Sources and Applications of Performance and Security-Augmented Flow Data

Avi Freedman, CEO

FloCon 2016

January 2016

kentik

Background: NetFlow/IPFIX/sFlow

- NetFlow is:
  - A 20-year old technology now supported in some variant by most network devices, hosts, and sensors.
  - And much smaller than storing all packets, so useful for longer term metadata storage and search.
- sFlow came later, is simpler and more accurate in real-time because it's just packet sampling.
- IPFIX and Netflow v9 are extensible via templates, and allow sending more than just 'basic flow' data via those templates.
- Both via IPFIX/v9 and other formats, there are many sources of app semantics + performance data that work will with flow-like analysis patterns.

‘Classic’ Flow

‘Classic’ Flow

- Classic flow records contain byte and packet counters, TCP Flags, AS, next-hop, and other data aggregated by (usually) the ‘5 tuple’ of (protocol, srcip, dstip, srcport, dstport). VLAN, mac, MPLS, packet size histogram and other data often available.
- Most devices support a fixed sampling rate.
- Despite the relative simplicity of data, there are many use cases for basic flow data for monitoring availability, efficiency, and security of networks, hosts, and applications.

Classic Flow Use Cases

- Classic use cases include:
  - Congestion analysis for providers and/or customers
  - Peering analytics
  - Trending, planning and forecasting
  - (d)DoS detection (primarily volumetric)
  - Basic forensic/historic (who did an IP talk to)
  - Modeling of TE, what-if analysis
  - Customer cost analysis (Flow + BGP communities)
  - Many security use cases for even 'classic' Flow:
    - Convolve with threat feeds, DNS, BGP
    - Finding extrusion (or at least indicators), fast flux, botnet c+c, service scanning, long-lived low-bw comms, service compromise, anomalies in many dimensions…

Classic View: Traffic by Source ASN

Bits/s by AS_src

TIME OPTIONS Custom 2015-10-04 14:00 to 22:00 2015-10-04 UTC | GROUP BY METRIC Source AS Number | UNITS Bits/s | DATASET Auto | Apply Reset

Devices Search

Select All / None Selected: 1

- [x] **cat2_cloudhelix_com**
- [ ] core_nyc_isp
- [ ] com
- [ ] rx1_cloudhelix_com

Single Multi

Filters

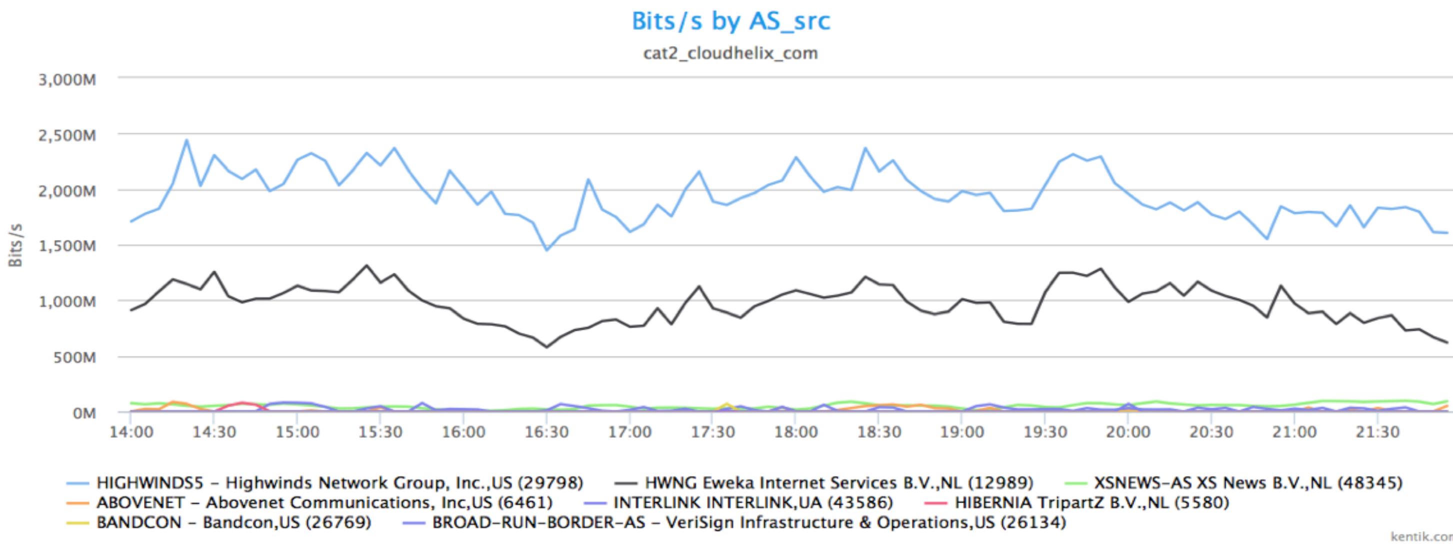
Add Group Clear All

Group 1

src_as <> 6450

Overlay - 0 days

Export SQL

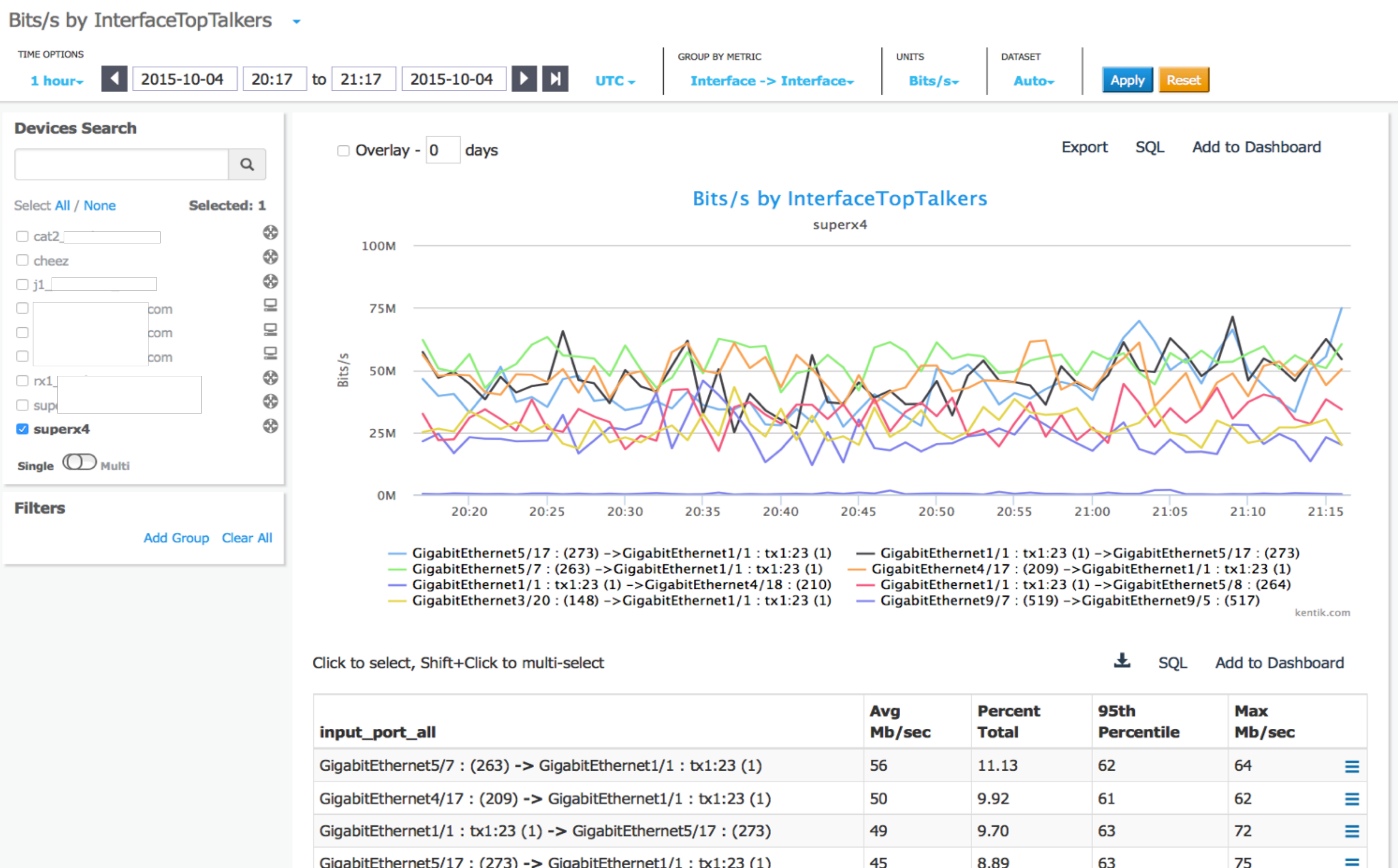


Click to select, Shift+Click to multi-select

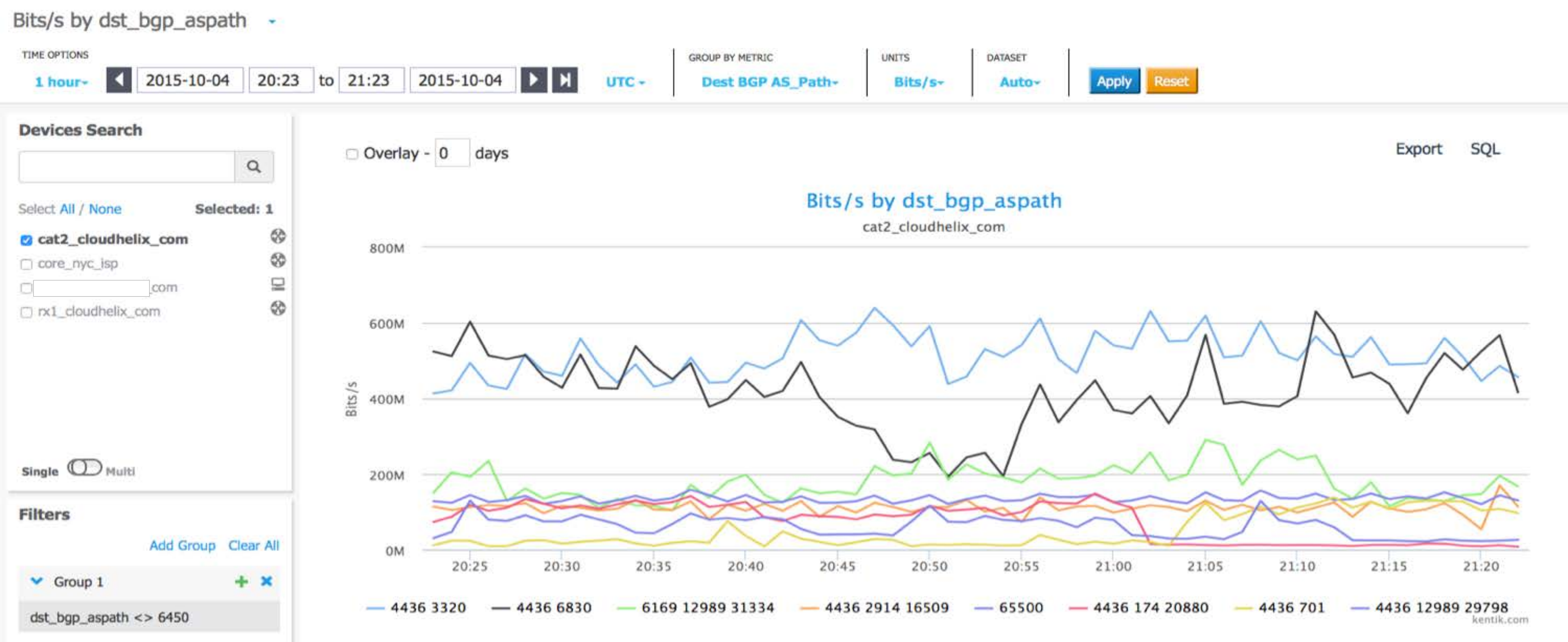
SQL

src_as	Avg Mb/sec	Percent Total	95th Percentile	Max Mb/sec
HIGHWINDS5 - Highwinds Network Group, Inc.,US (29798)	1,970	58.32	2,324	2,443
HWNG Eweka Internet Services B.V.,NL (12989)	981	29.02	1,247	1,315
XSNEWS-AS XS News B.V.,NL (48345)	52	1.53	92	96
COGENT-174 - Cogent Communications,US (174)	26	0.76	29	30
MICROSOFT-CORP-MSN-AS-BLOCK - Microsoft Corporation,US (8068)	22	0.65	32	32
HURRICANE - Hurricane Electric, Inc.,US (6939)	22	0.65	26	27
INTERLINK INTERLINK,UA (43586)	20	0.58	73	81

Classic View: Interface -> Interface Traffic



Classic View: Traffic by top AS_PATHs



Click to select, Shift+Click to multi-select

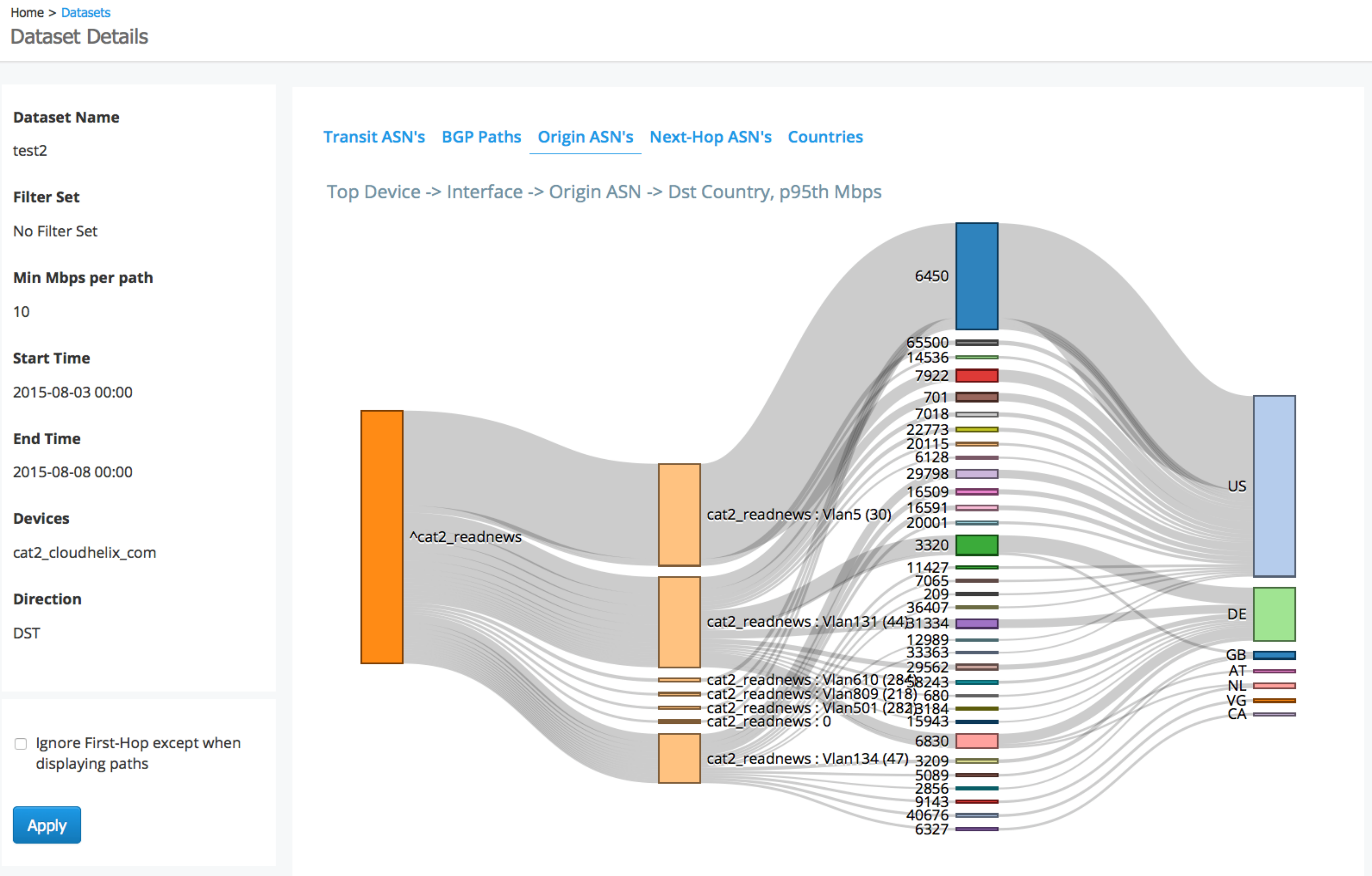
SQL

dst_bgp_aspath	Avg Mb/sec	Percent Total	95th Percentile	Max Mb/sec
4436 3320	522	20.67	612	640
4436 6830	429	16.99	569	631
6169 12989 31334	183	7.22	264	290
65500	138	5.44	152	159
4436 2914 16509	113	4.46	132	170
14536	104	4.09	118	124
4436 4436 7065	80	3.14	95	112
4436 174 20880	77	3.02	130	149

Classic View: dDoS Detection

Key	Alert Name	Criticality	State	Key Type	Output 1 Name:Value	Output 2 Name:Value	Alert ID	Start	End	Time Over Threshold	Recent Comment
	many_src_ips_to_1_dst	Major	ACK_REQ	ipv4_dst_addr	src_ips : 189	pps : 3277	3536	2015-08-26 20:25	2015-08-26 20:46	45%	
	high_fps_per_dst_ip	Major	ACK_REQ	ipv4_dst_addr	fps : 110	pps : 118835	3537	2015-08-26 20:25	2015-08-26 20:45	42%	
	all_dst53_or_src53_to_1ip ...	Major	ACK_REQ	ipv4_dst_addr	pps : 51166	mbps : 576	462	2015-08-26 20:25	2015-08-26 20:44	31%	
	udp_srcdst0_	Major	ACK_REQ	ipv4_dst_addr	pps : 86391	mbps : 914	452	2015-08-26 20:25	2015-08-26 20:44	31%	
	many_src_ips_to_1_dst	Major	ACK_REQ	ipv4_dst_addr	src_ips : 137	pps : 13517	3536	2015-08-26 20:37	2015-08-26 20:47	33%	

Classic View: Device to AS to Geo



‘Augmented’ Flow

‘Augmented’ Flow

- ‘Who talked to who’ data is great, but if we can get:
  - Semantics (URL, DNS query, SQL query, …)
  - Application performance info (latency, TTFB, …)
  - Network performance info (RTT, loss, jitter, …)

  from passive observation, it unlocks even more/more interesting use cases!
- With many of the same basic report structures.
- Some of this is already available via IPFIX/V9 from many devices. Or via nprobe and argus for host/sensor. Or as flow-like sources fo data.

Sources of ‘Augmented’ Flow

- Server-side
  - OSS sensor software: yaf, nprobe, argus
  - Commercial sensors: nBox, nPulse, and others
  - Packet Brokers: Ixia and Gigamon (IPFIX, potentially more)
  - IDS (bro) – a superset of most flow fields, + app decode
  - Web servers (nginx, varnish) – web logs + tcp_info for perf
  - Load balancers – advantage of seeing HTTPS-decoded URLs
  - CISCO AVC, Netflow Lite – generally only on small devices

- Common challenge: Some of the exporters don’t support sampling, and many tools can’t keep up with un-sampled flow. And many tools can’t easily map + store augflow fields. (Tradeoff: speed vs flexibility)

augflow Examples: yaf

- https://tools.netsa.cert.org/yaf/docs.html
- http://linux.die.net/man/1/yafdpi
- ‘Base’ yaf supports entropy, packet size distribution, estimation of TCP setup time (reverseFlowDeltaMilliseconds), app ID, and a few other fields.
- yafdpi can extract and send in IPFIX many varieties of application semantics: FTP, HTTP, IMAP, RTSP, SIP, SMTP, SSH, DNS, IRC, NNTP, POP3, SLP, TFTP, MySQL
- No performance data added

augflow Examples: nprobe

- http://ntop.org
- template.c in nprobe (and elsewhere)
- Includes both network and with pro plugins, some application performance, and many kinds of application semantics: DHCP, MySQL, RADIUS, DHCP, HTTP, SMTP, SIP, RTSP, GTP
- sysdig perhaps most interesting for APM-like use cases
- Can export in JSON and other formats in addition to IPFIX

```
  { 0, BOTH_IPV4_IPV6, FLOW_TEMPLATE, SHORT_SNAPLEN, NTOP_ENTERPRISE_ID,
NTOP_BASE_ID+110, STATIC_FIELD_LEN, 4, numeric_format, dump_as_uint,
"RETRANSMITTED_OUT_PKTS", "", "Number of retransmitted TCP flow packets (dst->src)" },
  { 0, BOTH_IPV4_IPV6, FLOW_TEMPLATE, SHORT_SNAPLEN, NTOP_ENTERPRISE_ID,
NTOP_BASE_ID+101, STATIC_FIELD_LEN, 2,  ascii_format, dump_as_ascii,
"SRC_IP_COUNTRY", "", "Country where the src IP is located" },
  { 0, BOTH_IPV4_IPV6, FLOW_TEMPLATE, SHORT_SNAPLEN, NTOP_ENTERPRISE_ID,
NTOP_BASE_ID+86,  STATIC_FIELD_LEN, 4, numeric_format, dump_as_uint,
"APPL_LATENCY_SEC", "", "Application latency (sec)" },
  { 0, BOTH_IPV4_IPV6, FLOW_TEMPLATE, SHORT_SNAPLEN, NTOP_ENTERPRISE_ID,
NTOP_BASE_ID+82,  STATIC_FIELD_LEN, 4, numeric_format, dump_as_uint,
"CLIENT_NW_DELAY_SEC", "",  "Network latency client <-> nprobe (sec)" },
```

augflow Examples: argus

- http://qosient.com/argus/
- http://qosient.com/argus/man/man1/ra.1.pdf
- Custom format to preserve richness that IPFIX does not allow
- Data can be sent in CSV for use (with augflow) with other tools
- Rich network-layer performance data (jitter, latency sliced many ways), infer topology, …

augflow Examples: Cisco AVC

docwiki.cisco.com/wiki/AVC-Export:PfR#PfR_NetFlow_Export_CLI

```
Client: Option Active Performance
Exporter Format: NetFlow Version 9
Template ID     : 268
Source ID       : 0
Record Size     : 61
Template layout
```

Field	Type	Offset	Size
flow end	153	0	8
pfr br ipv4 address	39000	8	4
reason id	39002	12	4
counter packets dropped	37000	16	4
transport packets lost counter	37019	20	4
transport round-trip-time	37016	24	4
transport rtp jitter mean	37023	28	4
mos worst 100	42115	32	4
counter packets dropped permanent short	37001	36	4
transport packets lost counter permanen	37020	40	4
long-term round-trip-time	39006	44	4
flow class wide	95	48	6
interface output snmp short	14	54	2
pfr status	39001	56	2
flow active timeout	36	58	2
ip protocol	4	60	1

augflow Examples: Citrix AppFlow

http://docs.citrix.com/en-us/netscaler/10-5/ns-system-wrapper-10-con/ns-ag-appflow-intro-wrapper-con.html

https://github.com/splunk/ipfix/blob/master/app/Splunk_TA_IPFIX/bin/IPFIX/information-elements/netscaper-iana.xml_full

tcpRTT

The round trip time, in milliseconds, as measured on the TCP connection. This can be used as a metric to determine the client or server latency on the network.

httpRequestMethod

An 8-bit number indicating the HTTP method used in the transaction. An options template with the number-to-method mapping is sent along with the template.

httpRequestSize

An unsigned 32-bit number indicating the request payload size.

httpRequestURL

The HTTP URL requested by the client.

augflow Examples: nginx, bro

- http://nginx.org/en/docs/http/ngx_http_core_module.html#variables
- https://www.bro.org/sphinx/logs/index.html
- **Not 'flow' but can be translated and stored similarly!**

nginx: log_format combined '$remote_addr - $remote_user [$time_local] ' '"$request" $status $body_bytes_sent ' '"$http_referer" "$http_user_agent"' '$tcpinfo_rtt, $tcpinfo_rttvar, $tcpinfo_snd_cwnd, $tcpinfo_rcv_space';

```
# cat conn.log | bro-cut id.orig_h id.orig_p id.resp_h duration
141.142.220.202    5353    224.0.0.251     -
fe80::217:f2ff:fed7:cf65  5353    ff02::fb        -
141.142.220.50     5353    224.0.0.251     -
141.142.220.118    43927   141.142.2.2     0.000435
141.142.220.118    37676   141.142.2.2     0.000420
141.142.220.118    40526   141.142.2.2     0.000392
141.142.220.118    32902   141.142.2.2     0.000317
141.142.220.118    59816   141.142.2.2     0.000343
141.142.220.118    59714   141.142.2.2     0.000375
141.142.220.118    58206   141.142.2.2     0.000339
[...]
```

Storing and Accessing Augmented Flow

- Data back-ends need to be able to understand and ingest the extra fields.
- Often requires integration (for OSS/big data tools) or vendor support.
- And if the tools aren't 'open' via API, SQL, or CLI, data can be trapped and not as useful.
- Many first use cases are ad-hoc to prove effectiveness, then drive to UI reports/dashboards.
- Holy grail: semantics + end user app perf + net perf + net flow + host perf + app internals instrumentation.
- Note: Semantics also useful for performance and performance data useful for security!

One Extensible Flow Storage: fastbit

- https://sdm.lbl.gov/fastbit/
- https://github.com/CESNET/ipfixcol/
- http://www.ntop.org

```
(nprobe CLI)
fbquery -c
'DST_AS,L4_SRC_PORT,sum(IN_BYTES) as
inb,sum(OUT_BYTES) as outb' \
-q 'SRC_AS <> 3 AND L4_SRC_PORT <> 80' \
-g 'DST_AS,L4_SRC_PORT' \
-o 'inb' \
-r -L 10 -d .
```

Storing Augmented Flow in Fastbit

```
root@s5:/data/fb/333/dev1/3/2015/10/03/20/49# ls
APPLATENCY                IPV4_DST_ADDR.idx              OUT_PKTS
APPLATENCY.idx            IPV4_DST_ROUTE_PREFIX          OUT_PKTS.idx
CTIMESTAMP                IPV4_DST_ROUTE_PREFIX.idx      PROTOCOL
CTIMESTAMP.idx            IPV4_NEXT_HOP                  PROTOCOL.idx
DEFAULT_COLUMN            IPV4_NEXT_HOP.idx              SAMPLEDPKTSIZE
DEFAULT_COLUMN.idx        IPV4_SRC_ADDR                  SAMPLEDPKTSIZE.idx
DEVICE_ID                 IPV4_SRC_ADDR.idx              SAMPLE_RATE
DEVICE_ID.idx             IPV4_SRC_ROUTE_PREFIX          SAMPLE_RATE.idx
DNS                       IPV4_SRC_ROUTE_PREFIX.idx      SRC_AS
DNSQ.idx                  IPV6_DST_ADDR_HIGH             SRC_AS.idx
DST_AS                    IPV6_DST_ADDR_HIGH.idx         SRC_GEO
DST_AS.idx                IPV6_DST_ADDR_LOW              SRC_GEO.idx
DST_GEO                   IPV6_DST_ADDR_LOW.idx          SRC_GEO_CITY
DST_GEO.idx               IPV6_SRC_ADDR_HIGH             SRC_GEO_CITY.idx
DST_GEO_CITY              IPV6_SRC_ADDR_HIGH.idx         SRC_GEO_REGION
DST_GEO_CITY.idx          IPV6_SRC_ADDR_LOW              SRC_GEO_REGION.idx
DST_GEO_REGION            IPV6_SRC_ADDR_LOW.idx          SRC_ROUTE_LENGTH
DST_GEO_REGION.idx        L4_DST_PORT                    SRC_ROUTE_LENGTH.idx
DST_ROUTE_LENGTH          L4_DST_PORT.idx                TCP_FLAGS
DST_ROUTE_LENGTH.idx      L4_SRC_PORT                    TCP_FLAGS.idx
INPUT_PORT                L4_SRC_PORT.idx                TCP_RETRANSMIT
INPUT_PORT.idx            MPLS_TYPE                      TCP_RETRANSMIT.idx
IN_BYTES                  MPLS_TYPE.idx                  TOS
IN_BYTES.idx              OUTPUT_PORT                    TOS.idx
IN_PKTS                   OUTPUT_PORT.idx                URL
IN_PKTS.idx               OUT_BYTES                      URL.idx
IPV4_DST_ADDR             OUT_BYTES.idx
```

Use Case: Network Performance

- If the flow system can aggregate by arbitrary dimensions by AS, AS_PATH, Geo, Prefix, etc…
- Then looking at raw network performance from passive sources can be very useful.
- Ex: TCP rexmit by AS_PATH (i.e. from nprobe for a server or, via span/tap, a sensor).
- Important to weight absolute relevance (not just % loss if a few 3 pkt flows).

SQL -> Fastbit Querying for rexmit

Retransmits > .1% by ASN at prime-time for ASNs with > 10k pkts:

```
SELECT i_start_time, src_AS, dst_AS,
sum(tcp_retransmit) AS f_sum_tcp_retransmit,
sum(out_pkts) AS f_sum_out_pkts,
round((sum(tcp_retransmit)/sum(out_pkts))*1000)/10
AS Perc_retransmits FROM _com WHERE
i_start_time >= '2015-01-09 22:00:00' AND
i_start_time < '2015-01-10 06:00:0' GROUP BY
src_AS, dst_AS, i_start_time HAVING sum(out_pkts) >
10000 AND (sum(tcp_retransmit)/sum(out_pkts))*100 >
0.1 ORDER BY Perc_retransmits DESC;
```

Augmented Flow: rexmit by Dest ASN

% Retransmits by AS_dst

TIME OPTIONS 1 hour 2015-10-04 20:57 to 21:57 2015-10-04 UTC | GROUP BY METRIC Dest AS Number | UNITS % Retransmits Min pps 500 | DATASET Auto | Apply Reset

Devices Search

Select All / None Selected: 1

- cat2_cloudhelix_com
- core_nyc_isp
- com
- rx1_cloudhelix_com

Single Multi

Filters

Add Group Clear All

Group 1

dst_bgp_aspath <> 6450

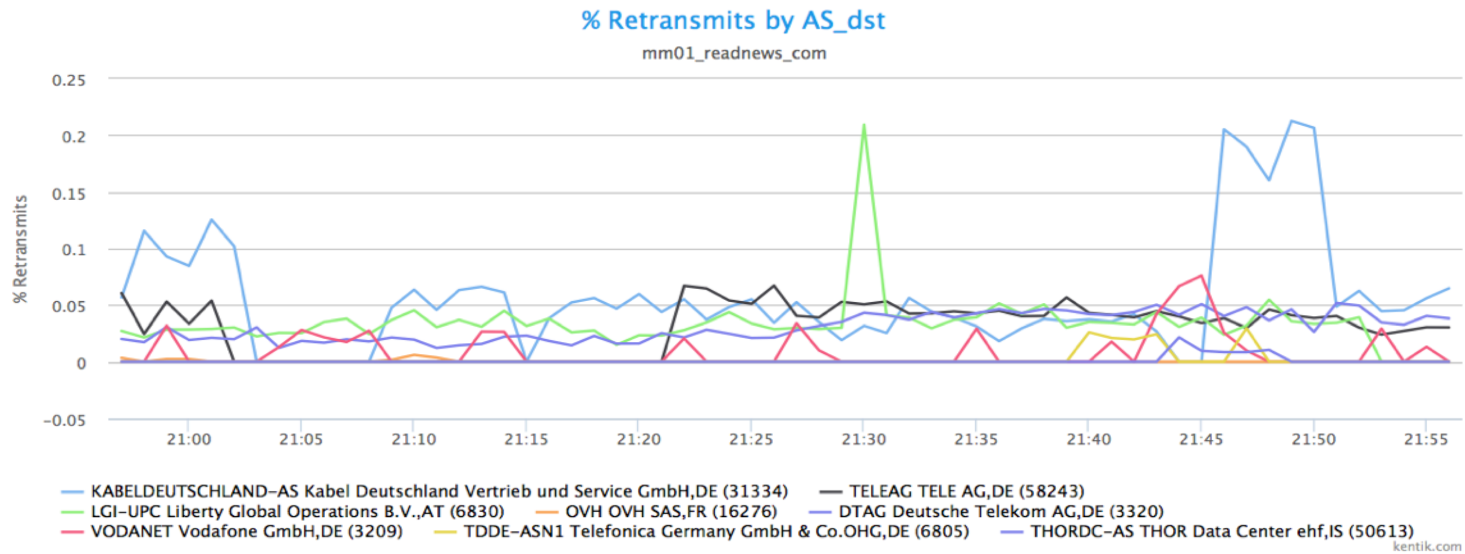
and

Group 2

dst_as <> 29562

Overlay - 0 days

Export SQL



Click to select, Shift+Click to multi-select

SQL

	# of Retransmits				% of Retransmits			Total Traffic	
dst_as	**total**	**Avg /sec**	**95th percentile**	**Max/sec**	**Avg /sec**	**95th percentile**	**Max/sec**	**Avg mbps Sent**	**Avg pkts/s Sent**
KABELDEUTSCHLAND-AS Kabel Deutschland Vertrieb und Service GmbH,DE (31334)	2015	0.55972	1.81667	2.03333	0.06236	0.20515	0.21272	36	898
LGI-UPC Liberty Global Operations B.V.,AT (6830)	2417	0.67139	0.96667	3.41667	0.03415	0.05154	0.20939	87	1,966
VODANET Vodafone GmbH,DE (3209)	236	0.06556	0.60000	0.71667	0.03096	0.06688	0.07601	8	212
TELEAG TELE AG,DE (58243)	1401	0.38917	0.80000	1.11667	0.04325	0.06483	0.06740	35	900
DTAG Deutsche Telekom AG,DE (3320)	3183	0.88417	1.50000	1.96667	0.03168	0.04959	0.05188	102	2,792

Augmented Flow: rexmit by AS_PATH

% Retransmits by dst_bgp_aspath

TIME OPTIONS: 1 hour | 2015-10-04 20:55 to 21:55 2015-10-04 | UTC

GROUP BY METRIC: Dest BGP AS_Path

UNITS: % Retransmits | Min pps 500

DATASET: Auto

Apply | Reset

Devices Search

Select All / None — Selected: 1

- cat2_cloudhelix_com
- core_nyc_isp
- com
- rx1_cloudhelix_com

Single / Multi

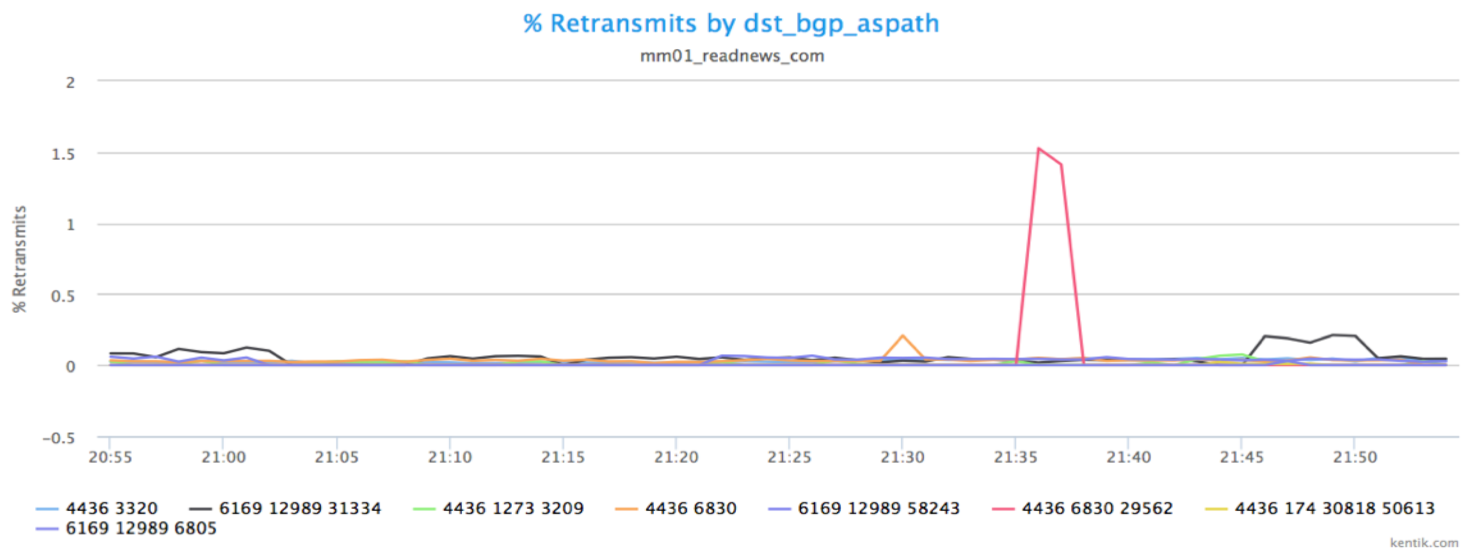
Filters

Add Group | Clear All

Group 1

dst_bgp_aspath <> 6450

Overlay - 0 days — Export | SQL



Click to select, Shift+Click to multi-select — SQL

	# of Retransmits				% of Retransmits			Total Traffic	
dst_bgp_aspath	total	Avg /sec	95th percentile	Max/sec	Avg /sec	95th percentile	Max/sec	Avg mbps Sent	Avg pkts/s Sent
4436 6830 29562	912	0.25333	7.95000	7.95000	1.46997	1.52733	1.52733	1	18
6169 12989 31334	2025	0.56250	1.81667	2.03333	0.06279	0.20515	0.21272	35	896
4436 6830	2451	0.68083	0.96667	3.41667	0.03415	0.05154	0.20939	88	1,994
4436 1273 3209	249	0.06917	0.60000	0.71667	0.03105	0.06688	0.07601	8	223
6169 12989 58243	1389	0.38583	0.80000	1.11667	0.04438	0.06483	0.06740	34	870
4436 3320	3071	0.85306	1.50000	1.96667	0.03092	0.04959	0.05188	102	2,760

Use Case: Application-Level Attacks

- With URL and performance data, many kinds of application attacks can be detected.
- To get * URL info in an HTTPS world, will need to get data from load balancers or web logs.
- Simplest is WAF – looking for SQL fragments, binary, or other known attack vectors.
- Can hook alerts to mitigation methods, even if running OOB (for example, send TCP FIN/RST in both directions)

Use Case: ‘APM Lite’

- Combining network with application data, you can answer questions like:
  - Show/aggregate cases where application performance is impaired but we know there is no network-layer issue (very useful), and agg by POP, server, app section.
  - Or where there is impairment in both.
  - And ignore network-layer issues where users are unaffected.
- Easy first use case: API perf debugging for web page assembly, or debugging CDN origin pull.

Use Case: Bot detection

- With performance information combined with URL, basic e-commerce bot detection is possible.
- Many attacks are advanced so may require a packet approach to get complete visibility, but basic visibility can often demonstrate a problem.
- Can sometimes be done with syslog analytics, but flow tools often aggregate in interesting ways (geo, AS) that syslog analytics don’t, at least out of the box.

Modern ‘Flow’ Format: kflow

- At today’s speeds, async(ly)-templated formats may not be the most efficient (space/CPU) implementation.
- See also: http://www.ntop.org/nprobe/yes-theres-life-after-netflow/ (but Kentik prefers binary to JSON/Kafka)
- Working on an open-spec format called kflow with open source tools to take to and from NetFlow, sFlow, IPFIX, nginx and bro logs, and Cisco, Citrix, ntop, and other vendor formats.
- Based on Cap’n Proto, which is a ‘serialization’ lib that is basically a struct with 0-packing - https://capnproto.org/
- Drawback: Can’t delete fields, just 0-pack them.
- Will shortly be live at https://github.com/Kentik

Flow with Cap’n Proto kflow v1

```
struct kflow_v1 {
        version @44: Int64;
        timestampNano @0: Int64;
        dstAs @1: UInt32;
        dstGeo @2: UInt32;
        dstMac @3: UInt32;
        headerLen @4: UInt32;
        inBytes @5: UInt64;
        inPkts @6: UInt64;
        inputPort @7: UInt32;
        ipSize @8: UInt32;
        ipv4DstAddr @9: UInt32;
        ipv4SrcAddr @10: UInt32;
        tcpRetransmit @27: UInt32;
        dstBgpAsPath @34: Text;
        dstBgpCommunity @35: Text;
        <...>
```

kflow v2

- Cap’n Proto is fast and compact.
- But to support many extended fields (imagine every bro and argus field) could still become unwieldly.
- And can’t easily delete fields/version.
- Don’t love templating, but ading OOB vs inband+async templates. Generated .H from server-side tools:

static short KFLOW_CUSTOM_COLULMNS[] = {KFLOW_CUSTOM_FOO_FID, KFLOW_CUSTOM_MYINT_FID, KFLOW_CUSTOM_MYFLOAT_FID};

static short KFLOW_CUSTOM_COLULMN_TYPES[] = {KFLOW_CUSTOM_FOO_TYPE, KFLOW_CUSTOM_MYINT_TYPE, KFLOW_CUSTOM_MYFLOAT_TYPE};

```
(*kfrec)->dst_as = 10;
(*kfrec)->src_as = num;
(*kfrec)->sample_rate = 1028;
(*kfrec)->src_bgp_as_path = strdup("100 200 300");
(*kfrec)->custom[KFLOW_CUSTOM_MYINT_OFF].val.i_val = 128;
(*kfrec)->custom[KFLOW_CUSTOM_MYFLOAT_OFF].val.f_val = 64.5;
(*kfrec)->custom[KFLOW_CUSTOM_FOO_OFF].val.s_val = strdup("FOO”);
```

Summary/Takeaways

- Many sources of ‘augmented flow’
- Even web and bro/snort/suricata logs
- Finding a tradeoff between flexibility and speed in storage can be a challenge
- But with unified augflow data, the same flow forensics repositories can do triple or more duty with operational, performance, and additional security analytics

Comments / Questions?

Avi Freedman
avi (at) kentik.com

kentik