TELCO challenge: Learning and managing the network behavior

M.Sc. Ljupco Vangelski
CEO, Scope Innovations

Kiril Oncevski
NOC, ISP Neotel Skopje

Presentation overview

- Challenges for the modern network monitoring
- Traditional approach vs. Machine learning approach
- Modeling the network behavior
- Anomaly detection
- Putting it all together in a system that works
- Visualization

About Scope

- Over the top service company
- Five founders, each with 6+ ICT experience & telco background
- Focused on open source product development and integration
- Particularly focused on monitoring solutions

BEYOND SOLUTIONS

scope

About Neotel

- Established in 2004 as an Investment of Macedonian ICT company - NEOCOM and private Investors
- Offices in Skopje, but network and operations across whole country.
- NEOTEL employees 50+
- Certification ISO 9001 and ISO14001

Private leased circuits

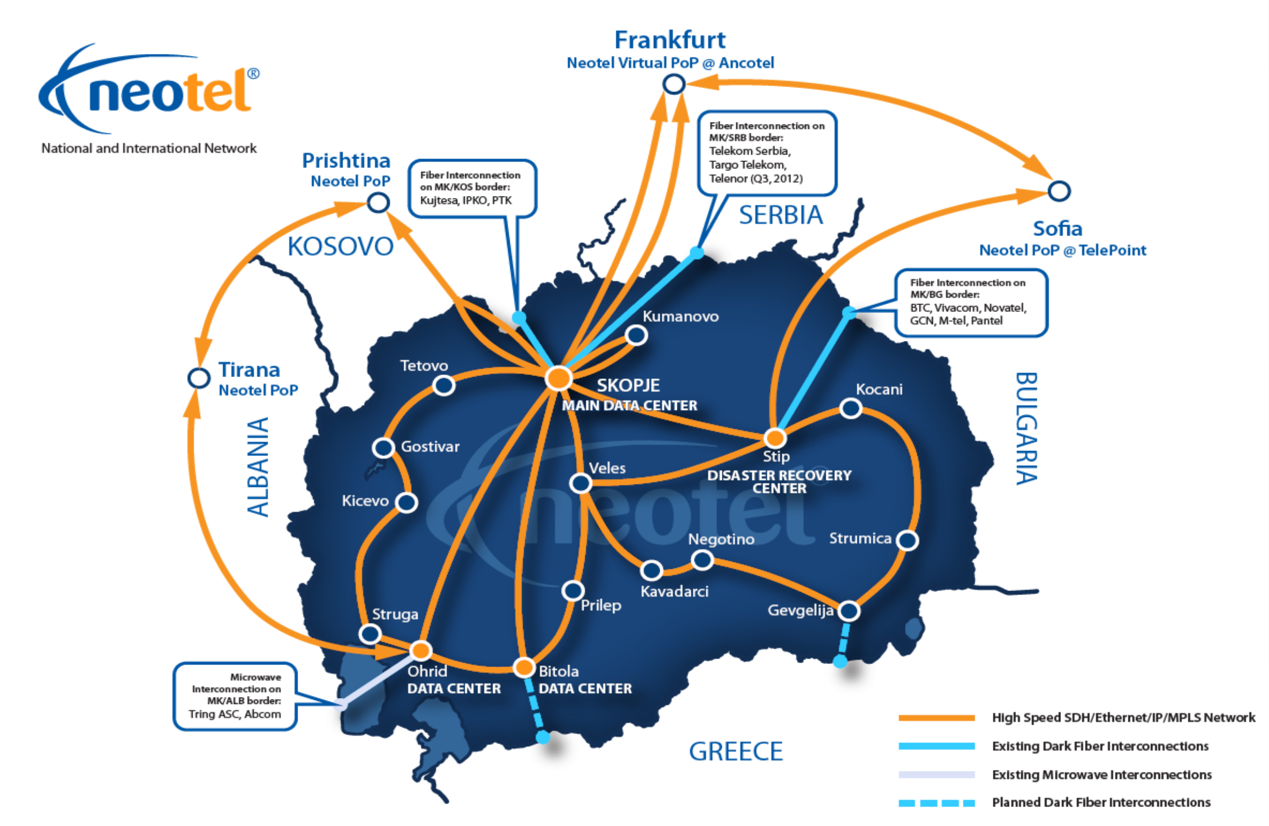
Internet

Voice

Colocation & Hosting

About Neotel

- Rely as much as possible on our own infrastructure:
- Dense Metro Ethernet network in Skopje
- Connection to 3 border crossings (Kosovo, Serbia, Bulgaria)



Fiber Interconnection on MK/KOS border:
Kujtesa, IPKO, PTK

Fiber Interconnection on MK/SRB border:
Telekom Serbia,
Targo Telekom,
Telenor (Q3, 2012)

Fiber Interconnection on MK/BG border:
BTC, Vivacom, Novatel,
GCN, M-tel, Pantel

Microwave Interconnection on MK/ALB border:
Tring ASC, Abcom

Goals of the project

- Investigate and evaluate an approach for modeling the network behavior

- Create better insight and better representation of existing data, for network monitoring

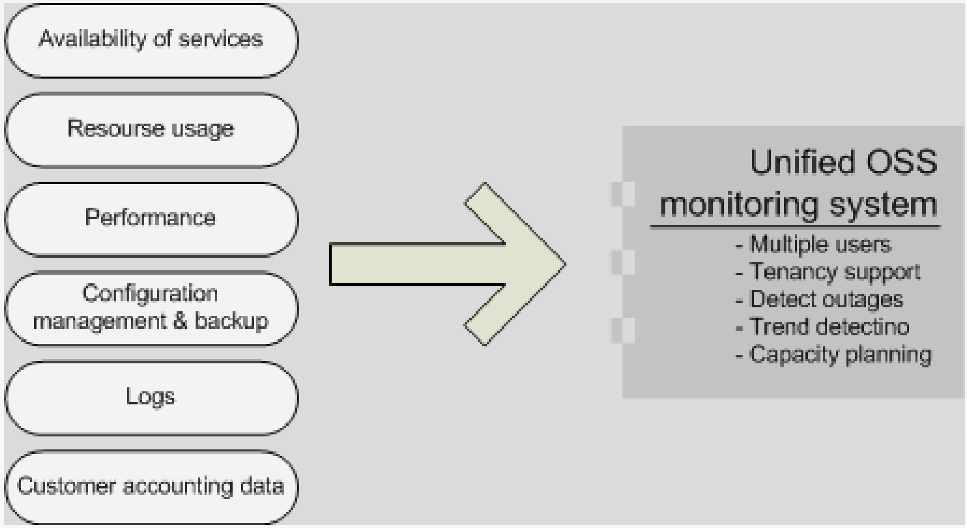
- Integrate a system which can be made available to Neotel's end users, for monitoring their own network activity

Challenges for the modern telcos

- Manage growth
- Consolidation
- Convergence
- Which should be my service domain?
- Make sure QoE and QoS are monitored... and guaranteed
- Change management

Modern network monitoring

- Netflow, sFlow, packet capturing, Syslog, SNMP, Routing data
- What is there to be monitored?



Availability of services

Resourse usage

Performance

Configuration management & backup

Logs

Customer accounting data

Unified OSS monitoring system

- Multiple users
- Tenancy support
- Detect outages
- Trend detectino
- Capacity planning

- Where are we looking for anomalies?
- And this is only the operations (OSS). What about BSS integration? Fraud monitoring and BI ?

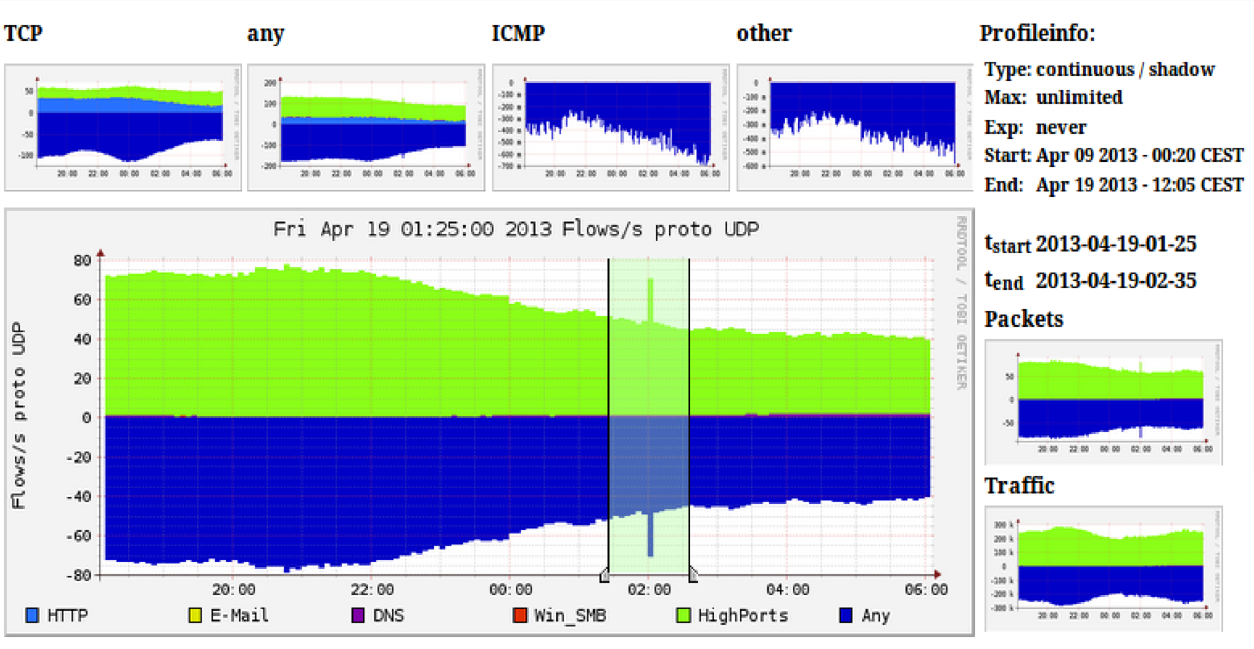
Modern network monitoring

- Detailed network traffic statistics: Netflow
- Yes we all do that... but is it sufficient to describe the behavior of our customers?
- We don't only want to detect outages but changes of behavior which could affect the quality of user experience.
- From netflow to big data... where do ordinary netflow collectors usually fail?

  Fancy graphs, containing not much useful data apart from ports and transport protocol statistics

Traditional approach

- Multiple netflow probes, one collector dashboard
- Monitor packets, flows, bytes, pps, fps, bps, per protocols, TCP/UDP, ports and IP addresses
- Identify top talkers
- Detect DdoS and basic infections



Type: continuous / shadow
Max: unlimited
Exp: never
Start: Apr 09 2013 - 00:20 CEST
End: Apr 19 2013 - 12:05 CEST

t_{start} 2013-04-19-01-25
t_{end} 2013-04-19-02-35

How our system works?

1) Put everything in one box

2) Put it at the Service Provider premises for 15 days

3) Accept netflow and mirror ports

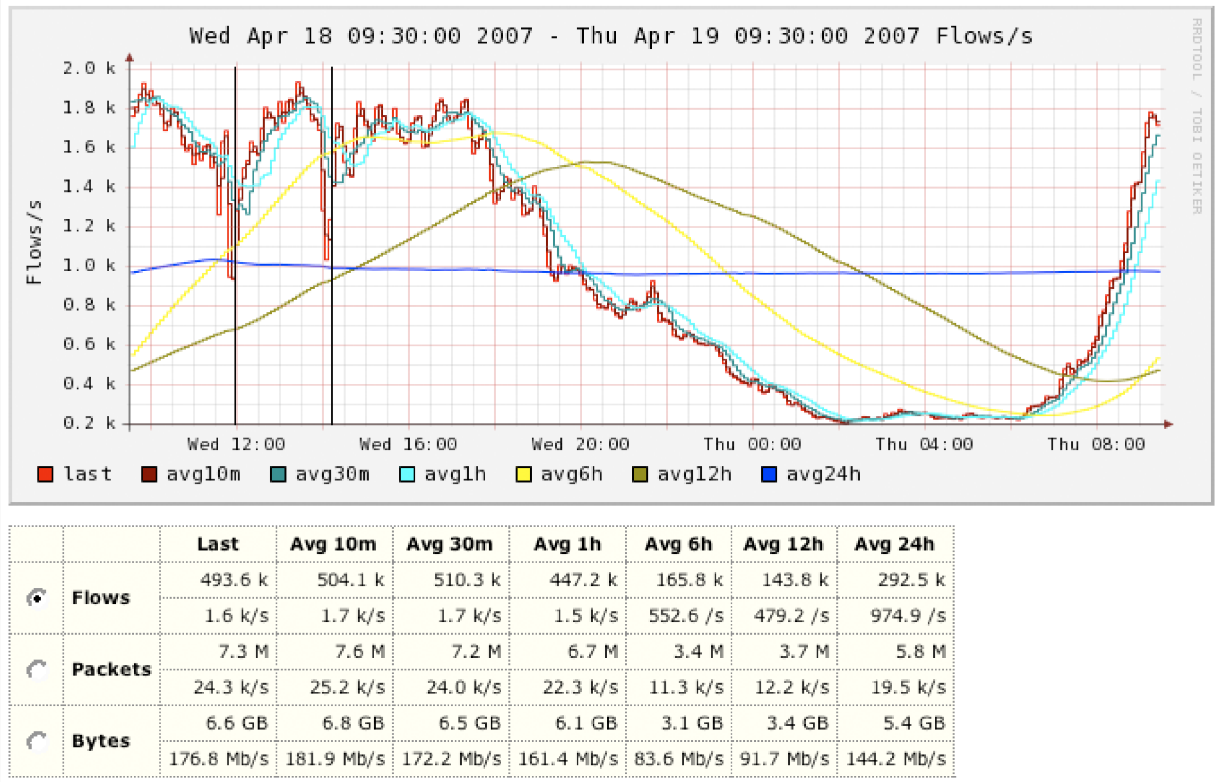
4) Learn the network behavior (Dynamic thresholds and seasonal parameters are created)

5) Integrate everything on a single dashboard with powerful graphing capabilities

6) Deal with alarms, deal with automatic traffic blocking

Modeling the network behavior

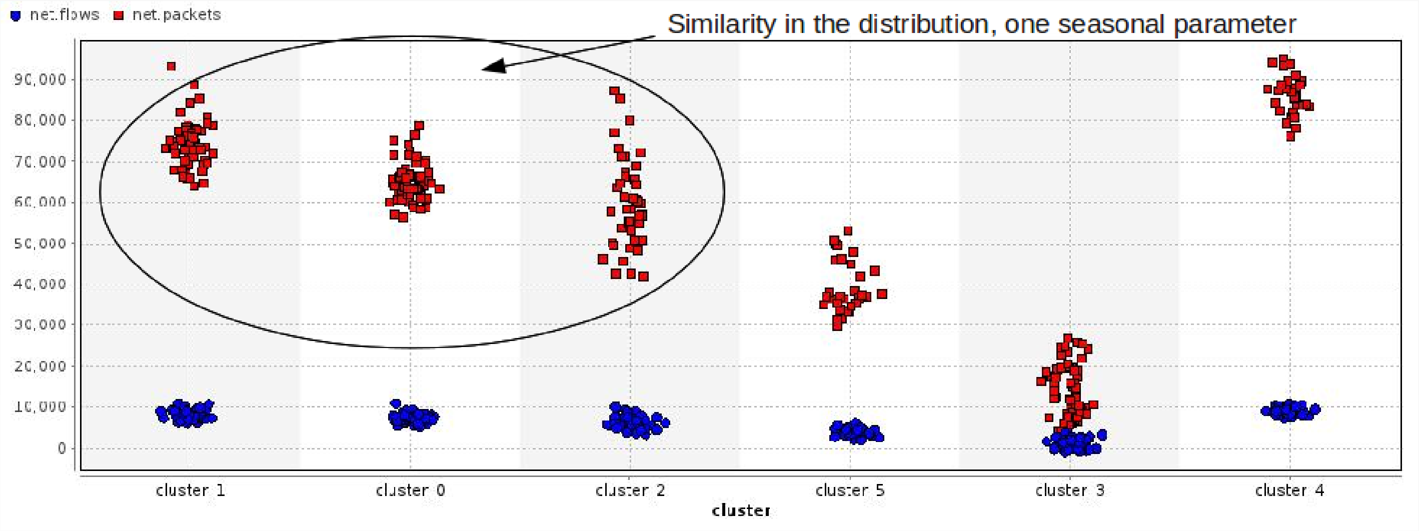
- What are the parameters and attributes?
- Seasonal behavior (weekdays / weekends / peak / off-peak hours)
- Introducing machine learning



		Last	Avg 10m	Avg 30m	Avg 1h	Avg 6h	Avg 12h	Avg 24h
●	Flows	493.6 k	504.1 k	510.3 k	447.2 k	165.8 k	143.8 k	292.5 k
		1.6 k/s	1.7 k/s	1.7 k/s	1.5 k/s	552.6 /s	479.2 /s	974.9 /s
○	Packets	7.3 M	7.6 M	7.2 M	6.7 M	3.4 M	3.7 M	5.8 M
		24.3 k/s	25.2 k/s	24.0 k/s	22.3 k/s	11.3 k/s	12.2 k/s	19.5 k/s
○	Bytes	6.6 GB	6.8 GB	6.5 GB	6.1 GB	3.1 GB	3.4 GB	5.4 GB
		176.8 Mb/s	181.9 Mb/s	172.2 Mb/s	161.4 Mb/s	83.6 Mb/s	91.7 Mb/s	144.2 Mb/s

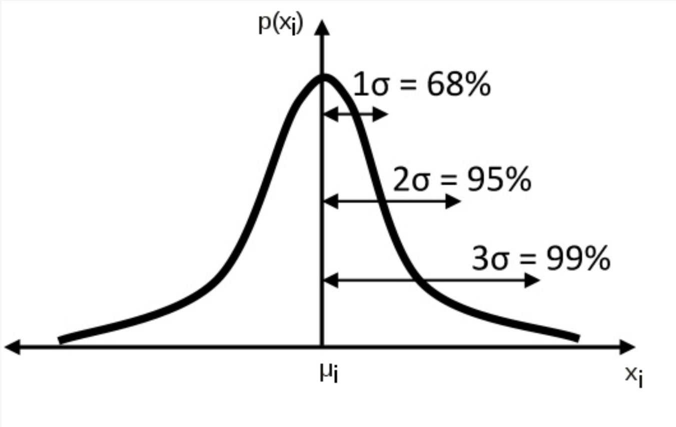
Modeling the network behavior

- Why can't we use supervised machine learning?
- When can we use unsupervised machine learning? We let the system learn itself:
    - Identify seasonal parameters
    - Find clusters and dynamic thresholds



Anomaly detection

- Having the seasonal distinction helps in identifying different anomaly thresholds for different parameters at a different point in time

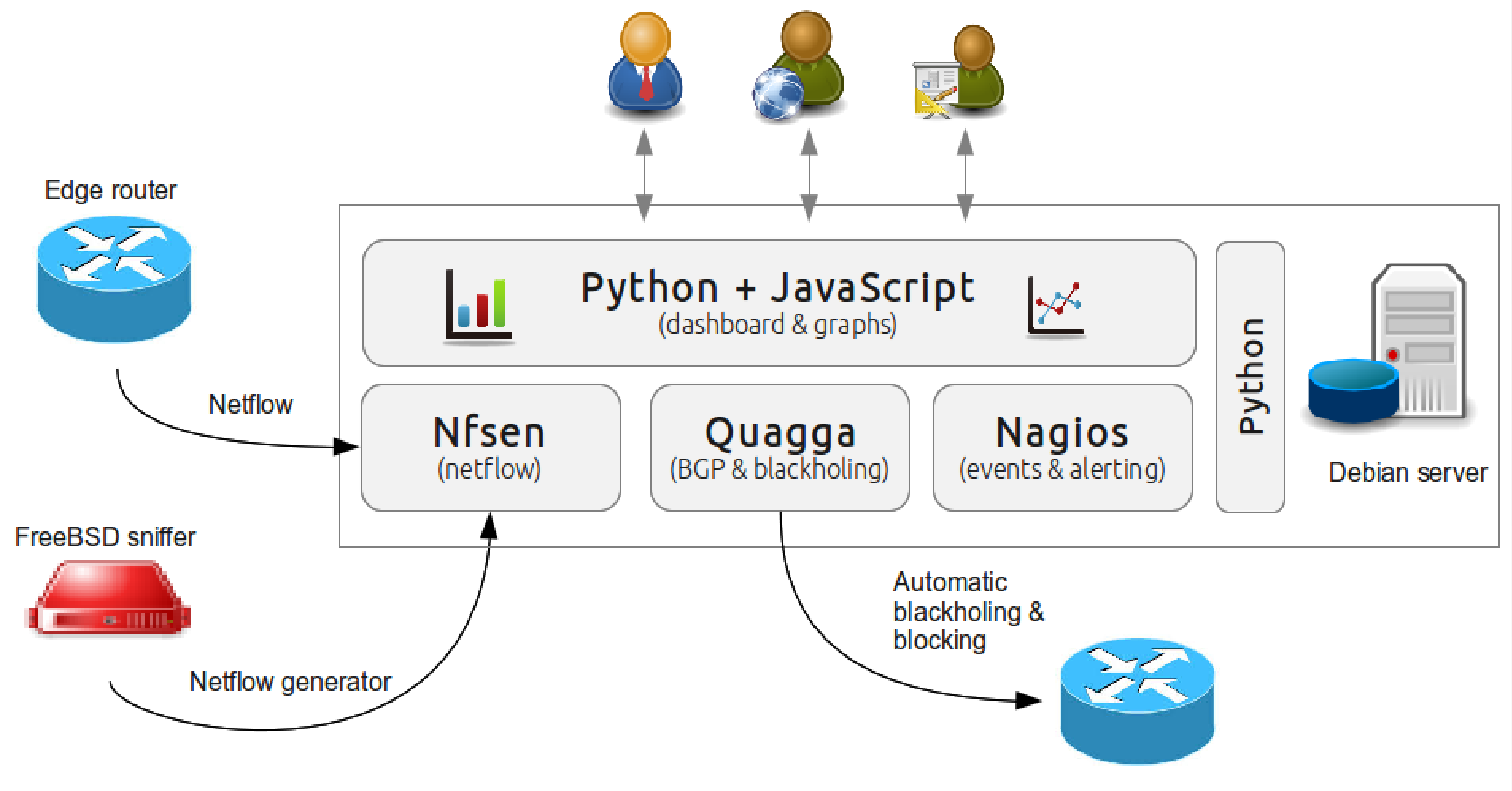


- Have the system model itself!
- We want the anomalous behavior to be part of a complete alert workflow with correlation

The frontend of the solution

- Using open source has huge advantages, particularly when adding new features, plugins and functionalities
- Nfsen for netflow aggregation
- Highcharts (JavaScript) for fancy frontend dashboard
- Nagios for alert the complete monitoring workflow + notifications (e-mail, SNMP traps to NMS)
- Quagga for eBGP integration for black-holing potential attackers
- Python scripts for backend modeling and dynamic thresholds of the system (using PyCluster and SciPy libraries)

System diagram



Edge router

Netflow

FreeBSD sniffer

Netflow generator

Python + JavaScript
(dashboard & graphs)

Nfsen
(netflow)

Quagga
(BGP & blackholing)

Nagios
(events & alerting)

Python

Debian server

Automatic blackholing & blocking

Practical implementation

- System implemented in ISP Neotel – Skopje, Macedonia
- 8Gbps traffic, 1 netflow source, sampling rate: 1 out of 1024
- Learning period: 15 days
- Seasonal parameters (Monday to Friday):
  - Night time01:00 – 08:00
  - Daytime normal activity 08:00 – 10:30 and 18:00 – 01:00
  - Daytime high activity: 10:30 – 18:00
- System implemented on OpenStack cloud
- Debian server integrating: Nfsen, Highcharts, Nagios, Quagga

Netflow generator box

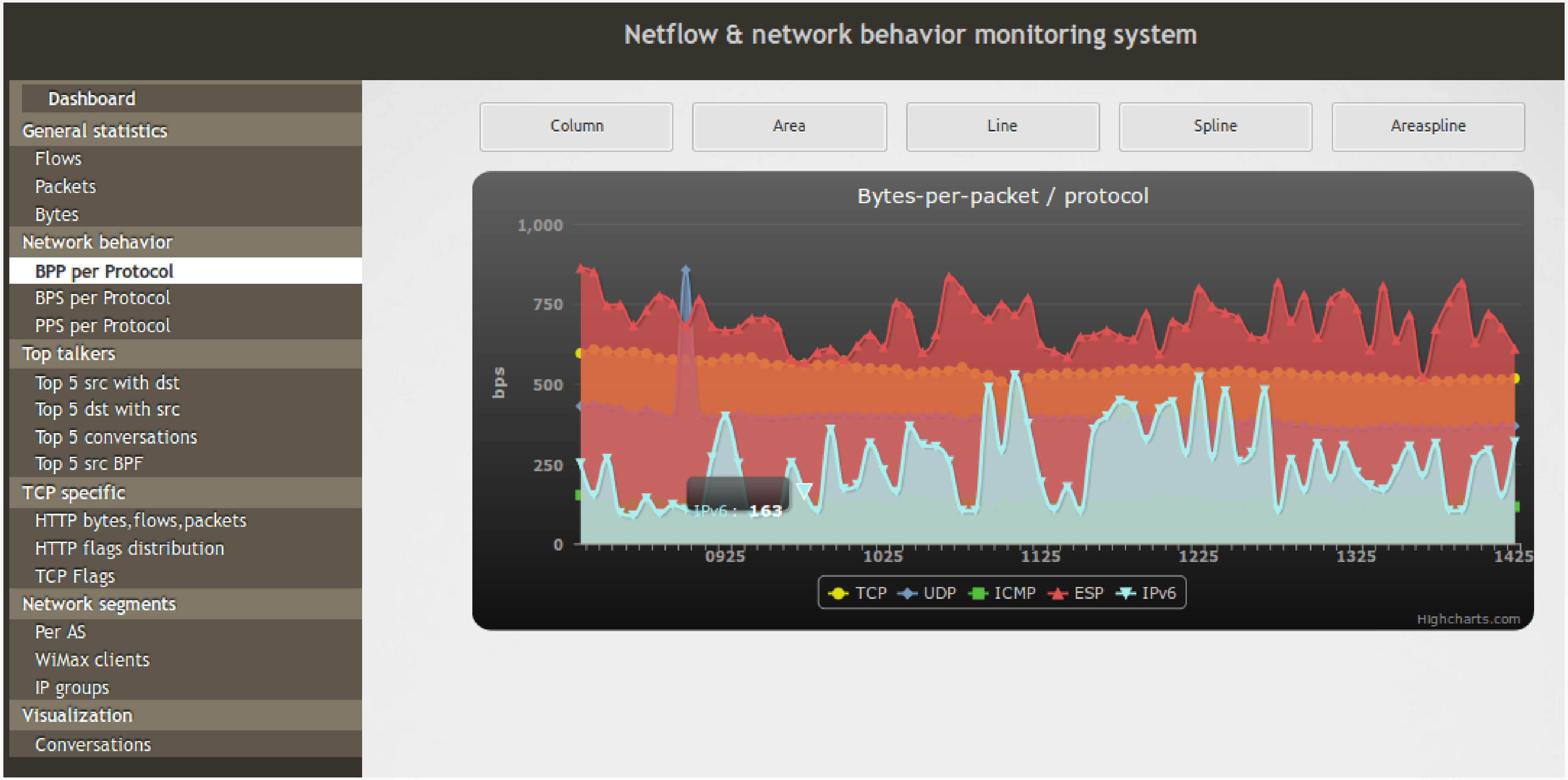
- What if we don't have netflow generator?
- We have tested netflow generator box sniffing data (from tap or port mirror) and generating netflow
- System:
  - FreeBSD 9.0
  - Customized and optimized kernel for high network load
  - ng_netflow netgraph kernel implementation of netflow (v5 and v9)
  - Broadcom GbE adapters
  - 1 cpu 4 x 2.4GHz (Xeon quadcore)
  - Sniffing ~900mbps traffic and generating netflow with 20% cpu load

Visualization

- Highcharts is used as frontend to Nfsen
- JavaScript running on client side
- Providing flexible and powerful visualization
- Integrating also Nagios and Quagga statistics and information on a single interface
- Additional graphing capabilities in addition to Nfsen (rrdtool)

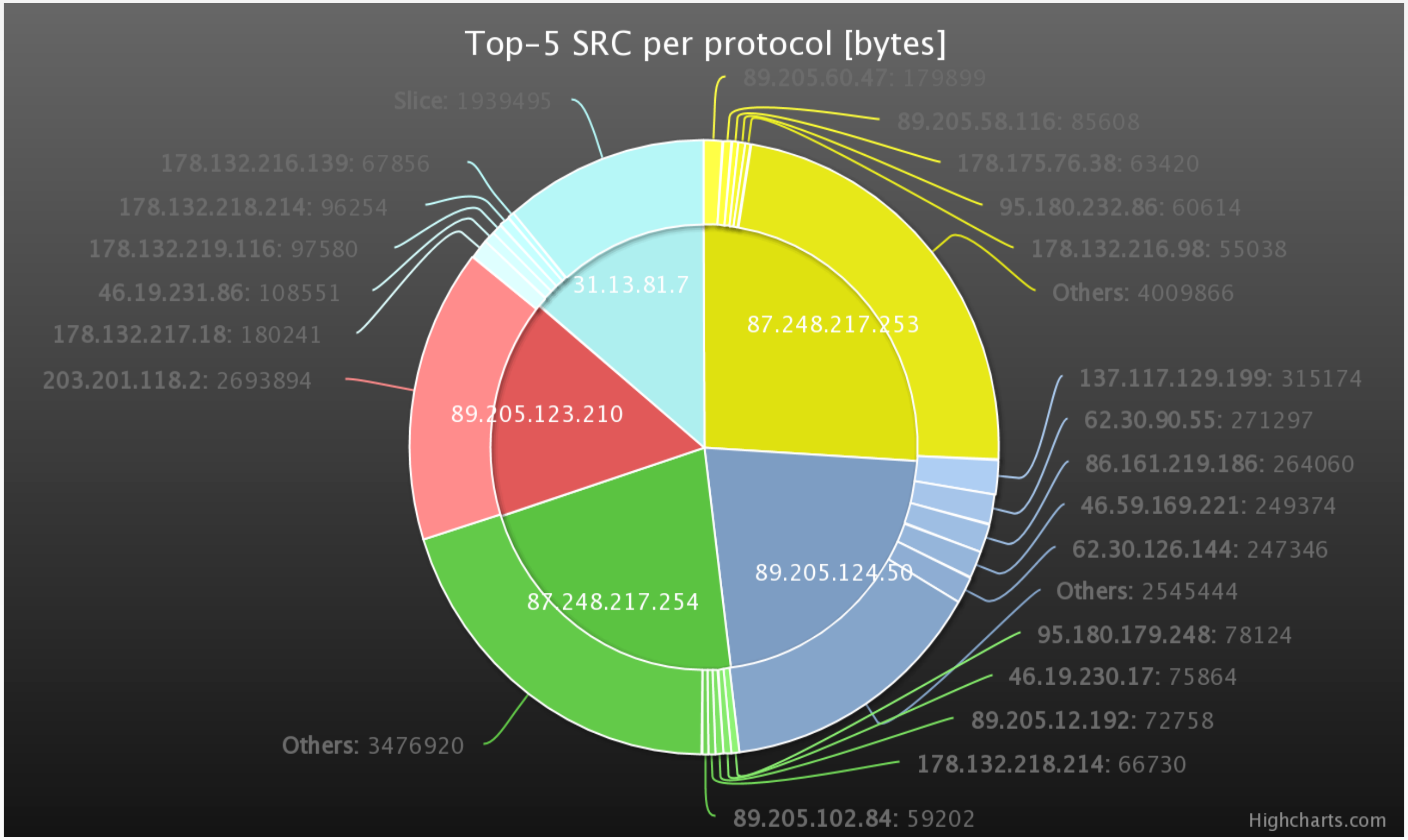
Visualization

- Bpp distribution over time (bytes-per-packet)



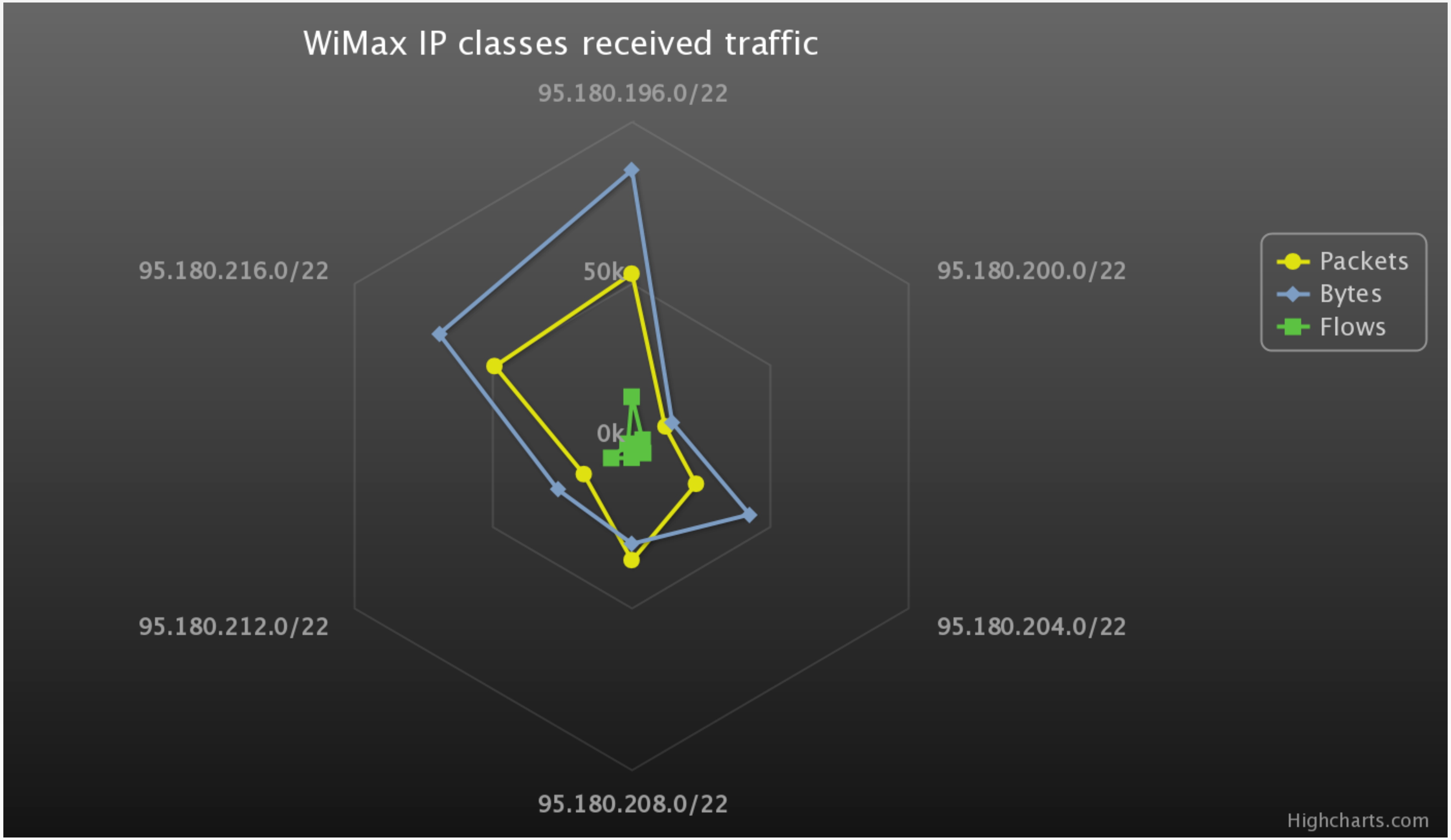
Visualization

- Top 5 src Ips with destinations



Visualization

- IP groups compared activity visualization



For the end customers

- The system is web based, Python + JavaScript (Highcharts)
- Customers can have insight in their network usage
- Submit requests for blocking traffic from a remote host
- Integrate with ticketing systems
- Customer should be able to submit a QoS/QoE problem and record of his current network behavior can be submitted as well. This allows for QoE issues to be addressed in a proper manner.

Conclusion

- Based on open source tools (excluding Highcharts library) – highly configurable and flexible
- Taking network monitoring to the next level: designing the system for the user
- Possible benefits for Neotel
- Future development:
  - REST API for integration with other applications
  - Integrating the application layer (pf layer-7 on FreeBSD)

Questions & answers

Thank you for your time

Ljupco Vangelski
ljupco@scope.mk