobs. These processes fills up the NAV database (PostgreSQL). Although NAV consists of many processes this report focuses only on those that are relevant for data retention.

The crucial part for DR is to collect information about an association of a user, user's IP address, MAC address and switch port the user is connected to. This allows to answer typical queries such as who is behind given IP address, what is the user's location, what addresses does the user use. SNMP is used to obtain data from switches every fifteen minutes. The mapping between the IPv6 address and its corresponding MAC address is downloaded from the router's neighbor cache. Port, VLAN number and other information comes from the switch's FDB (Forwarding Database) table.

First of all, the administrator must seed the NAV database with IP addresses of monitored network elements. The ipdevpoll polls each device for inventory information (includes interfaces, serial numbers, modules, VLANs and prefixes), for load information and for logging information such as ARP tables or Neighbor Discovery cache. The obtained information is regularly stored in the database. NAV defines following ARP table schema:

The mactrace process collects mac addresses, port, vlan and other information from Forwarding Database table for all switches and stores the information

Table 1. Schema of ARP table in NAV DB (taken from NAV doc [4])

arpid	primary key
netboxid	router the arp entry comes from
sysname	the same router in name (in case the router is deleted, arp has historic data)
prefixid	prefix the arp entry belongs to
ip	ip address of the arp entry
mac	mac address of the arp entry
start_time	time the arp entry was first discovered
end_time	time the arp entry disappeared (typically 4 hours after the last packet sent).

into CAM table. Its schema is described in Table 3.1. The process also checks for spanning tree blocked ports.

Table 2. Schema of CAM table in NAV DB (taken from NAV doc [4])

camid	primary key
netboxid	switch that has the cam entry
sysname	name of the same switch, in case switch is deleted, cam data are historic
ifindex	infindex of the switch port for the cam entry
module	module number for the cam entry
port	port number for the cam entry
mac	mac address found on the port
start_time	time the mac address was first seen
end_time	time the mac address disappeared (idle timer in bridge tables are typically 5 minutes)
misscnt	count how many times the cam entry has been tried updated and failed. We do not want to terminate these cam entries right away. It is configurable how many misscnt the camlogger should tolerate.

NAV also supports collection of RADIUS data from a FreeRadius server. The FreeRadius server can be configured to push data about authentication requests directly into NAV database. In addition to this interface, there is a NAV radius process which regularly parses the FreeRadius log file. It extracts important messages and pushes them in the database as well. The most important columns of NAV table are described in table 3.1.

We have added a support of another Radius server Radiator [?]. Radiator provides event-hooks which we utilize to parse on-going authentication requests.

Table 3. Schema of RADIUS table in NAV DB

username	username used as login to RADIUS
radacctid	Radius accounting ID
acctsessionid	Accounting Session ID
acctuniqueid	Accounting Unique ID
nasipaddress	IP address of an access point
acctterminatecause	Accounting Terminate Cause
acctstarttime	Start timestamp of accounting (time of log in)
acctstoptime	Stop timestamp of accounting (time of log off)
calledstationid	ID of called station
callingstationid	ID of calling station

Parsed requests are temporarily stored in a file. Another process picks up these stored requests, transforms them into a schema of NAV radius table in NAV database and performs an insert operation. The separation of Radiator and parsing process allows to store messages when the NAV database is temporarily unreachable. It also allows to select and store only those items which fit the current Radius table schema. Probably the most important fields are username and callingstationid. In case of IP network we store a MAC address of a user as a callingstationid.

```
timestamp            username           callingstationid   result
-------------------------------------------------------------------
2011-11-21 12:58:46  xsejno00@vutbr.cz  00:1d:e0:0a:d0:7d  accept
2011-11-23 09:21:49  hazmuk@vutbr.cz    00:21:5c:81:61:91  accept
2011-11-23 07:32:11  xkisli01@vutbr.cz  00:13:02:51:e0:fd  accept
2011-11-22 21:31:58  2763@vutbr.cz      00:26:82:1b:c0:cd  accept
2011-11-22 16:02:30  xkisli01@vutbr.cz  00:13:02:51:e0:fd  reject
2011-11-21 20:45:03  tpoder@vutbr.cz    58:1f:aa:82:39:6c  accept
2011-11-21 20:23:54  xgregr01@fit.vutb  00:26:82:e5:6b:0f  accept
```

We have further extended the schema of radius table with unique user-ID. A database trigger is implemented to check if a username in the incoming message already exists in the table. If so, then its unique user-ID is used if not then a new ID is generated for this user. The unique user-ID enables to join collected information with other external data such as NetFlow records (this will be discussed in further sections).

The collected ARP entries (MAC-IP pair) and CAM entries (MAC-port pair) and Radius entries (username-MAC or username-IP address) enable to keep relation among users, IP addresses, MAC addresses and switch ports. This is especially important in IPv6 deployment where a user typically generates its own IP addresses and changes them regularly. Moreover, all records contain start time and end time of their validity. This is important for handling history-oriented queries which are typical for data retention system.

NAV does not collect any information regarding the network traffic itself. Therefore it must be complemented with other tool capable of collecting these data (as depicted on Figure 5). Based on our previous experience, we select to gather traffic metadata via NetFlow. Collecting NetFlow records is typically supported by routers or dedicated probes.

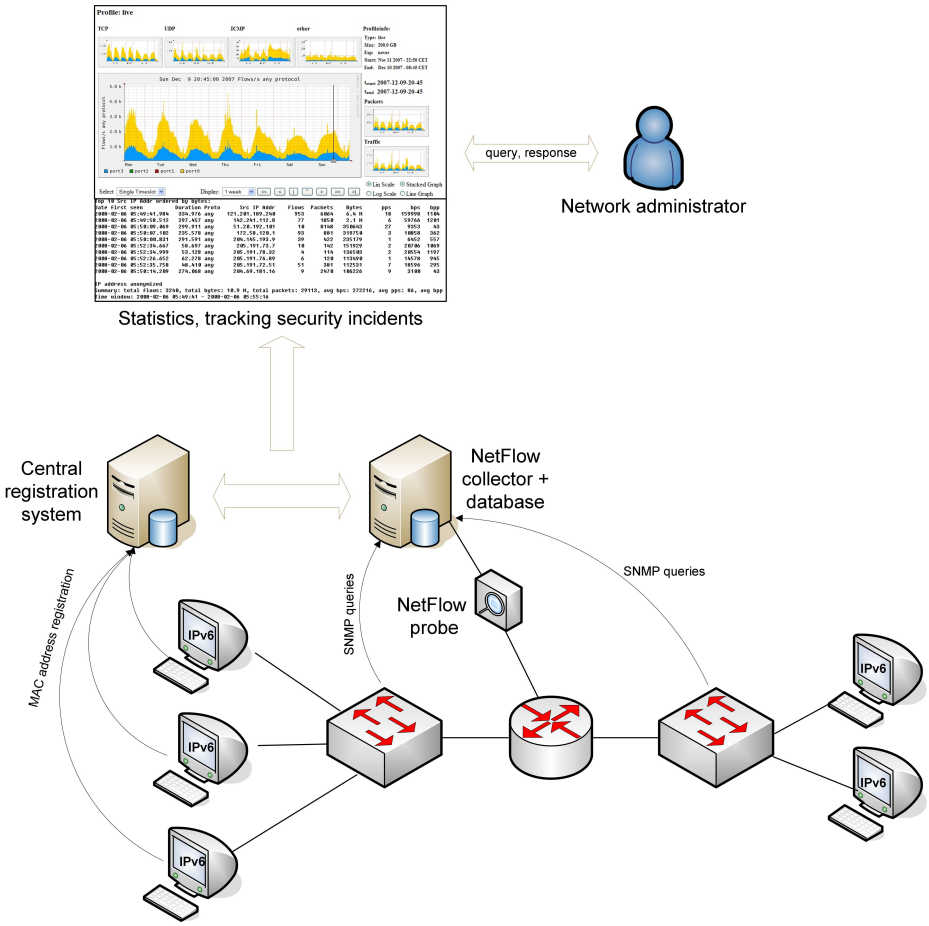

Fig. 5. The Central Monitoring System for IPv4 and IPv6 at BUT

NetFlow protocol transfers these records about passing IP flows from probes to a collector. The NetFlow records contain flow ID and statistics. The flow ID consists of source and destination IP addresses, ports and protocol. The statistics include packet count, byte count, timestamps of first and last packets and others.

The large amount of traffic on current networks turns into a large number of NetFlow records transferred to the collector. Approximate volumes of received

NetFlow data are around 300 MB per hour for a loaded 100 Mbps network and 600 MB per hour for a 1 Gbps moderately utilized network. But it always depends on the specific composition and type of traffic. In any case, the crucial part of the collector is its storage and in particular, its capabilities such as capacity, write and read speeds. Since most of the collectors run on the similar hardware (commodity PC with extended storage capacity) the capabilities varies by utilized underlying storage format – either database or raw files.

A database server provides a comfort of SQL interface and automatic data indexing. But this comes with a prize. Database systems have not been optimized for NetFlow data processing. The NetFlow data require fast storage, fast retrieval of large amount of data but no update of the data at all. Therefore the indexing structures of the database might not fit these requirements. The collectors using proprietary raw files to store NetFlow data must process the queries at the application level which limits flexibility of the queries. Also there is no indexing involved hence the retrieval of selected data may take longer than in case of DB systems.

Currently, the widely-spread collector called nfcapd is based on storage of records into raw files. The nfdump tool is then used to access and filter these flowrecords. We briefly compared its performance with some well known DB systems such as SQLite, MySQL, CouchDB. The nfcapd outperforms these systems when receiving flow records quite dramatically while the time to answer queries on the stored data do not differ that much.

We are aware of possible DB optimization that might improve the performance of some DB systems. But it would require a significant effort with uncertain outcome. Therefore we select current nfcapd and nfdump tool to serve as NetFlow collector. As mentioned previously, such collectors are less flexible than DB collectors. Therefore some implementation effort is necessary to modify nfdump in order to suit data retention requirements. In particular, to extend NetFlow data with data collected by NAV (IP-MAC, MAC-user assignments).

3.2 Data store management function

The data store management function constitutes of some background processes which handle collected data and provides interface for Administrative function.

As mentioned before, we use nfcapd to capture NetFlow data which are generated by probes in our network, and nfdump tool to process and access this data. The goal is to extend this data with source and destination MAC addresses and source and destination user-IDs. We store the corresponding usernames-user-ID pairs in a separate file for later reference. As a result it is possible track activities of a particular machine or user. This requires to modify nfdump file format and patch nfdump itself to understand the modified format. It also requires a tool that would allow to extract data of interest from NAV DB running in parallel and continuously update stored NetFlow data in nfdump files.

The main issue with adding user-IDs to nfdump files is that there is no defined field for this IDs. There are two possibilities how to solve this issue. Either to store these values into other unused fields that are already available, such as

MPLS 1 and MPLS 2 fields, or to change the structure by extending the existing format with new fields. We implemented both options and each has its pros and cons. While the former MPLS solution is backward-compatible with nfsen (GUI for nfdump) since it does not require any patch of nfdump the latter solution does not allow a nfsen to display user-ID unless patched to understand it.

Figure 6 shows the print out of nfdump when MPLS 1 and 2 are used to store user-ID. A typical point of NetFlow observation is an Internet gateway. Therefore it is possible to observe only communication of a local user and a host which is outside of the administered network. As a result one of the MPLS tags is zero which means no association with user identity.

```
[root@coyote 2011-10-21]# nfdump -R . -a -A mpls1,mpls2
Date flow start          Duration   MPLS lbl 1   MPLS lbl 2    Packets    Bytes      bps    Bpp Flows
2011-10-21 10:00:43.975  8410.237        0-0-0     2372-0-0       5649    4.4 M     4166    775   475
2011-10-21 11:31:21.783 16010.021        0-0-0     9788-0-0       2393    1.0 M      505    422   297
2011-10-21 08:03:28.914 24896.951        0-0-0     9134-0-0        467  419519      134    898    52
2011-10-21 09:01:56.269 16728.820        0-0-0     5634-0-0        360  348797      166    968    38
2011-10-21 09:16:00.810  8617.751        0-0-0     1289-0-0        742  839089      778   1130    24
2011-10-21 08:02:02.592 28123.507     1302-0-0        0-0-0         85   13363        3    157     8
2011-10-21 14:03:25.947  2981.902        0-0-0     1423-0-0        191  144824      388    758    15
2011-10-21 09:06:17.782 21162.140        0-0-0     5276-0-0       2994    1.4 M     546    482  1537
2011-10-21 10:02:06.060  6876.676        0-0-0     1257-0-0        149   43776       50    293    32
2011-10-21 12:16:38.796     0.000        0-0-0     2171-0-0          1      52        0     52     1
2011-10-21 14:31:31.272     0.489        0-0-0     1930-0-0         38   30685   502004    807     2
2011-10-21 07:00:16.888 32313.369        0-0-0     3524-0-0        148   54385       13    367    36
2011-10-21 08:42:24.512 14792.651     2949-0-0        0-0-0       7740    1.1 M     606    144   573
2011-10-21 09:12:05.202  9878.254     1241-0-0        0-0-0        830  120975       97    145   136
2011-10-21 00:07:36.050 55639.835     1051-0-0        0-0-0        750   57142        8     76    22
2011-10-21 09:54:39.713  5982.408     1291-0-0        0-0-0         78   16257       21    208     7
2011-10-21 10:55:11.267     6.070     1993-0-0        0-0-0         30    4886     6439    162     2
2011-10-21 09:13:54.536  1398.331        0-0-0     4875-0-0        111  108655      621    978    16
2011-10-21 07:32:17.780  7473.156     3466-0-0        0-0-0       1434  332716      356    232   219
2011-10-21 09:47:31.848     2.958        0-0-0     9142-0-0         11    6937    18761    630     1
2011-10-21 13:40:36.864     1.125      923-0-0        0-0-0         16    1749    12437    109     1
2011-10-21 08:43:42.374     5.221        0-0-0     5349-0-0         17    7290    11170    428     1
2011-10-21 08:08:55.302 32639.955        0-0-0     1100-0-0     245400  319.6 M   78322   1302   908
2011-10-21 09:11:46.579  5695.752        0-0-0     1256-0-0        296   94748      133    320    42
2011-10-21 10:08:51.378   507.453        0-0-0     3665-0-0        198  187480     2955    946    11
2011-10-21 07:20:44.878 16100.667     1081-0-0        0-0-0      20036    2.6 M    1291    129  1371
2011-10-21 12:52:50.178  6651.431        0-0-0     2823-0-0          8    2638        3    329     2
2011-10-21 11:32:30.516  8365.080        0-0-0     6815-0-0         52   19038       18    366    11
2011-10-21 07:59:52.563  4894.640        0-0-0     9122-0-0        389  308403      504    792    30
2011-10-21 08:46:21.720 16021.339        0-0-0     5609-0-0        915    1.3 M     645   1412     9
2011-10-21 07:39:28.750   501.202        0-0-0     6154-0-0         40   11605      185    290     3
2011-10-21 12:34:39.450  2302.626        0-0-0     2991-0-0          6     753        2    125     3
2011-10-21 08:53:58.041 14616.742        0-0-0     5610-0-0       1453  921018      504    633   263
2011-10-21 12:59:49.184  9040.173        0-0-0     9166-0-0        531  477424      422    899    34
2011-10-21 20:19:58.516   286.164     3079-0-0        0-0-0         85   11825      330    139     9
2011-10-21 08:00:11.293  7744.506        0-0-0     1528-0-0        110   68227       70    620    18
2011-10-21 07:38:53.586    22.598        0-0-0     6194-0-0         17    5162     1827    303     2
2011-10-21 06:36:42.066 36235.024        0-0-0     1107-0-0     210827  299.1 M   66029   1418   492
2011-10-21 13:14:58.687     0.419     2930-0-0        0-0-0          2     116     2214     58     1
```

Fig. 6. User-ID stored in MPLS labels (nfdump print out)

It is important to note that all the identification data are not available at the time of arrival of NetFlow data. For instance, NetFlow data are available when they are sent to the collector. However, there is no information available in the NAV database about IP, MAC and user yet. Such information is downloaded later from the switch's ND cache. The same holds for Radius data but RADIUS data are not available for every user - only for those who are connected using

802.1x authentication. For other users, only the IPv6 address and the switch port number and MAC address are used for identification.

An additional tool (nftool), a helper script and helper DB are used to extract (MAC, user-IDs) and (MAC, IP) pairs from NAV DB and store them into nfdump files. The architecture is depicted in the Figure 7.

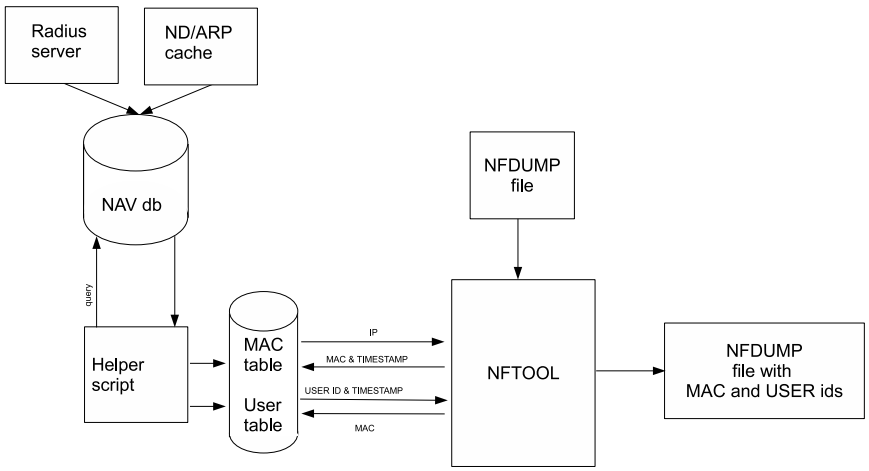

Fig. 7. Framework for fusion of NetFlow and user identification

The helper script is written in Perl and runs every hour (as a cron job) to import data regarding the user-ID, MAC, IP addresses into helper DB (Berkeley DB). This is done to alleviate NAV DB from intensive queries which are caused by nftool when it figures out which MAC address and user-ID belongs to each flow record in the processed nfdump file. The script assembles two helper DB tables (files), user-ID table and MAC table. The user-ID table consists of user-ID, MAC address, start timestamp and end timestamp. The MAC table consists of MAC address, IP address, start timestamp and end timestamp. The timestamps determine the interval during which the assignment is valid.

The nftool is written in C due to a large amount of data and operations it must execute to match nfdump flow records with their corresponding user-IDs and MAC addresses. The nftool runs every hour after the helper script finishes importing data into helper DB. The nftool reads all nfdump files stored in previous hour and parses record by record. It tries to associate each record with its source and destination MAC address based on a match of IP addresses in MAC table. Based on the associated MAC address, the nftool tries to assign a corresponding user-ID from User table.

The deployment in a production network revealed several issues. The associated start timestamps in MAC table may be up to 15 minutes late in relation to the real appearance of the assignment in the network. This delay is caused by NAV setup which by default scans neighbor caches every fifteen minutes. The address may appear right after the scan of the cache and stays unregistered till the next scan. Therefore the nftool considers a each MAC record valid 15 minutes prior to its timestamp. There are also records in which the end timestamps are missing. This might be due to records that has not yet expired from the neighbor cache or due to events such as switch reboot, etc. There might also be situations when one IP address matches multiple MAC records. It is up to nftool to handle these corner cases correctly, e.g., to match the most probable MAC record based on timestamps.

The time dependency of the gathering of different data is crucial when accessing the ND Cache. This temporary memory at the router stores information needed to build the link between the IPv6 address and the MAC address. Because IPv6 addresses change in time and have limited validity, if the ND entry is lost, there is no way to link the IPv6 address and the user/host. To ensure that all information is stored properly in the monitoring system, the SNMP polling interval has to be shorter than the expiration timeout of the ND Cache. Otherwise, some entries in the ND Cache could expire without being downloaded into the central system. Typical timeouts for collecting SNMP and RADIUS data are fifteen minutes. The ND Cache expiration timeout is usually set to more than one hour.

3.3 Administrative function

The main task of Administrative function is to implement handover interface of retained data (RDHI). The ETSI specifies a model for RDHI. It constitutes of several layers, each providing specific functionality.

The message flow layer deals with communication establishment and control. It defines two operational modes: General and Authorized-Organization-initiated. In the case of General mode, both entities are able of full-two way transport of messages whereas in the case of Authorized-Organization-initiated situation, AO must query CSP every time AO wants to receive any data. Next layer specifies contents of each message that is what information must be included in transferred data (identifiers, timestamps, etc.). The encoding and delivery layer defines techniques to handle transparent data exchange. It defines several options such as Direct TCP data exchange, or exchange over HTTP.

In order to hand over retained data collected by NAV and nfdump, we choose to implement a single HTTP client/server communication operated in a Authorized-Organization-initiated mode, i.e., CSP runs an HTTP server and web interface serves to answer AO requests. The administration function is written in PHP and is interpreted by the server. The script receives a request via a POST parameters filled out in the web form. The query typically contains constraints on time interval, IP address, username or MAC in question. Based on the constraints, a nfdump query is constructed and executed. The processing

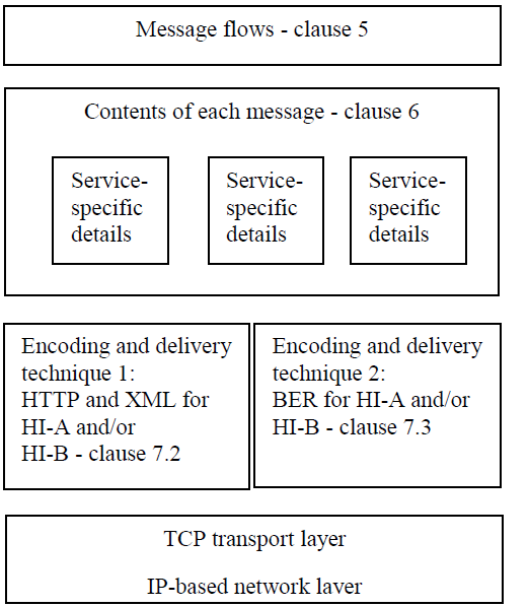

Fig. 8. RDHI model (taken from [2])

of retained data may take a while since nfdump must filter out records that do not match the constraints. When the result is returned the script assembles a web page which is sent to an AO. We recommend to use HTTPS in order to assure secure delivery as well as authenticity.

Please note, that the above described handover interface does not strictly conform to ETSI standard yet. The message flow such as request-acknowledge must be embedded into application level of communication as it is not sufficient to return HTTP status codes as control messages. Further, multiple parallel requests generated in a single session must be addressed.

4 Practical experience

4.1 Network @ Brno University of Technology

This chapter describes how the issues of IPv4 and IPv6 monitoring discussed above are solved at the BUT campus network. The BUT campus network includes 134 active routing devices on the backbone and thousands of connected users (especially students). The chapter presents the data and data sources required for monitoring and how they are obtained. Some results and statistics about IPv4 and IPv6 traffic are given at the end of this chapter.

The campus network at Brno University of Technology (BUT) was built up as a result of cooperation among other universities placed in Brno and Czech

Academy. The campus network connects together several institutions (University faculties, research Institutions, Czech Academy, high schools) placed on over 20 locations in different parts of Brno. Each location is connected at least with two optical cables from two independent directions to achieve maximum reliability of the network. The total length of the optical cables is over 100 km. The core architecture is depicted on Figure 9.

The core of the network is based on 10 Gbps Ethernet technology using HP ProCurve and Extreme Networks devices. OSPF and OSPFv3 routing protocols are used as the interior routing protocol. External connection to the National Research and Education Network (NREN) that is run by CESNET is provided over two 10 Gbps lines with BGP and BGP+ routing. The topology of the core of the network is shown in the Figure 9. From the user perspective, the BUT university campus network connects more than 2,500 staff users and more than 23,000 students. The top utilization is at student dormitories where more than 6,000 students are connected via 100 Mb/s and 1 Gbps links. The IPv6 campus connectivity is implemented according to the Internet Transition Plan [1]. Most of the parts of the university already provide native IPv6 connectivity and significant part of devices connected to the campus network can fully use IPv6.

4.2 Practical Configuration of IPv6 at BUT

At the university environment, the best practice is to identify hosts based on the hosts' IPv4 addresses. Usually, it is done by central system for user registration with the users' MAC addresses. The user registers his MAC address in the system. The MAC address is then used in the DHCP configuration to assign a corresponding IPv4 address. Registered MAC addresses, together with system logs of DHCPv4 servers and data from RADIUS servers, are sufficient to uniquely identify the user, based on the IPv4 address. For a long-term history, NetFlow data are gathered using special NetFlow probes, working on 10 Gbps links. DHCP logs, RADIUS logs and NetFlow records are stored at the central monitoring system, where the users' activity can be looked up as required by the Data Retention Act.

User monitoring of IPv6 traffic is more complicated. The IPv6 address is no longer a unique identifier, as in the case of an IPv4 address. This is mainly because of temporary addresses, as described above. There are two ways to assign IPv6 addresses. Practical experience at BUT indicates that stateful configuration using DHCPv6 does not work properly, so only stateless configuration can be deployed.

One of the basic problems is address assignment to the client systems. The mixture of various OSs requires a solution of automatic address assignment that is supported by most systems. The stateful autoconfiguration using DHCPv6 is very difficult to use today because the lack of support in Windows XP which is still very widespread OS, and older version of MAC OS. DHCPv6 does not support all configuration options (e.g. option for default route), so the stateless autoconfiguration (SLAAC) [3] has to be used as well. Unfortunately, the stateless autoconfiguration in some operating systems turns on privacy extensions.

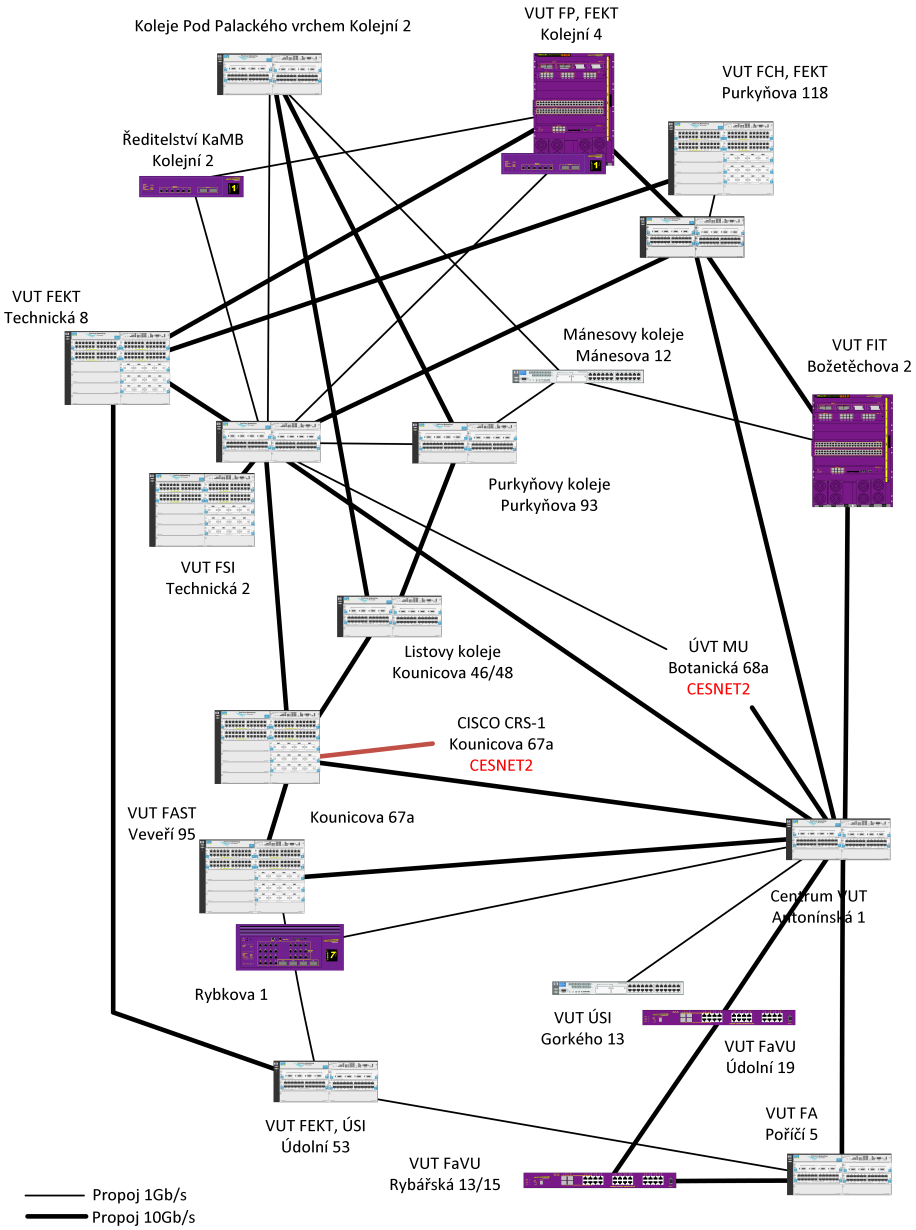

Fig. 9. Topology of VUT network

As a result, the devices use a random end user identifier (EUI) named Temporary IPv6 Addresses. This is a brand new IPv6 feature that allows a node to automatically generate a random IPv6 address on its own. However, this feature contradicts the need to identify a malevolent user. Private, temporary addresses hinder the unique identification of users/hosts connecting to a service. This affects logging and prevents administrators from effectively tracking which users are accessing IPv6 services. Many internal resources require the ability to track the end user's use of services.

IPv6 auto configuration options also increase complexity. There are two fundamentally different mechanisms and protocols where one cannot fully work without the other. Configuration of Recursive DNS servers is nowadays not possible using SLAAC and with DHCPv6 it is not possible to configure the default gateway address (default route). As a result, the only working method is to use both protocols simultaneously. Failure of either mechanism whether through faulty configuration, bugged software or targeted attack, leads to denial of IPv6 connection to the user. Moreover diagnostics are fairly complicated and it requires good knowledge of both mechanisms.

There are two scenarios used for assigning IPv6 addresses. Both Stateless IPv6 configuration and Stateful IPv6 configuration is used.

4.3 Installed DR components

In order to feed our data retention system with information about IP and MAC addresses of connected users we collect ARP tables and FDB tables from all routing switches located in different buildings of BUT campus. Each of these switches serves as a gateway for a given building. We setup NAV to poll these switches regularly every 15 minutes and if a change occurs it is logged in the NAV database. The NAV system runs on a dedicated machine.

The basic NetFlow data are obtained from three monitoring probes that had been installed in the different part of the campus network. Two of them on the upstream lines to collect complete data exchanged between the campus network and rest of the Internet. The third probe was installed at the student's dormitory where most traffic of the network is concentrated. In cooperation with the company INVEA-TECH the probes were ported to HP Procurve ONE service module. That allowed to process data directly from the backplain on the switch. Data obtained from the probes are collected on the single NetFlow collector. The collector pulls out data from NAV machine and merge these data with NetFlow.

4.4 Obtained results

Many interesting statistics were obtained as the side result of implementation of data retention system. The Figure 10 displays visibility of a single machine under various IP addresses. The first IP address starting with *fe80* is link local address which remains constant for the whole period of observation. The same

holds for IPv4 address. The machine has multiple self-generated IPv6 addresses which are used in ad-hoc mode.

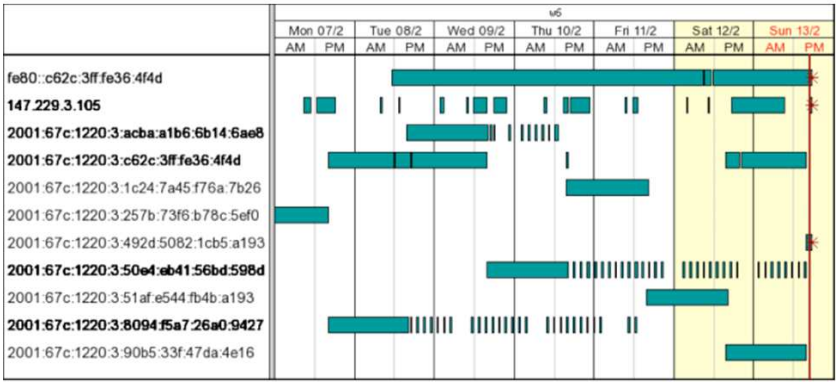

Fig. 10. Visibility of a computer under various IPv6 addresses

Since the deployment of the system we have been able to observe differences between IPv4 and IPv6. In total, we have observed 41032 unique MAC addresses. There have been 18480 MAC addresses which have been visible under any IPv6 address. Nearly all these MAC addresses (except 100) have been associated with link local IPv6 address. There have been 26277 unique IPv6 link local addresses. But more importantly there have been 13733 MAC addresses with nearly a million of global IPv6 addresses. This means that on average each IPv6 capable machine changed its global IPv6 address more than 60 times. On the other hand, we have seen only 43786 unique IPv4 addresses in total. The observation period has been approximately 10 months.

We have also observed evolution of the traffic during a shorter period (a week, gray background marks weekends) with timescale resolution of one hour. Some of the findings are plotted in Figures 11, 12, 13.

The Figure 11 shows the number of hosts with assigned IPv4 or IPv6 address. We consider a host to be uniquely identified by the MAC address. We can see that the number of hosts follows the daily pattern with a significant decrease during Friday till Sunday afternoon when students return to dormitories. The amount of IPv6 hosts is close to the number of IPv4 hosts. In comparison to the total statistics presented above the ratio of IPv6 and IPv4 host has increased significantly. This increase is most likely caused by migration of users to a newer operation systems during this year.

We have also focused on the IPv6-capable hosts that actively utilize IPv6 during communication. The graph on Figure 12 displays the number of internal and external hosts involved in active communication. The term internal host means a host which belongs to the BUT network whereas external host is located

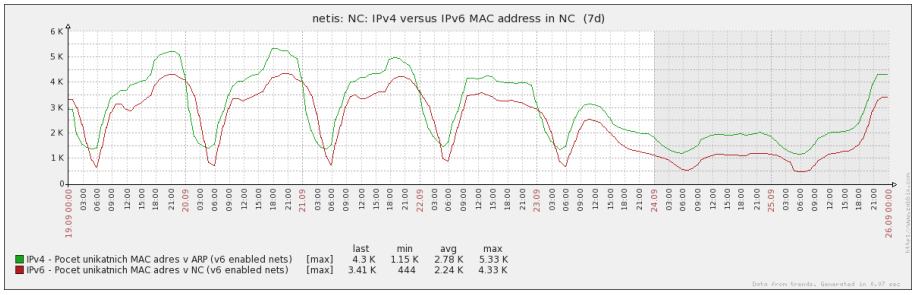

Fig. 11. Number of clients using IPv4 or IPv6

outside of the BUT network. We can observe that the number of internal IPv6 hosts is smaller than the number of external IPv6 hosts, i.e., an internal host communicates on average with more than two external IPv6 hosts.

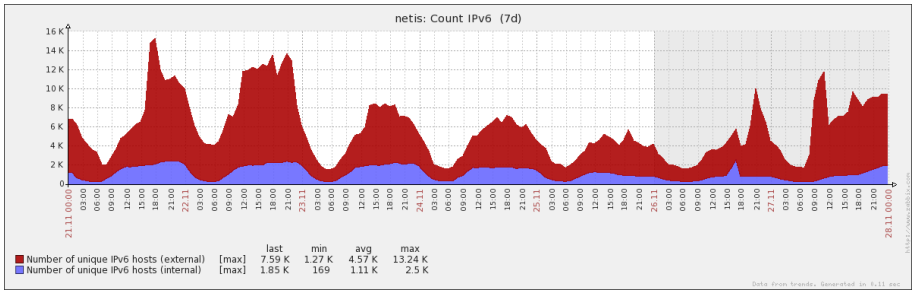

Fig. 12. Number of internal and external IPv6 hosts

Finally, the graph on Figure 13 displays the amount of traffic with respect to the IP protocol used. We introduce a third category which account for the tunneled traffic such as Teredo. The amount of traffic strongly follows the daily and weekly period. The amount of ingress traffic is significantly larger than the amount of egress traffic for both IP protocols. On average, the amount of IPv4 traffic is ten times higher than IPv6 which is ten times higher than the amount of traffic utilizing tunneling mechanisms. The large difference between the amount of IPv4 traffic and IPv6 traffic is in contrast with the small difference of IPv4 and IPv6 capable hosts. Nevertheless this discrepancy is expected as a result of small support of IPv6 by network applications.

Please bare in mind that the presented statistics are valid for a campus network which might be specific due to its users and strong effort to keep up with the IPv6 transition plan. A commercial provider might observe a different statistics. We would expect to see even a larger difference in the amount of IPv4

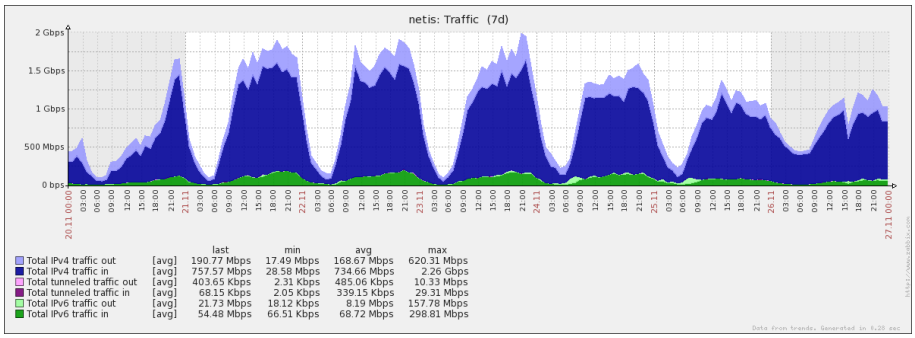

Fig. 13. Breakdown of the traffic mix based on IP protocol

and IPv6 traffic. The main cause could be older OS of users and missing native support of IPv6. In such a case, tunneling mechanisms come into play and there might be tunneled IPv6 or IPv4 traffic only.

5 Conclusions

This technical report was focused on the design and implementation of data retention system in IP network. The stress was given on addressing issues related to the deployment of IPv6 in terms of recovering user identity.

The designed system consists of several monitoring tools. The results of these tools are combined together to obtain data about the past and on-going traffic enriched with the information about a user identity. The system has been successfully deployed in BUT network. Since its deployment it has been used to manage violations of network usage policy and to observe IPv6 network behavior and its trends.

References

1. Curran, J.: *RFC 5211 An Internet Transition Plan*. 07 2008.
   URL http://tools.ietf.org/html/rfc5211
2. European Telecommunications Standards Institute: *ETSI TS 102 657: Lawful Interception (LI);Retained data handling;Handover interface for the request and delivery of retained data*. 12 2009, version 1.4.1.
3. Thomson, S.; et al.: *RFC 4862 IPv6 Stateless Address Autoconfiguration*. 09 2007.
   URL http://tools.ietf.org/html/rfc4862
4. UNINETT and Norwegian University of Science and Technology: *NAV*. 3 2011, version 3.9.
   URL http://metanav.uninett.no