PERISCOPE: Standardizing and Orchestrating Looking Glass Querying

Vasileios Giotsas
UCSD/CAIDA
vgiotsas@caida.org
Purpose of this Talk

• Inform the operational community about Periscope.

• Solicit feedback:
  o Details that we may have missed
  o Ways to make Periscope more useful
  o Technical insights, usage statistics, historical data …

• Encourage engagement and contributions
High-level Goals and Principles of Periscope

Periscope unifies the discovery and querying of Looking Glasses under a uniform API

- Respect resource limitations and preserve conservative query rates.
- Provide transparency and accountability in Looking Glass querying.
- Be responsive and compliant to operators’ requests.
Benefits to Operators and Researchers

- Easier to discover and query VPs for reverse paths.
- Improved monitoring and troubleshooting capabilities.
- Easier policing of Looking Glass usage through an access-control layer.
- Improved utilization and load distribution.
- Avoid redundant measurements by capturing and making public historical measurement data.
What is a Looking Glass (LG)?

- Web interfaces to routers and servers that allow the execution of non-privileged networking commands:
  - show bgp summary, show ip bgp, traceroute, ping, ...
Advantages of LG measurements

- LGs combine features not available in other platforms:
  - Access to non-transitive BGP attributes (e.g. Local Preference)
  - Co-located control-plane and data-plane monitors
  - Monitors inside critical infrastructures (IXPs, Colocation Facilities, border routers)
  - Vantage points in ASes not covered by other platforms

- LGs are among the few public measurement tools that provide direct interfaces to routers.

LGs are widely used by researchers, operators and users.
Problems with LG measurements

- Lack of standardization and consistency:
  - Disparate input interfaces, output formats, supported commands
- LGs are hard to discover
  - No centralized index of LGs and their corresponding locations
- Historical measurements are not archived:
  - Loss of reusable information, potential query redundancy
- LGs have high attrition rates:
  - Hard to maintain an up-to-date database of LGs

Problems with LG measurements

• Lack of standardization and consistency:
  o Disparate input interfaces, output formats, supported commands

  **Periscope** implements a common querying scheme, indexing, and data persistence features.
  
  o Loss of reusable information, potential query redundancy

• LGs have high attrition rates:
  o Hard to maintain an up-to-date database of LGs

Periscope Workflow

1. API Request

```
{
  "command": "bgp",
  "destination": "103.22.203.0/24",
  "sources": [
    {
      "asn": 680, "host": "Stuttgart_DE"},
    {
      "asn": 766, "host": "Madrid_ES"}
  ]
}
```

2. HTTP Request

- Request URL: https://www.noc.dfn.de/lg/
- Request Method: POST
- Status Code: 200 OK
- Remote Address: 194.95.237.14:443

3. HTTP Response

```
BGP routing table entry for "
<b>103.22.203.0/24</b>,
" version 126801054
BGP Bestpath: deterministic-med
Paths: (2 available,)
  (font color="#FF0000")best 
  #2</font>,
  table default)
Advertised to update-groups:
  8
Refresh Epoch 1
  3356 3356 6453 13335
  188.1.200.77 (metric 1141) from "
<a href="/lg?query=bgp&protocol=IPv4&as680">
```

4. API Response

```
{
  "source": "AS680_XR-STU1_Stuttgart_DE",
  "destination": "103.22.203.0/24",
  "AS_path": ["3356", "3356", "6453", "13335"],
  "best": true,
  "communities": ["680:66", "3356:86", "6453:3000"],
  "localpref": "100",
  "next_hop": "188.1.200.77",
  "datetime": "2016-03-23 05:41:05"
}
```
Periscope Workflow

1. API Request

Periscope receives measurement requests in standardized format

```
{
    "command": "bgp",
    "destination": "103.22.203.0/24",
    "sources": [
        {"asn": 680, "host": "Stuttgart_DE"},
        {"asn": 766, "host": "Madrid_ES"}
    ]
}
```

2. HTTP Request

- General
  - Request URL: https://www.noc.dfn.de/lg/
  - Request Method: POST
  - Status Code: 200 OK
  - Remote Address: 194.95.237.14:443

- Form Data
  - Query: bgp
  - Protocol: IPv4
  - Addr: 103.22.203.0/24
  - Router: Stuttgart%3A+XR STU1

3. HTTP Response

```
BGP routing table entry for "103.22.203.0/24",
version 126601054
BGP Bestpath: deterministic-med
Paths: (2 available),
  (best #2),
    best #2: 3356 3356 6453 13335
    version 126601054
    BGP Bestpath: deterministic-med
    Paths: (2 available),
      (best #2),
        best #2: 3356 3356 6453 13335
        version 126601054
        BGP Bestpath: deterministic-med
        Paths: (2 available),
          (best #2),
            best #2: 3356 3356 6453 13335
            version 126601054
            BGP Bestpath: deterministic-med
            Paths: (2 available),
              (best #2),
                best #2: 3356 3356 6453 13335
                version 126601054
                BGP Bestpath: deterministic-med
                Paths: (2 available),
                  (best #2),
                    best #2: 3356 3356 6453 13335
                    version 126601054
                    BGP Bestpath: deterministic-med
                    Paths: (2 available),
                      (best #2),
                        best #2: 3356 3356 6453 13335
                        version 126601054
                        BGP Bestpath: deterministic-med
                        Paths: (2 available),
                          (best #2),
                            best #2: 3356 3356 6453 13335
                            version 126601054
                            BGP Bestpath: deterministic-med
                            Paths: (2 available),
                              (best #2),
                                best #2: 3356 3356 6453 13335
                                version 126601054
                                BGP Bestpath: deterministic-med
                                Paths: (2 available),
                                  (best #2),
                                    best #2: 3356 3356 6453 13335
                                    version 126601054
                                    BGP Bestpath: deterministic-med
                                    Paths: (2 available),
                                      (best #2),
                                        best #2: 3356 3356 6453 13335
                                        version 126601054
                                        BGP Bestpath: deterministic-med
                                        Paths: (2 available),
                                          (best #2),
                                            best #2: 3356 3356 6453 13335
                                            version 126601054
                                            BGP Bestpath: deterministic-med
                                            Paths: (2 available),
                                              (best #2),
                                                best #2: 3356 3356 6453 13335
                                                version
```

4. API Response

```
{
    "source": "AS680.XR-STRU1_Stuttgart_DE",
    "destination": "103.22.203.0/24",
    "AS_path": ["3356", "3356", "6453", "13335"],
    "best": true,
    "communities": ["680:66", "3356:86", "6453:3000"],
    "localpref": "100",
    "next_hop": "188.1.200.77",
    "datetime": "2016-03-23 05:41:05"
}
```
Periscope Workflow

1. API Request

```
{
    "command": "bgp",
    "destination": "103.22.203.0/24",
    "sources": [
        {"asn": 680, "host": "Stuttgart_DE"},
        {"asn": 766, "host": "Madrid_ES"}
    ]
}
```

2. HTTP Request

- **Request URL**: https://www.noc.dfn.de/lg/
- **Request Method**: POST
- **Status Code**: 200 OK
- **Remote Address**: 194.95.237.14:443
- **Form Data**:
  - `query`: bgp
  - `protocol`: IPv4
  - `addr`: 103.22.203.0/24
  - `router`: Stuttgart%3A+XR-STU1

3. HTTP Response

```
BGP routing table entry for "
  version 126800654
  Bestpath: deterministic-med
  Paths: (2 available, )
  <font color="#FF0000">best #2</font>
  , table default) 
    Advertised to update-groups: 
      8 
    Refresh Epoch 1
      3356 3356 6453 13335
      188.1.200.77 (metric 1141) from "
      <a href="/lg/?query=bgp&protocol=IPv4&"..."
```

4. API Response

```
{
    "source": "AS680_XR-STU1_Stuttgart_DE",
    "destination": "103.22.203.0/24",
    "AS_path\": ["3356", "3356", "6453", "13335"],
    "best": true,
    "communities": ["680:66", "3356:86", "6453:3000"],
    "localpref": "100",
    "next_hop": "188.1.200.77",
    "datetime": "2016-03-23 05:41:05"
}
```
Periscope Workflow

1. API Request

```
{
  "command": "bgp",
  "destination": "103.22.203.0/24",
  "sources": [
    {"asn": 680, "host": "Stuttgart_DE"},
    {"asn": 766, "host": "Madrid_ES"}
  ]
}
```

2. HTTP Request

- **Request URL:** https://www.noc.dfn.de/lg/
- **Request Method:** POST
- **Status Code:** 200 OK
- **Remote Address:** 194.95.237.14:443

- **Form Data**
  - `query`: bgp
  - `protocol`: IPv4
  - `addr`: 103.22.203.0%F24
  - `router`: Stuttgart%3A+XR-STU1

3. HTTP Response

```
LGs return the raw HTML output
```

4. API Response

```
{
  "source": "AS680_XR-STU1_Stuttgart_DE",
  "destination": "103.22.203.0/24",
  "AS_path":["3356","3356","6453","13335"],
  "best": true,
  "communities":["680:66","3356:86","6453:3000"],
  "localpref": "100",
  "next_hop": "188.1.200.77",
  "datetime": "2016-03-23 05:41:05"
}
```
Periscope Workflow

1. API Request

2. HTTP Request

3. HTTP Response

4. API Response

Translation of the HTML output to a standardized JSON representation

```json
{
    "command": "bgp",
    "destination": "103.22.203.0/24",
    "sources": [
        {
            "asn": 680,
            "host": "Stuttgart_DE"
        },
        {
            "asn": 766,
            "host": "Madrid_ES"
        }
    ]
}
```

```json
{
    "source": "AS680_XR-STU1_Stuttgart_DE",
    "destination": "103.22.203.0/24",
    "AS_path": ["3356", "3356", "6453", "13335"],
    "best": true,
    "communities": ["680:66", "3356:86", "6453:3000"],
    "localpref": "100",
    "next_hop": "188.1.200.77",
    "datetime": "2016-03-23 05:41:05"
}
```
Implementation Challenges

• Automatically understand the disparate input/output formats of each LG.
• Automatically discover new LGs, detect changes in the status and capabilities of already supported LGs:
  o Manual parsing is impractical
• Support multiple concurrent users while preserving the query rates of native LG querying.
• Optimize the number of satisfied queries within restrictive querying budgets.
LG Ingestion Process

- PeeringDB
  - LG URLs
- Traceroute.org
  - LG URLs
  - http://www.phplg.com/
  - https://github.com/Cha0sgr/BGP-Looking-Glass-NG
  - http://sourceforge.net/projects/klg
  - http://mrlg.op-sec.us/
  - https://www.gw.com/sw/stripes/

- Web Crawler
- HTML forms
- Automatic configuration extraction
  - Successful mapping
    - Yes: LG specs
    - No: Manual inspection needed
- LG templates

- Health checker
LG Ingestion Process

- Collect and filter LG URLs
- Exclude LGs that prohibit scripts
- Extract LG input forms

PeeringDB
Traceroute.org

LG URLs

Web Crawler

HTML forms

LG templates

Automatic configuration extraction

Successful mapping

Manual inspection needed

No

Yes

Health checker

LG specs

http://www.phplg.com/
https://github.com/Cha0sgr/BGP-Looking-Glass-NG
http://sourceforge.net/projects/klg
http://mrlg.op-sec.us/
https://www.gw.com/sw/stripes/
Only ~65% of the collected LG URLs point to responsive LG, the rest return errors or do not correspond to LGs.
LG Ingestion Process

Match the extracted input forms against LG templates derived from open-source LG implementations

PeeringDB
Traceroute.org

Web Crawler

HTML forms

LG URLs

Automatic configuration extraction

Manual inspection needed

Successful mapping

Health checker

LG specs

http://www.phplg.com/
https://github.com/Cha0sgr/BGP-Looking-Glass-NG
http://sourceforge.net/projects/klg
http://mrlg.op-sec.us/
https://www.gw.com/sw/stripses/
LG Ingestion Process

- Over 85% of the LGs are mapped to an LG template → Periscope automatically interprets their input parameters
LG Ingestion Process

A health-checker executes weekly test queries to ensure the validity of the extracted LG specifications.
Periscope Architecture v0.1

• The **Query Interpreter** uses the outcome of the Ingestion Process to translate standardized API queries to LG-specific formats.
• The **LG Client** executes the native HTTP queries, and returns the raw HTML responses for parsing by the Query Interpreter.

What about rate limits?
API requests are not executed immediately, but first queued in the **Measurements DB** as pending jobs.

The **Controller** oversees the rate limits and decides when to allocate the pending jobs to the Query Interpreter.
Periscope enforces per-user and per-LG query rate limits

- LGs may communicate their limits explicitly (through disclaimers), or implicitly (through error codes).
- Two limits control the rate of issued LG queries:
  - User-specific: Each user can issue only 1 query per 5 minutes to the same LG.
  - LG-specific: Each LG will execute up to 3 queries per minute from all the users.
- A query is allocated if neither limit is exceeded.
- Exponential back-off when LGs respond with errors.
Support for multiple concurrent users requires multiple LG clients

Native LG querying

LGs use the users’ IP address to impose per-user querying quotas

Single-client Periscope

Putting multiple Periscope users behind the same IP causes all the users to share the quotas of a single user

Multi-client Periscope

Using different client per user allows Periscope to provide the same querying quotas as native querying
For each Periscope User the controller allocates a different cloud-hosted **VM instance** to execute the user queries.
Each VM instance takes an IP address from the cloud operator’s address space.
The same rate limits are still enforced.
Transparency of Periscope requests

• Periscope sets three custom HTTP request headers:
  o "X-Request-Origin: periscope"
  o "X-Request-For:<user-ip>"
  o "X-Request-Client:<gcloud OR aws OR ark>"

• IP addresses used by Periscope LG Clients are configured with the appropriate reverse DNS record:
  o client.periscope.lg
User accountability

• Periscope uses a 1-to-1 mapping between users and LG Client IP addresses:
  o A static VM Instance corresponds to each user.
  o Each VM Instance is assigned with a static IP address.
  o User-specific blocking works in the same way as native LG querying.

• Periscope maintains historical logs for every measurement:
  o Violations can be traced back to the responsible users
Coverage of Periscope LGs

- **Monitors:**
  - 572 ASNs with 2,951 VPs.
  - 77 countries, 492 cities.

- **To geo-locate LGs:**
  - Use locations encoded in LG interfaces.
  - Geo-locate the source IP of the LG using NetAcuity.
Commands supported by Periscope LGs

- Over 75% of the LG nodes provide both traceroute and BGP commands.
- Over 60% of the LGs support IPv6 queries.
LGs capture largely complementary topology compared to other platforms.

Topology obtained after querying 2,000 randomly selected IPs from each LG (2K VPs), and from every VP available in RIPE Atlas (8K probes) and CAIDA’s Ark (100 VPs) during August 2015.
Unique ASes in each dataset differ in terms of customer-cone sizes

- LGs tend to capture more peripheral and stub ASes.
- Ark and Atlas capture ASes with larger customer cone.
Intelligent Load Distribution

• Some LGs receive higher query loads than other LGs:
  o LGs in large providers receive more queries than their customers.

• Some queries can be satisfied by multiple LGs:
  o “What is the path between Level3 and Cloudflare?”
  o The path can be returned either by the Level3 LG, or by an LG that reaches Cloudflare through Level3

• Some queries can be satisfied by multiple platforms:
  o Overlap of vantage points among Atlas, Ark and Periscope.
Intelligent Load Distribution

**AS3356 Looking Glass**
(high query load)

**AS680 Looking Glass**
(low query load)
Utility Optimization

- If the same path can be satisfied by multiple LGs, allocate the query to the LG with the lowest load.
- Learn which LGs use the same paths through historical measurements and CAIDA’s AS relationships dataset.
- Use historical measurements from multiple platforms to improve the selection of alternative LGs.

4x increase in satisfied queries for the same querying budget

In native LG querying users are not aware of archived and concurrent queries:
  o Redundant queries to popular destinations (e.g. 8.8.8.8)*.
Periscope has a bird’s-eye view of measurements across users and across LGs:
  o Redundant queries are bundled together to reduce query load.
Measurement results are made publicly available:
  o Users can consume completed queries instead of issuing new.

* https://www.reddit.com/r/sysadmin/comments/1f9kv4/what_are_some_public_ips_that_are_ok_to/
  http://superuser.com/questions/769005/what-is-a-external-reliable-ip-address-to-ping-to-check-if-internet-is-available
Case study 1: Prefix Hijack Detection

- Combining Periscope with passive BGP collectors enables the detection of prefix hijacks in less than 1 minute.
- Faster detection in 60% of the hijack cases compared to using only BGPMon.

Case study 2: Troubleshooting Network Disruptions

- **NetDiagnoser (ND):** Identify the location of network failures through distributed traceroutes:
  - Unresponsive hops (*) or private IP address can cause errors

- Combine BGP and traceroute data from LGs to infer the ASes of the unidentified hops:
  - Up to 60% improvement in correct diagnoses in simulations

Periscope enables the practical implementation of the simulation-based methodology

Current status

• Periscope is accessible after email request: periscope-info@caida.org
• API documentation: http://www.caida.org/tools/utilities/looking-glass-api/
• Ongoing and future work:
  o Development of a graphical user interface.
  o Hosting of Periscope clients inside Ark monitors.
  o Improve optimization of LG utilization through cross-platform interoperability.
Request for Contributions

• Please contribute feedback regarding:
  1. Per-user query limits
  2. Global query limits
  3. Opt-in or opt-in requests
• Utilization statistics and archived queries.
• Infrastructure support (e.g. VM instances, cloud-computing credit).

Contact us at:
periscope-info@caida.org
Conclusion

• Periscope goals:
  o Unify LGs under a standardized overlay API.
  o Enforce per-user and per-LG rate limits.
  o Provide transparency and accountability.

• Benefits of Periscope:
  o Extends topology coverage.
  o Optimizes LG utilization.
  o Improves troubleshooting capabilities.
BACKUP SLIDES
Periscope uses path prediction to optimize query distribution

- Periscope is based on SIBYL to predict which LGs will return the same path, and selects the LG with the lower query load:
  - Use *previously issued measurements* to predict *unmeasured* paths.
  - Use *path splicing* when there are no archived measurements for the all the possible (source,destination) pairs.
  - Use *RuleFit* to assess the confidence in the predicted paths.

Optimization of Load Distribution across measurement platforms

Traceroute to caida.org (192.172.226.78), 48 byte p

1 193.174.247.1 1.827ms
2 188.1.239.45 7.658ms
3 62.40.124.217 7.173ms
4 62.40.125.18 1.047ms
5 198.71.45.16 1.157ms
6 198.71.45.13 1.157ms
7 198.71.45.21 1.157ms
8 137.164.26.200 1.747ms
9 137.164.26.34 1.763ms
10 192.12.207.10 1.7649ms
11 192.172.226.78 176.606ms

Type escape sequence to abort.
Tracing the route to ns1.caida.org (192.172.226.78)
VRF info: (vrf in name/id, vrf out name/id)
1 xr-fzk1-pc2.x-win.dfn.de (188.1.145.81) [MPLS: Label 1274]
2 cr-fra2-be9.x-win.dfn.de (188.1.144.121) 4 msec 8 msec 4 msec
3 dfn.mxl.fra.de.geant.net (62.40.124.217) [AS 20965] 4 msec
4 internet2-gw.mxl.fra.de.geant.net (62.40.125.18) [AS 20965]
5 et-7-3-0.4072.rtsw.atl.net.internet2.edu (198.71.45.6) []
6 et-10-2-0.105.rtr.hous.net.internet2.edu (198.71.45.13) []
7 et-7-1-0.4070.rtsw.losa.net.internet2.edu (198.71.45.21) []
8 hpr-lax-hpr2--l2-r&e.cenic.net (137.164.26.200) [AS 2153]
9 hpr-sdsc-10ge--lax-hpr.cenic.net (137.164.26.34) [AS 2153]
10 medusa-mx960.sdsc.edu (192.12.207.10) [AS 195] 180 msec 176 msec
11 ns1.caida.org (192.172.226.78) [AS 1909] 188 msec 176 msec

AS680  RIPE Atlas probe
AS680  Looking Glass