Building an IPv6 Address Management System

Athanasios Douitsis

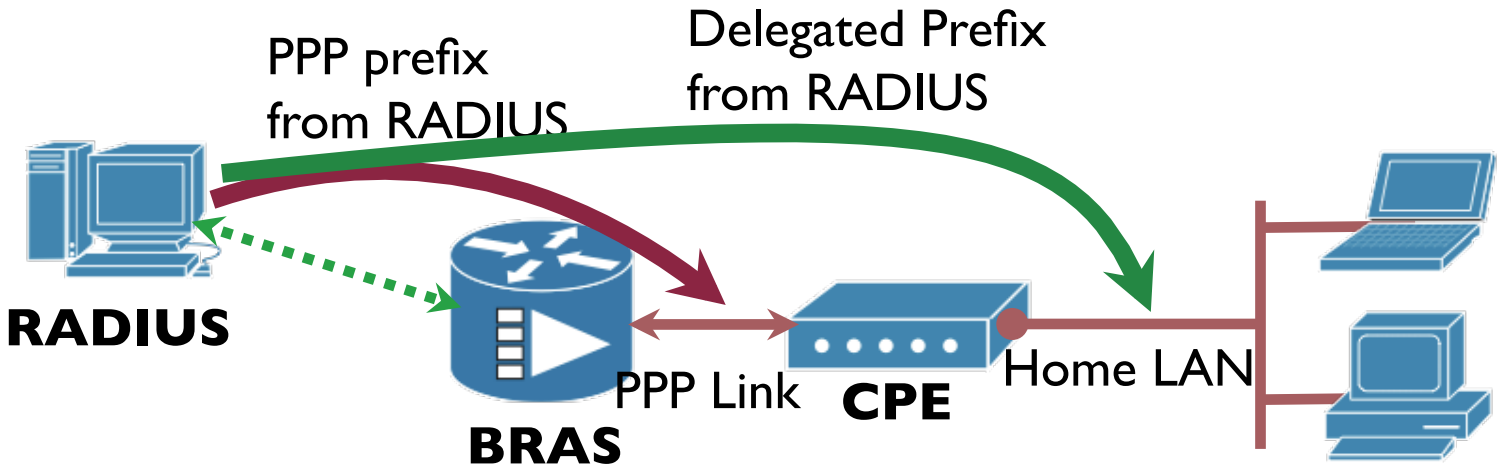
National Technical University of Athens NOC

Outline

- **Background**
  - Full RADIUS-based Prefix Assignment
- **The Greek Student Network (EDUDSL) case**
  - Previous IPv6 setup (IPv4-derived prefix assignment)
  - On-the-fly assignment of static IPv6 prefixes
  - Implementation and performance
- **The Greek School Network (SCH) case**
  - Previous IPv6 setup (manual assignment)
  - Proposed future addressing scheme
  - Static IPv6 assignment method (based on EDUDSL codebase)
- **Conclusion**
  - Best practice: Offset-based storage of IPv6 prefixes
  - Future Ideas

Background: RADIUS-based prefix assignment

- Access network, IPv6 based on SLAAC (PPP) and DHCPv6 PD (Home LAN)
- Assignment of **all** prefixes by the RADIUS server
  - **Framed-IPv6-Prefix, Delegated-IPv6-Prefix**
  - Pro: Most vendor independent solution
  - Con: Complexity in RADIUS server



Case #1

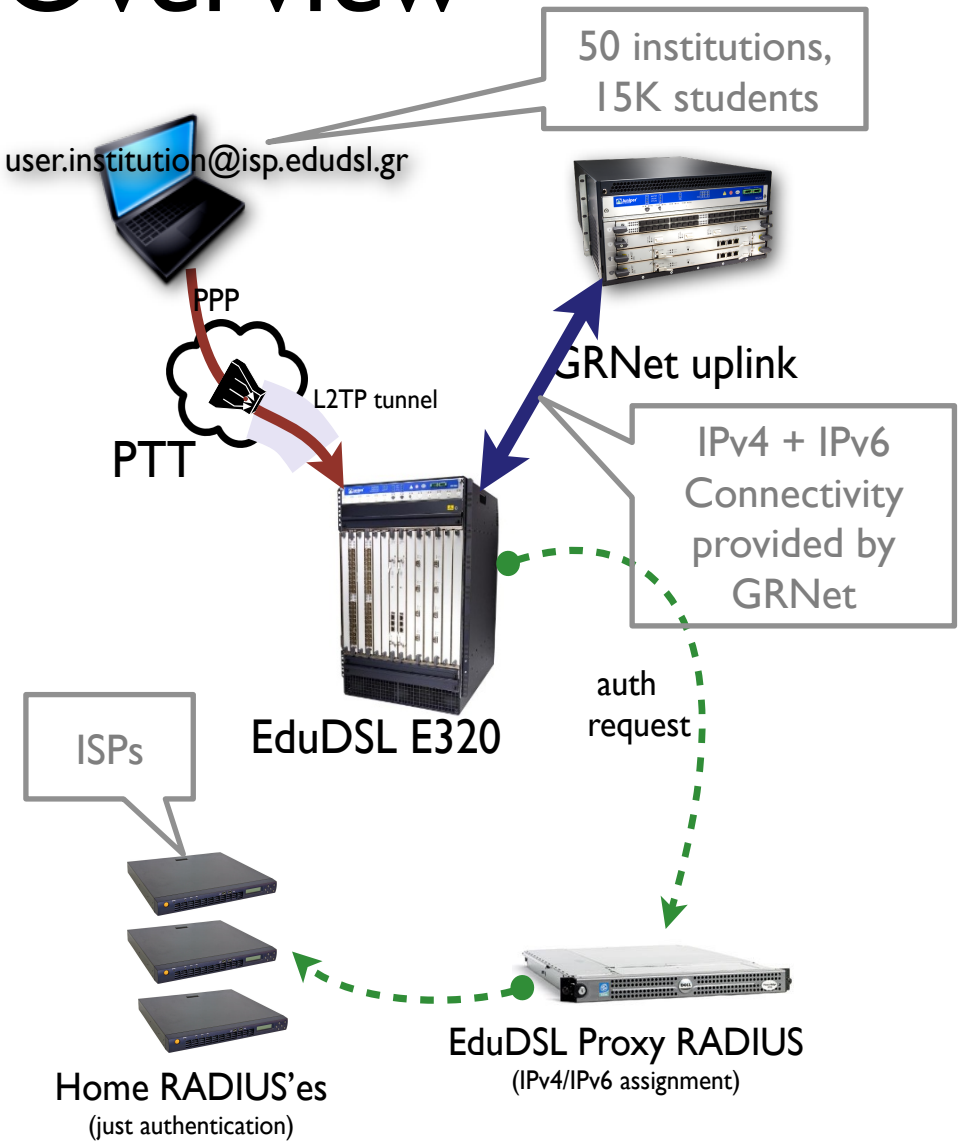
Greek Student Network (EDUDSL)

EDUDSL Overview

- User billing & registration outsourced to ISPs

- EDUDSL Proxy RADIUS:
  - **IPv4 and IPv6 Address assignment**
  - Communication with ISP RADIUS for authentication **only**

- Complications:
  - Account usernames unknown until time of **first login**
  - Deleted accounts unknown, **time of deletion unknown**

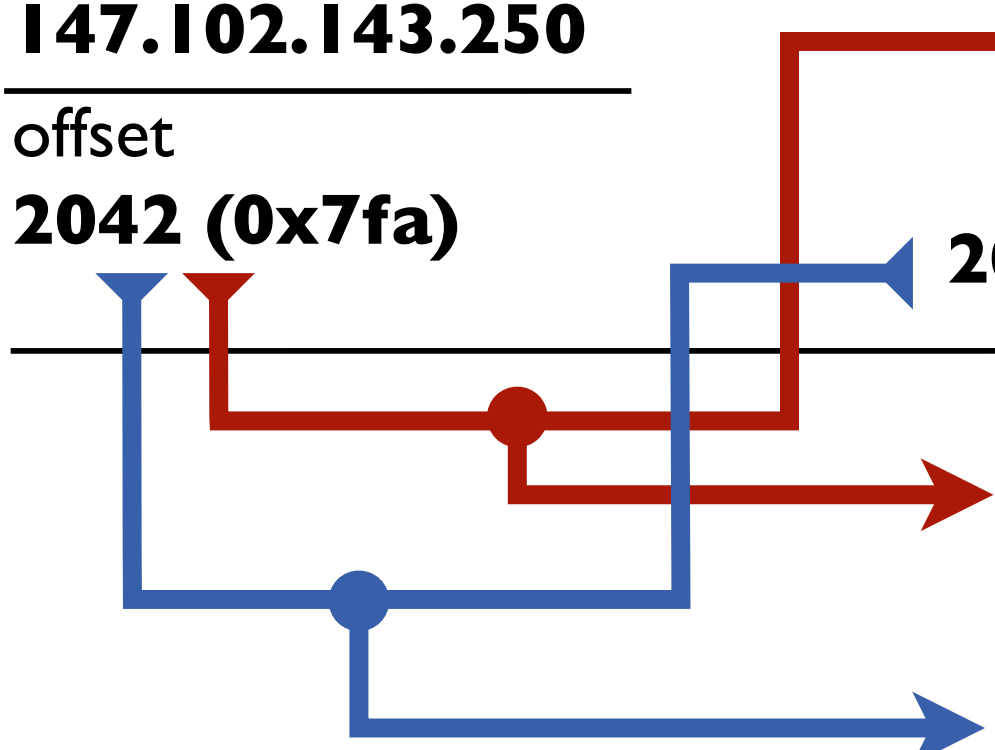


Previous IPv6 assignment method

IPv4 space
147.102.136.0/21

IPv4 user
147.102.143.250

offset
2042 (0x7fa)



Entire pool
2001:648:2001::/48

PPP
2001:648:2001::/49

Delegated
2001:648:2001:8000::/49

Framed-IPv6-Prefix
2001:648:2001:7fa::/64

Delegated-IPv6-Prefix
2001:648:2001:87fa::/64

Goal : Static Prefixes per user

- **Static Framed-IPv6-Prefix, Delegated-IPv6-Prefix**
  - Randomly chosen (not deterministic from username)
  - Assigned Per Username
- **Persistence** across changes, reloads, etc.
- **Recycling of Prefixes**
  - Expiration after user inactivity period (e.g. 5 months)

Static Prefix System Requirements

- **On the fly** IPv6 prefix assignment to newly appearing usernames
- **Same already assigned** IPv6 Prefix in subsequent logins of already-seen username
- **Automatic reuse of inactive** prefixes
  - **Recycling** of **least recently used prefix**
  - **Guaranteed period** e.g. 6 months before recycling
  - **Retention** of prefixes as long as possible
- **Speed:** Requirement for sub-second responses
  - Synchronous to AAA requests
  - **Performance** monitoring
- Support for subscriber groups ➔ different prefix pools
- **Logging** of past prefixes (audit log)

Static Prefix Assignment Approach

- **Elect one (1) unique static integer offset** per user
  - Used to enumerate Framed, Delegated prefixes
  - Example: **Pool size: 8096 → Offset range: 0 - 8095**
- Appearance of **new** username:
  - If unused offset available → creation of a new record with username, offset pair
  - If no free offsets available → finding record of **oldest offset not in use,** replace username
    - Storing of the **old** username, offset pair in the log
- **Existing** username:
  - Simply: Retrieval of offset already stored for username

Prefix Calculation from Offset

- **Storage of address offset** instead of full prefix
  - **Storage in ordinary DB**
  - Easier sorting, easier counting
  - Renumbering possible without alteration of thousands of user records
    - Simple change of pool spaces

Implementation

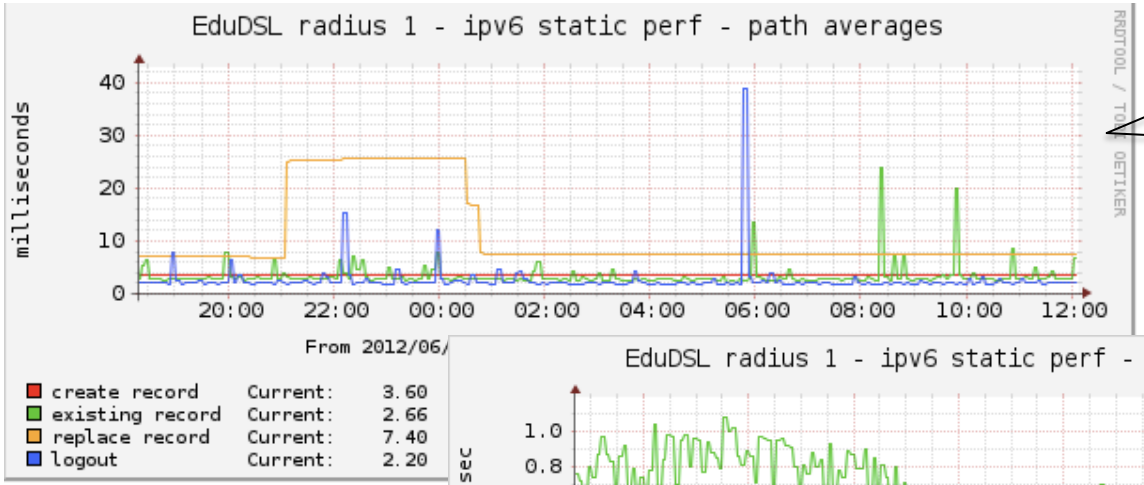
- Perl module
- Integration with FreeRADIUS (rlm_perl)
- MySQL ➔
  - IPv6 Prefix Pools table
  - Static Addresses table (offsets)
  - Log tables (old records log, audit log)

`https://github.com/aduitsis/IPv6-Static`

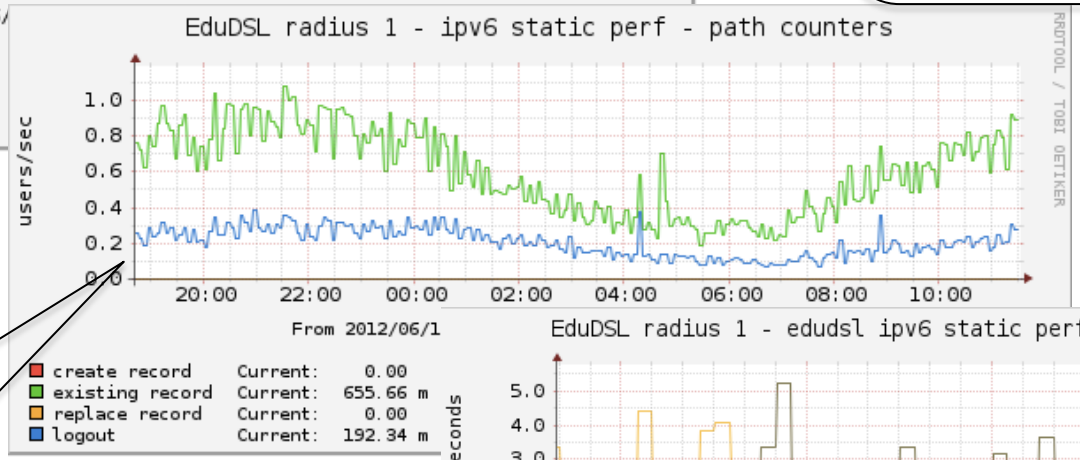
Miscellaneous Features

- Grouping feature (many different groups)
- Keeping track of online users (configurable)
  - Double login detection
- Configurable guaranteed inactive address retention
  - e.g. candidacy for recycling after min. 5 months since last logout
- Multiple RADIUS operation on same DB via table locking

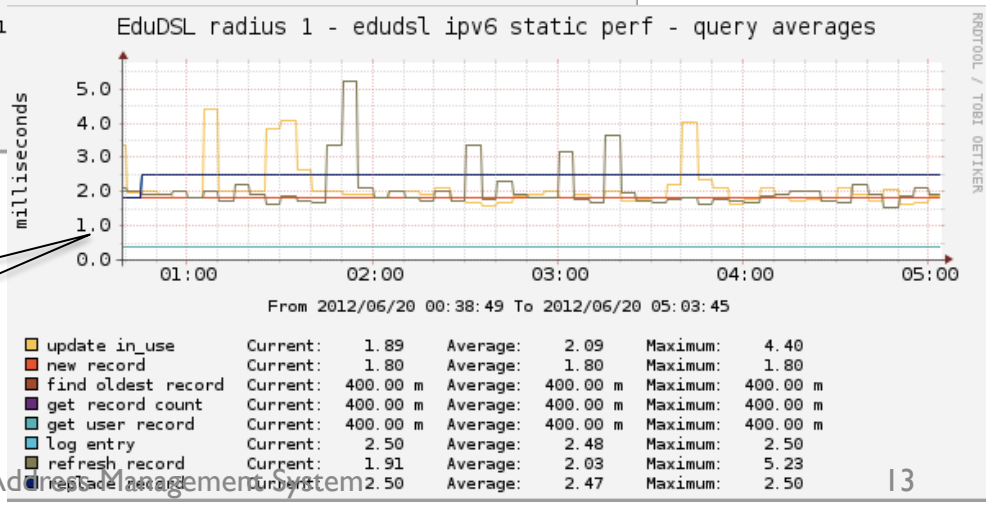
Performance Monitoring



Average duration of each different case (create, existing, replace, logout)



Counters for each different case



Drilldown timers for each different type of SQL query

Operational Experience

- Fairly fast (<50 milliseconds per AAA request)
  - Performance monitoring
- In production for almost 2 years
- Start: 1 Initial master pool – almost everybody
- Today: 2 Pools

Case #2

Greek School Network (SCH)

Greek School Network (**SCH**)

- SCH: Country-wide broadband access network
  - **18000** schools and administrative **units**
  - **Content filtering**
  - Information services (web hosting, email)
- **>10000 CPEs, 6 BRAS's, 2 RADIUS servers, LDAP**

SCH Previous IPv6 Setup

- In place for almost **10 years**
  - Case study in book "Global IPv6 Strategies: From Business Analysis to Operational Planning"
- Same prefix pool for all units
- **/63** per unit
  - /64 for WAN/PPP, /64 for DHCPv6 PD
- **Manual** assignment of prefixes
  - Maintenance by SCH operators
  - Error-prone, cumbersome
- Vendor specific IPv6 RADIUS attributes
  - **stored verbatim** in directory as *radiusReplyItem(s)*

SCH Future IPv6 Requirements

- Design for another **10 years** ahead
- **Static /56** per school → **256** VLANs
  - *plus a static /64 for the PPP/WAN link*
- **Automated** Prefix assignment/maintenance
- Storage of clean IPv6 prefixes in LDAP (*Vendor neutral)*
  - Extension of LDAP schema with dedicated IPv6 attributes
- RADIUS translates to VSAs ***only if necessary***
- Grouping of unit prefixes according to category
  - e.g. **high school, administrative, elementary**
  - Easier policy enforcement, access lists, content filtering
    - very important for **elementary category**

IPv6 Pool Dimensioning

- Assumption of double space requirements in next 10 years
  - Separate prefix group per unit category

2001:648:3400::/40	2001:648:3400::/44	*core network / datacenter*	
	2001:648:3410::/44	*administrative*	4000
	2001:648:3420::/43	*high school units*	8000
	2001:648:3440::/42	*elementary units*	16000
	2001:648:3480::/41		

RADIUS and LDAP modifications

- **Directory service (LDAP)**
  - 2 new attributes
    - **FramedIPv6Prefix**
    - **DelegatedIPv6Prefix**
- **RADIUS**
  - **Framed-IPv6-Prefix** (from LDAP attribute)
  - **Delegated-IPv6-Prefix** (from LDAP attribute)
  - Framed-Interface-ID (TBD: unset, static or random)
  - DNS-Server-IPv6-Address (TBD: static, dynamic)

Software goals

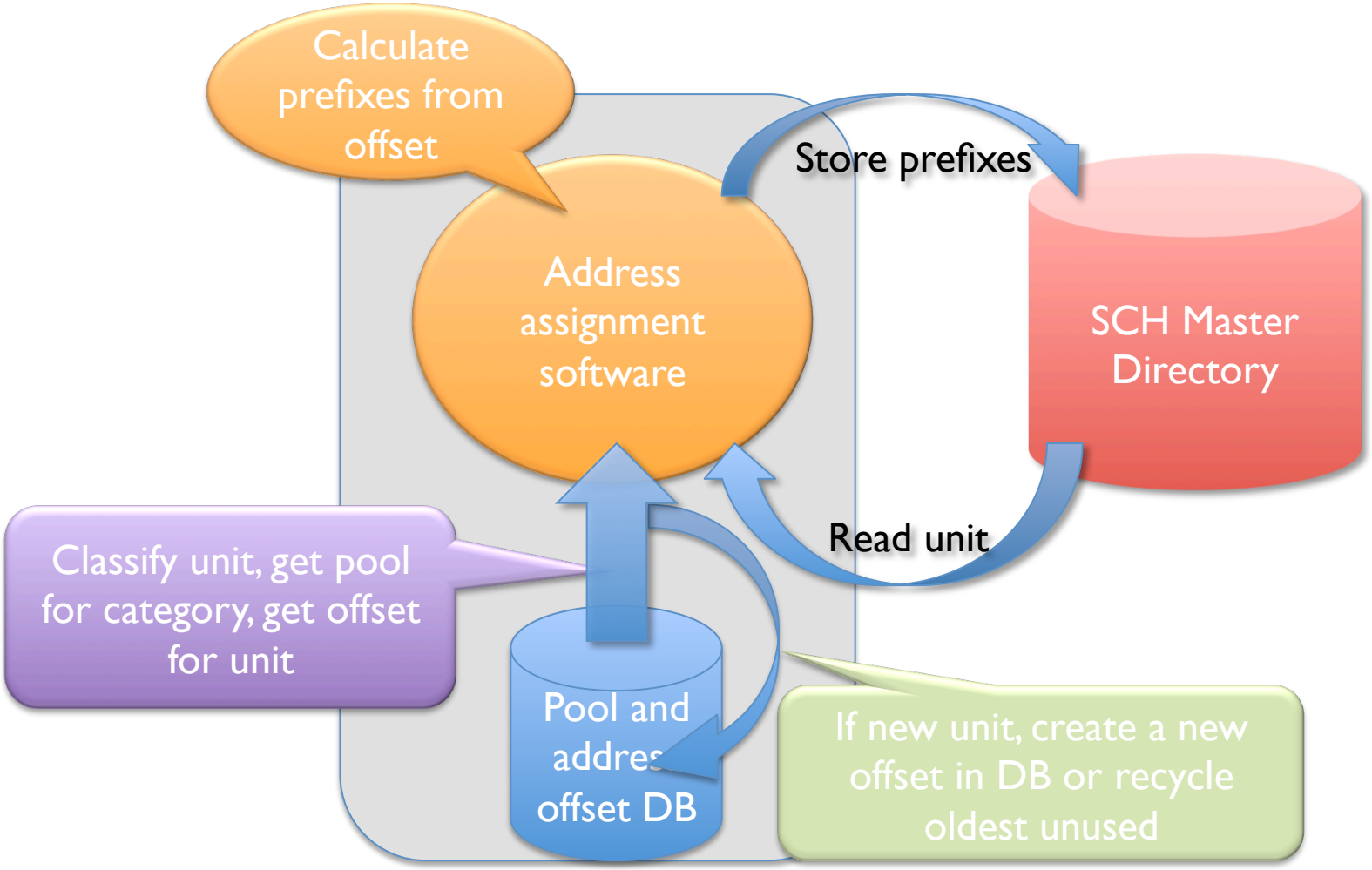
- **Automated** operation
- Batch mode
  - Assign prefix to every unit in LDAP
- Single unit mode
  - Assign prefix to specific unit supplied as argument
- Ability for on the fly **renumbering**
  - In case of IPv6 pools space reconfiguration
- Lifecycle automation (auto detection of creation and deletion of units)

Software requirements

- Update directory entries
- Multiple configurable groups/pools
  - **Different** delegated prefix length per group
- Assignment of framed, delegated prefixes per unit

- **Existing unit** → Retain **same** prefix
- **New unit** → Assignment of **free** prefix
- **Deleted unit** → **Recycle** prefix
  - Deletion / prefix reassignment logging (for audit/ accounting purposes)

System Operation Overview



Software code

- Standalone software
  - Contrast with EDUDSL software integrated into EDUDSL RADIUS
- Perl >= 5.14
- Communication with DB & LDAP
- Approx. 35 CPAN module dependencies
- MySQL 5.x

Conclusion & Future Ideas

Best practices

- **Offsets** instead of full prefixes in DB
  - Indexed appropriately→ **speed**
- Usage of Prefix Pools to group subscribers
- **Primary storage: DB**
  - Copy in LDAP
  - Ability to recreate all prefixes from DB
- Sparse Mapping(?)
- Single username mode equally important as batch mode

Future Directions

- Addition of triggers for external tools (API)
- Possibility: IPv4 enumeration with same offsets

- Code cleanup
- Some features difficult to actually really test
  - Need more rigorous testing
- More documentation

Lastly: Sparse Allocation of Offsets

User Offset	Mapped Offset	User Delegated Prefix
0	0	2001:648:3000::/56
1	4	2001:648:3000:400::/56
2	2	2001:648:3000:200::/56
3	5	2001:648:3000:500::/56
4	1	2001:648:3000:100::/56
5	6	2001:648:3000:600::/56
6	3	2001:648:3000:300::/56
7	7	2001:648:3000:700::/56

Usage of Sparse Allocation (2)

- Described in http://www.ripe.net/ripe/docs/ripe-343#3
- Question: Still useful after extensive offset recycling **?**
  - Excessive recycling causing "**fragmentation**" in the pool
    - Defragmentation maybe possible with external tool

Thank you for your attention!
Questions?

athanasios.douitsis@noc.ntua.gr